



**UNIVERSIDAD NACIONAL AUTÓNOMA  
DE MÉXICO**

---

---

**FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN**

**DESARROLLO DE SOFTWARE PARA GRAFICAR Y ANALIZAR  
DATOS DE DETECTORES DE PARTÍCULAS SOLARES**

**T E S I S**

**QUE PARA OBTENER EL TÍTULO DE:**

**INGENIERO EN COMPUTACIÓN**

**P R E S E N T A:**

**TRUJILLO TORRES OSVALDO CESAR**



**DIRECTOR DE TESIS:**

**M. en C. Marcelo Pérez Medel**

**MÉXICO 2015**

Ciudad Nezahualcóyotl, Estado de México



Universidad Nacional  
Autónoma de México



**UNAM – Dirección General de Bibliotecas**  
**Tesis Digitales**  
**Restricciones de uso**

**DERECHOS RESERVADOS ©**  
**PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

## DEDICATORIAS

A mis padres, **Isabel** y **Octavio** a quienes amo profundamente, por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo perfectamente mantenido a través del tiempo y sobre todo en estos años en la universidad.

Todo este trabajo ha sido posible gracias a ellos.

A mis hermanos, **Octavio** y **Omar** quienes de una u otra forma han colaborado e impulsado a conseguir este logro.

A mi abuelita **Manuela**, que siempre la llevaré en mi corazón y que no está aquí físicamente para celebrar este triunfo conmigo, pero sé que desde el cielo siempre me cuida y me envía su bendición para que todo salga bien.

## **AGRADECIMIENTOS**

A la **UNAM**, por brindarme una excelente educación pública, laica y gratuita.

Al **Dr. Luis Xavier González Méndez** por la oportunidad, la confianza y el apoyo total brindado a lo largo de este proyecto.

A mis sinodales, por darme la oportunidad y por el tiempo que me han dedicado para leer este trabajo.

A mis amigos **Ulises Carrillo** y **Braulio García** por compartir los buenos y malos momentos conmigo durante todos estos años.

Gracias a toda mi familia y amigos que no están aquí, pero que me ayudaron a que este gran esfuerzo se volviera realidad.

# CONTENIDO

<b>INTRODUCCIÓN</b> .....	8
<b>CAPÍTULO 1: EMISIONES Y ACTIVIDAD SOLAR</b> .....	13
1.1 INTRODUCCIÓN .....	14
1.2 EL SOL .....	14
1.2.1 Interior solar.....	16
1.2.1.1 Núcleo .....	17
1.2.1.2 Zona radiativa.....	17
1.2.1.3 Zona convectiva .....	18
1.2.2 Atmósfera solar .....	18
1.2.2.1 Fotósfera .....	19
1.2.2.2 Cromósfera.....	20
1.2.2.3 Corona.....	21
1.2.2.4 Viento solar .....	23
1.3 ACTIVIDAD SOLAR.....	25
1.3.1 Ciclo de manchas solares.....	25
1.3.2 Regiones altamente activas.....	26
1.3.3 Fulguraciones solares.....	27
1.3.4 Emisiones de las fulguraciones .....	29
1.3.4.1 Rayos X Suaves .....	29
1.3.4.2 Rayos X Duros .....	30
1.3.4.3 Rayos Gama .....	31
1.3.4.4 Ultravioleta .....	32
1.3.4.5 Óptico .....	33
1.3.4.6 Radio.....	34
1.3.5 Tipos de fulguraciones y su clasificación.....	35
1.3.6 Liberación y transporte de energía en fulguraciones .....	36
1.3.6.1 Disipación de energía en fulguraciones.....	37
<b>CAPÍTULO 2: MONITOR DE NEUTRONES DE LA CIUDAD DE MÉXICO</b> .....	39

2.1 INTRODUCCIÓN .....	40
2.2 MONITOR DE NEUTRONES (6-NM64) .....	41
2.3 DISEÑO .....	42
2.3.1 Contador proporcional de BF <sub>3</sub> .....	43
2.3.2 Moderador .....	43
2.3.3 Productor .....	44
2.3.4 Reflector .....	44
2.4 ADQUISICIÓN DE DATOS .....	46
<b>CAPÍTULO 3: LENGUAJE DE PROGRAMACIÓN PYTHON .....</b>	<b>47</b>
3.1 HISTORIA .....	48
3.2 CARACTERÍSTICAS .....	49
3.3 FILOSOFÍA .....	50
3.4 MODO INTERACTIVO .....	51
3.5 ELEMENTOS DEL LENGUAJE .....	52
3.5.1 Variables .....	52
3.5.2 Tipos de datos .....	53
3.5.3 Tipos de datos complejos .....	55
3.5.3.1 Listas .....	55
3.5.3.2 Tuplas .....	55
3.5.3.3 Diccionarios .....	55
3.5.4 Funciones .....	56
3.5.5 Clases .....	58
3.5.6 Estructuras de control de flujo .....	59
3.5.6.1 Identación .....	59
3.5.6.2 Operadores .....	60
3.5.6.3 Estructuras de control condicionales .....	61
3.5.6.4 Estructuras de control iterativas .....	63
3.5.7 Comentarios .....	63
3.6 PROGRAMACIÓN ORIENTADA A OBJETOS .....	65
3.6.2 Clases, atributos y métodos .....	66
3.6.3 Objetos .....	68

3.6.4 Herencia .....	69
3.6.5 Encapsulamiento .....	71
3.6.6 Polimorfismo .....	72
3.7 SUBPROCESAMIENTO MÚLTIPLE.....	73
3.7.2 Programación concurrente .....	74
3.7.3 Multithreads en Python .....	75
3.7.4 Multiprocessing en Python.....	78
3.7.5 Conclusión.....	79
3.8 INTERFACES GRÁFICAS DE USUARIO (GUI).....	80
3.8.2 Las GUI en Python .....	81
3.8.2.1 Tkinter .....	82
3.8.2.2 wxPython.....	84
3.8.2.3 PyQT .....	87
3.8.2.4 PyGTK.....	90
3.9 APLICACIONES DESARROLLADAS CON PYTHON .....	93
<b>CAPÍTULO 4: PYSMG (SOLAR MONITORS GRAPHER).....</b>	<b>94</b>
4.1 CATEGORÍAS DEL SOFTWARE LIBRE Y NO LIBRE.....	95
4.1.1 Free Software .....	96
4.1.2 Open Source Software .....	97
4.1.3 Diferencia entre Free Software y Open Source .....	99
4.2 DISEÑO E IMPLEMENTACIÓN DE PYSMG.....	101
4.2.1 Análisis de requerimientos del software .....	101
4.2.2 Diagrama de casos de uso .....	103
4.2.3 Diagrama de paquetes .....	106
4.2.3 Diagrama de clases .....	107
4.2.4 Diagrama de actividades .....	108
4.2.5 Diagrama de secuencia .....	110
4.2.6 Codificación .....	110
4.2.6.1 Lectura e interpretación del archivo de datos.....	111
4.2.6.2 Creando nuevos procesos.....	113
4.2.6.3 Generando las gráficas .....	114

4.3 USO DE PYSMG .....	118
4.3.1 Requerimientos del sistema .....	119
4.3.2 Ejecutando PySMG .....	121
<b>CAPÍTULO 5: ANÁLISIS DE RESULTADOS .....</b>	<b>128</b>
5.1 INTRODUCCIÓN .....	129
5.2 GRÁFICAS CON RATE DE 1 MINUTO Y 5 MINUTOS .....	130
5.3 GRÁFICAS CON RATE DE 30 MINUTOS Y 1 HORA .....	132
5.4 GRÁFICAS DE 1 DÍA.....	134
<b>CONCLUSIONES .....</b>	<b>135</b>
<b>APÉNDICES .....</b>	<b>139</b>
<b>GLOSARIO .....</b>	<b>160</b>
<b>BIBLIOGRAFÍA .....</b>	<b>164</b>



# **INTRODUCCIÓN**

El Instituto de Geofísica (IGEF) es una dependencia del Subsistema de la Investigación Científica de la Universidad Nacional Autónoma de México (UNAM), abrió sus puertas en 1949 una vez que el H. Consejo Universitario aprobara su creación en el año de 1945.

Este Instituto apoya la investigación, docencia, difusión y divulgación científica, así como difundir y promover la superación académica de su personal; a lo largo del tiempo se han desarrollado nuevas disciplinas y grupos de trabajo que cubren gran parte de las Ciencias de la Tierra y Espaciales. Actualmente está formado por seis departamentos, una sección, un conjunto de observatorios y laboratorios, 3 servicios geofísicos nacionales y de apoyo académico:

**Departamento de Ciencias Espaciales,**

Departamento de Geomagnetismo y Exploración,

Departamento de Recursos Naturales,

Departamento de Sismología,

Departamento de Vulcanología y

Sección de Radiación Solar.

El Departamento de Ciencias Espaciales del IGEF de la UNAM, creado en 1962 por la investigadora polaca Ruth Gall, Se dedica al estudio de la dinámica, la estructura y las propiedades del Sistema Solar; teniendo como principales áreas de investigación la Física Solar, la Física de Rayos Cósmicos, la Física Magnetosférica y la Física del Medio Interplanetario.

El grupo de Rayos Cósmicos encargado del observatorio de rayos cósmicos ubicado en Ciudad Universitaria, forma parte de una red de más de 50 observatorios distribuidos por todo el mundo y para el cual se desarrolla este software.

El Observatorio de rayos cósmicos de la Ciudad de México detecta las partículas que viajan desde el espacio y que diariamente llegan a la tierra desde todas las direcciones. Los datos que arroja el sistema son recolectados por un sistema de adquisición de datos que se desarrolló por el personal técnico del departamento a cargo de este Observatorio. El sistema almacena la información y tiene la capacidad de mostrarlos en tiempo real.

Sin embargo, surge la necesidad de poder visualizar e interpretar la información vertida por el sistema de adquisición de datos de forma gráfica y que realice cálculos de estadística básica para el apoyo de los investigadores del departamento.

El Dr. Luis Xavier González Méndez encargado del proyecto para la interpretación y graficación de datos, es quién me da la oportunidad de realizar mi servicio social desarrollando dicho software con las características que se necesitan para la investigación como código claro y de fácil lectura, multiplataforma, de uso intuitivo y sobre todo que sea bajo licencia de software libre.

La hipótesis que sustenta esta tesis, plantea que se puede desarrollar un software de propósito específico capaz de satisfacer las necesidades del departamento a partir de herramientas de propósito general, que sea de uso amigable y bajo la licencia de software libre.

Este trabajo tiene como objetivo general, brindar al departamento de Rayos Cósmicos una herramienta para la interpretación de forma gráfica de los datos arrojados por el Monitor de Neutrones (MN) de la Ciudad de México.

Asimismo, el objetivo específico es que a partir de la interpretación gráfica de los datos, se optimice el análisis de los fenómenos solares y agilizar la generación de reportes por medio de gráficas con una cantidad de datos considerable (1 año  $\approx$  525, 600 datos), así como la precisión de datos de 1 min.

La metodología principal para el desarrollo del software es la Programación Orientada a Objetos, ya que es el paradigma de programación más utilizado en los últimos años, forma parte de la filosofía del lenguaje Python y permite una fácil lectura y mantenimiento de los archivos fuente además de la ventaja que ofrece al poder reutilizar código.

De esta manera, en los próximos capítulos, describiré la forma en que realicé el desarrollo del software que se encuentra en producción y que ha arrojado interesantes resultados que también se abarcarán en este documento.

En el capítulo uno describo la estructura del Sol, tanto su interior como su exterior. Una parte importante de este proyecto tiene que ver con la actividad solar, las fulguraciones y la forma en la que las emite, ya que las partículas generadas por estos fenómenos solares llegan a nuestro planeta en todo momento.

Debido a esto, el capítulo dos está dedicado a los detectores de partículas solares en la UNAM. El Observatorio de rayos cósmicos de la Ciudad de México detecta el fondo de rayos cósmicos galácticos y las partículas energéticas solares cuando hay explosiones en el Sol.

El capítulo tres lo dedico al lenguaje de programación Python que aunque relativamente es nuevo, ha ganado mucho prestigio y popularidad debido a su potencia, es de alto nivel, independiente de plataforma y de propósito general. Se pueden desarrollar desde aplicaciones GNU Linux / Windows hasta servidores de red.

En el capítulo cuatro explico varios puntos importantes para el desarrollo de PySMG. En primer lugar, la filosofía de desarrollo de software libre y de código abierto, también la Programación Orientada a Objetos que es el paradigma más usado en los últimos años. A diferencia de otros lenguajes de programación, en Python tenemos varias librerías gráficas de donde escoger, debido a esto describo las cuatro librerías más importantes para el desarrollo de interfaces gráficas de usuario. Otro aspecto crucial para el desarrollo del software es el subprocesamiento múltiple, en otros lenguajes la decisión sería trivial, sin embargo

en este capítulo veremos algunas características de Python que lo vuelven difícil. Al final del capítulo describo el uso básico del software, así como la descripción de los menús y herramientas.

Un análisis más profundo lo podemos realizar con datos reales arrojados por el Monitor de Neutrones. Además describiré algunos eventos que fueron detectados con este observatorio en el capítulo cinco.

Finalmente en los apéndices se muestra la licencia completa de PySMG y el manual de usuario.

**CAPÍTULO 1:**  
**EMISIONES Y ACTIVIDAD**  
**SOLAR**



## 1.1 INTRODUCCIÓN

*"Todos los efectos de la naturaleza son sólo la consecuencia matemática de un pequeño número de leyes inmutables".  
Pierre Simon Laplace.*

El funcionamiento de este software está basado en la graficación de los datos arrojados por el Observatorio de Rayos Cósmicos de la Ciudad de México, que recaba la cantidad de partículas solares que llegan a la tierra provocada por la actividad solar.

En este capítulo se describe la composición solar, la actividad del sol y las fulguraciones que provoca para poder entender mejor el área de trabajo en el que se desenvuelve PySMG.

## 1.2 EL SOL

Es una de las  $15 \times 10^{10}$  estrellas que componen la Vía Láctea y se encuentra en un brazo, girando alrededor de la Galaxia con un periodo aproximado de  $250 \times 10^6$  años. Es la estrella más grande del sistema solar, contiene el 99.8% de la masa total del sistema planetario. Su distancia promedio a la Tierra, denominada como Unidad Astronómica (U.A.), es de ~150 millones de Kilómetros (máxima 152,106 millones de Km. y mínima 143,103 millones de Km), esto, debido a que la órbita de la Tierra es elíptica y no mantiene una distancia fija. El Sol está compuesto de un 92.1% de Hidrógeno y 7.8% de Helio. Otros elementos químicos más pesados no sobrepasan el 0.1%. La tabla 1.1 muestra el porcentaje de los diez elementos más abundantes en el Sol, (González, 2008).

**Tabla de elementos en el Sol.**

<b>H</b>	<b>He</b>	<b>C</b>	<b>N</b>	<b>O</b>	<b>Ne</b>	<b>Mg</b>	<b>Si</b>	<b>S</b>	<b>Fe</b>
92.1	7.8	0.043	0.0088	0.078	0.0035	0.0038	0.0045	0.015	0.0030

**Tabla 1.1.** Porcentaje de los diez átomos más abundantes en el Sol, (Lang, 2008). 2008.

Debido a su alta temperatura y a pesar de tener una densidad elevada, el núcleo solar es un plasma<sup>1</sup>. El Sol rota sobre sí mismo con un eje norte-sur perpendicular a la Tierra, con una inclinación de 7 grados respecto al plano de la órbita, gira en la misma dirección que la Tierra y se le definen un ecuador, meridianos y paralelos.

El tamaño del Sol se puede calcular a través de su distancia y apertura angular, la masa se puede determinar con precisión con base en las órbitas de los planetas, y la edad es determinada por el estudio de la abundancia de elementos radioactivos pesados en los meteoritos. En la tabla 1.2 se muestran los parámetros físicos representativos del Sol.

### Generalidades del Sol.

<b>Volumen</b>	$1.412 \times 10^{27} \text{ m}^3$ (1.3 millones Tierras)
<b>Luminosidad</b>	$3.854 \times 10^{26} \text{ W}$
<b>Masa</b>	$1.989 \times 10^{30} \text{ Kg}$ (332,946 masas terrestres)
<b>Radio</b>	$6.955 \times 10^8 \text{ m}$ (109 radios terrestres)
<b>Distancia media al sol</b>	$1.4959787 \times 10^{11} \text{ m}$
<b>Temperatura en la fotosfera</b>	5780 K
<b>Temperatura en la corona</b>	$2-3 \times 10^6 \text{ K}$
<b>Temperatura en el núcleo</b>	$15.6 \times 10^6 \text{ K}$
<b>Densidad media</b>	$1409 \text{ kg m}^{-3}$
<b>Densidad en el núcleo</b>	$151300 \text{ kg m}^{-3}$
<b>Presión en el núcleo</b>	$2.334 \times 10^{16} \text{ Pa}$
<b>Presión en la fotosfera</b>	10 Pa
<b>Constante solar</b>	$1361 \text{ W m}^{-2}$
<b>Edad</b>	$4.55 \times 10^9$ años

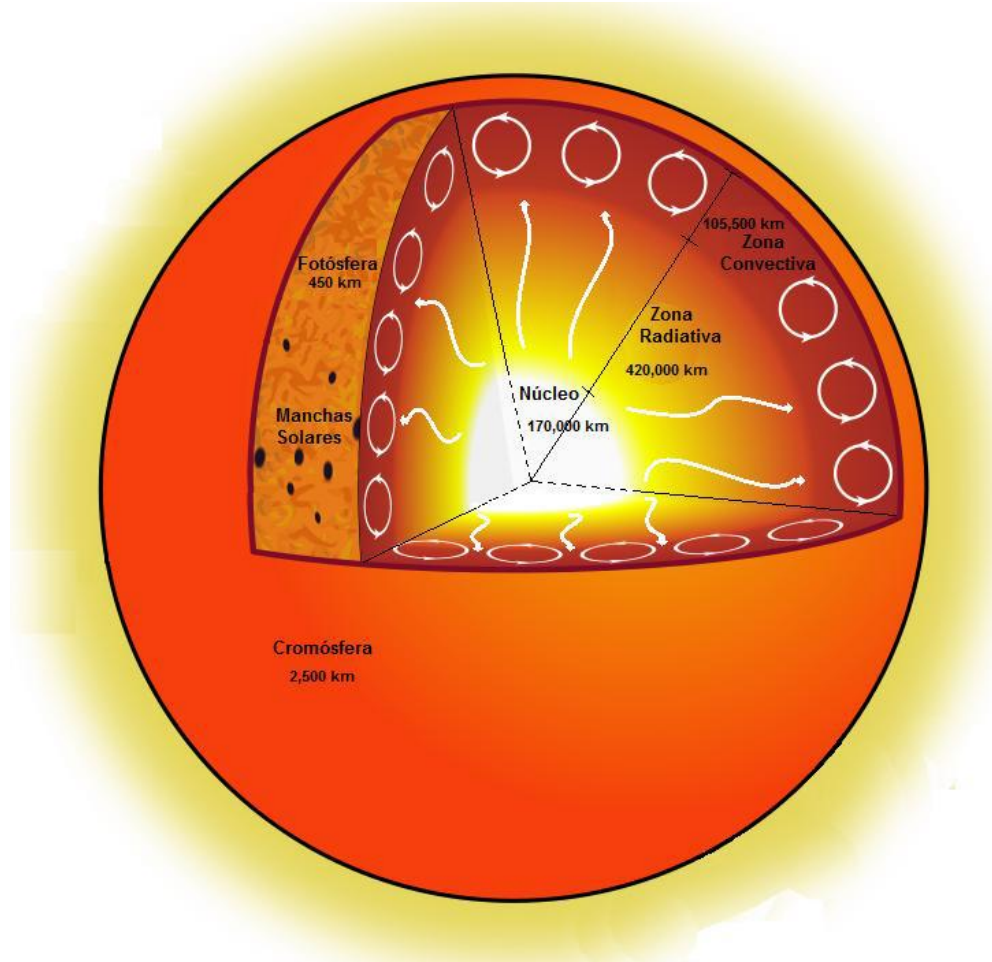
**Tabla 1.2.** Parámetros físicos en el Sol, (Lang, 2008). (2008).

<sup>1</sup> El plasma es un gas totalmente ionizado, que consta de igual número de iones y electrones, por lo que es cuasi neutro en estado estacionario y cuya dinámica presenta efectos colectivos dominados por las interacciones electromagnéticas.



### 1.2.1 Interior solar

El modelo que ha evolucionado a lo largo de los años con el fin de entender el Sol se muestra en la figura 1.1. En el esquema se puede observar que el núcleo es la fuente de energía del Sol, después del núcleo, dos terceras partes del interior solar corresponden a la zona radiativa, donde se transporta la energía desde el núcleo por la radiación, posteriormente se encuentra la zona convectiva, donde se lleva a cabo el proceso de convección que es el principal mecanismo de transporte de energía.



**Figura 1.1.** Esquema del interior solar.

**Fuente:** <http://www.nasaimages.org>. 2008

### **1.2.1.1 Núcleo**

El núcleo corresponde aproximadamente al 2% del volumen del Sol, sin embargo contiene aproximadamente la mitad de la masa. Su densidad es 15 veces la densidad del plomo. Aquí se realizan las reacciones nucleares exotérmicas<sup>2</sup> que proporcionan la energía que el Sol produce, (González, 2008).

De acuerdo con (González, 2008), la reacción  $p^+ - p^+$  (protón - protón) corresponde al 99% de la fusión en el Sol, donde cuatro protones se fusionan en un núcleo de Helio, liberando 2 protones, 2 neutrinos y 2 gamas. La energía total liberada en la formación de un núcleo de Helio es de 26.7 MeV<sup>3</sup>. La cadena  $p^+ - p^+$  necesita una temperatura de al menos  $5 \times 10^6$  K. El ciclo CNO (Carbono-Nitrógeno-Oxígeno) genera sólo el 1% de la energía en el Sol.

### **1.2.1.2 Zona radiativa**

Se encuentra rodeando el núcleo, entre 0.25 y 0.8 radios solares. La energía producida en el núcleo es transportada hasta aquí por fotones a través del proceso de radiación. Constituye el 32% del volumen del Sol y 48% de su masa.

En la parte inferior, la densidad es de  $\sim 22 \text{ g cm}^{-3}$  y la temperatura es de  $\sim 8 \times 10^6$  K, mientras que en la parte superior, la densidad baja hasta  $\sim 0.2 \text{ g cm}^{-3}$  y la temperatura es de  $\sim 2 \times 10^6$  K.

Entre la zona radiativa y la zona convectiva existe una capa de transición llamada *tachocline*, donde el régimen cambia entre la rotación uniforme de la zona radiativa y la rotación diferencial de la zona convectiva, (González, 2008).

---

<sup>2</sup> Cualquier reacción química que desprenda energía, ya sea como luz o calor.

<sup>3</sup> El Electronvoltio (eV) es la unidad de energía que representa la variación de energía potencial que experimenta un electrón al moverse desde un punto de potencial  $Va$  hasta un punto  $Vb$ .

1 MeV =  $10^3$  KeV =  $10^6$  eV.

### **1.2.1.3 Zona convectiva**

Se encuentra entre 0.8 y 1 radios solares y consiste de células de convección en ebullición. Constituye el 66% del volumen del Sol, pero sólo ~2% de su masa. Como ya se mencionó, el transporte de energía se da por el proceso de convección. En la parte alta, la densidad es  $\sim 2 \times 10^{-7} \text{ g cm}^{-3}$  (la densidad prácticamente no cambia) y la temperatura es  $\sim 5800 \text{ K}$ , mientras que en la parte más baja, la temperatura es  $\sim 2 \times 10^6 \text{ K}$ . Esta diferencia de temperaturas hace que el plasma caliente suba y el frío baje, formando corrientes en forma de columnas térmicas que arrastran el material caliente hacia la fotosfera del Sol. Estas columnas térmicas se muestran en la superficie solar en forma de celdas de granulación y supergranulación. Las celdas de granulación son de alrededor de 1000 km de diámetro, mientras que las de supergranulación pueden alcanzar los 30,000 km de diámetro.

### **1.2.2 Atmósfera solar**

La atmósfera solar es la parte del Sol que se extiende hacia el exterior de la zona convectiva, donde la energía generada en el núcleo del Sol empieza a escapar hacia el espacio en forma de radiación. En la Atmósfera solar se pueden identificar tres estructuras, la fotosfera que es el disco aparente del Sol, la cromósfera y la corona que es la más externa, las dos últimas son visibles únicamente durante los eclipses naturales o artificiales del disco solar.

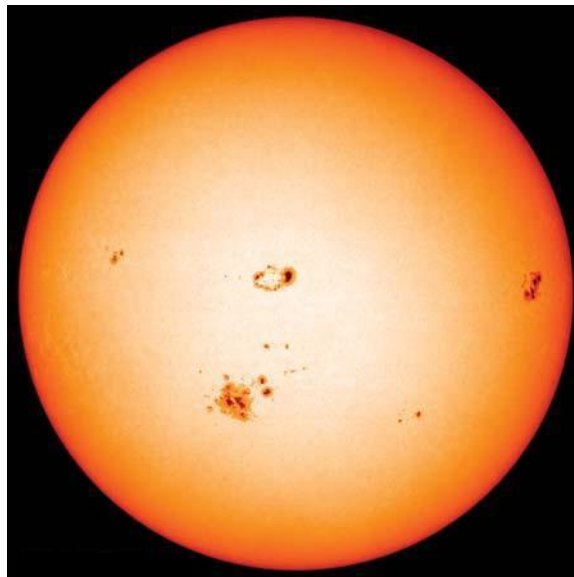
Entre la cromósfera y la corona hay una capa de menor espesor, llamada región de transición. La temperatura de la atmósfera solar disminuye hacia fuera en la fotosfera, alcanzando un valor mínimo de  $\sim 5,780 \text{ K}$ ; después aumenta con la altura en la cromósfera, región de transición y corona, finalmente, hacia el exterior, en el espacio interplanetario la temperatura (T) del viento solar se mantiene casi constante.

### **1.2.2.1 Fotósfera**

Es la superficie visible del Sol y la capa más baja de la atmósfera. Tiene ~300 km. de espesor, ~0.04% del radio solar y emite casi la totalidad del visible. Su densidad y temperatura aumentan en el interior, donde toda la radiación se absorbe, impidiendo la observación de las capas más profundas. La temperatura de la fotosfera es de alrededor de 4,000 K a 6,400 K.

La fotosfera no es uniforme, en su superficie se encuentran las manchas solares (figura 1.2). Una mancha solar consta de una región central oscura denominada sombra o umbral, rodeada por una zona más clara o penumbra consistente en filamentos claros y oscuros que parten de forma aproximadamente radial de la sombra, (González, 2008). En promedio el diámetro de la penumbra suele ser unas dos veces y media mayor que el de la sombra, pero en grupos muy desarrollados puede llegar a representar hasta el 80% del total de la extensión de la mancha. Si la mancha no posee penumbra, se le denomina poro.

Según (González, 2008), el tamaño de las manchas varía, desde poco más de un millar de kilómetros (poro aislado) hasta más de 100,000 kilómetros en los grupos bien desarrollados. Las manchas aparecen en grupos; típicamente un grupo consiste en dos manchas de polaridad magnética opuesta.

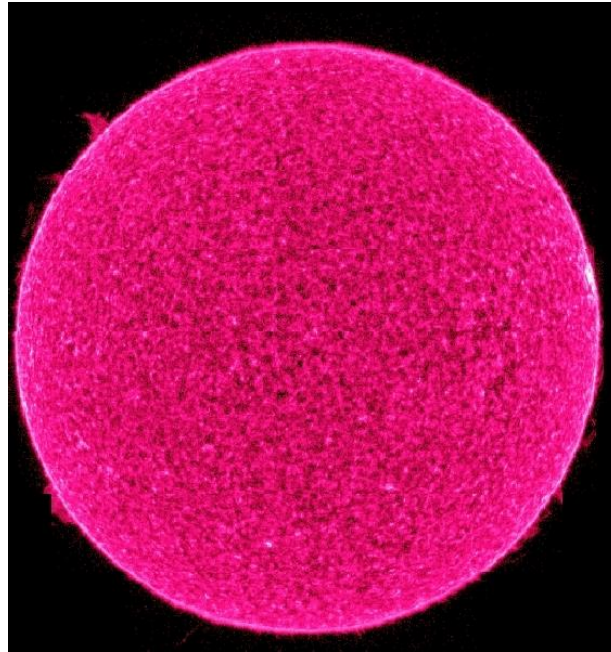


**Figura 1.2.** Manchas solares en la fotosfera solar

**Fuente:** <http://www.nasaimages.org/>. 2008.

### 1.2.2.2 Cromósfera

Es la capa de la atmósfera situada por encima de la fotosfera (figura 1.3). Es muy tenue, de unos 2,000 km. de espesor. Presenta un color rojizo cuando es visible debido a que su emisión es dominada por el  $H\alpha^4$  al principio y al final de un eclipse de Sol.



**Figura 1.3.** Cromósfera solar. Línea de emisión a 1548 Å y  $10^5$  K. Imagen compuesta, tomada por el satélite SOHO.

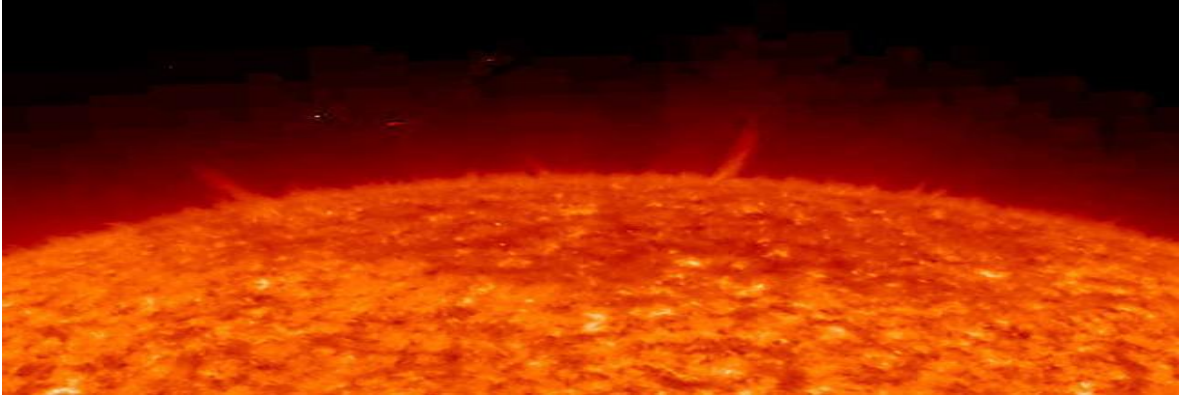
**Fuente:** <http://sohowww.nascom.nasa.gov>. 2008.

Su aspecto no es homogéneo, está compuesta de estructuras heterogéneas, llamadas espículas (figura 1.4), que ascienden y descienden a una velocidad del orden de 20 km/s.

Las espículas adoptan la forma de cilindros casi verticales de gas cromosférico, de unos 700 km. de diámetro y hasta 150,000 km. de altura, tienen una vida media de 5 a 15 minutos y en un momento dado puede haber medio millón.

---

<sup>4</sup> H-alfa es una de las líneas de emisión del espectro del Hidrógeno.



**Figura 1.4.** Espículas en la cromósfera solar, (<http://www.nasaimages.org/>).

La temperatura de la cromósfera es más alta que en la fotosfera, es de 4,500 K hasta un máximo de 20,000 K. Por encima de la cromósfera, hay una región de transición, donde la temperatura aumenta rápidamente hasta la corona, que forma la parte exterior de la atmósfera solar.

### **1.2.2.3 Corona**

La corona es la región más externa de la atmósfera solar, que ocupa millones de kilómetros y es vista como un halo blanco extendido durante un eclipse (figura 1.5), transformándose gradualmente en viento solar.

Hay tres componentes principales de la luz coronal: corona-K, corona-F y corona-E. La K está formada de luz fotosférica, dispersada por los electrones coronales. La corona-F es resultado de la dispersión de luz fotosférica por partículas de polvo. La corona-E es la fuente de las líneas de emisión de átomos altamente ionizados. En el mínimo solar, la corona tiene largas corrientes simétricas en el ecuador y plumas en los polos (figura 1.5). Durante el máximo solar, hay muchas estructuras, indicando un campo magnético complejo, que elimina la simetría de la corona.



**Figura. 1.5.** Superior: corona solar cercana al mínimo de actividad (2006). Inferior: corona solar durante el máximo de actividad (1999),

**Fuente:** <http://www.nasaimages.org/>. 1999, 2006.

La energía necesaria para calentar la corona es alrededor del 0.01% de la emisión luminosa total del Sol, sin embargo, el mecanismo no es totalmente conocido. Existen dos tipos de mecanismos propuestos para el calentamiento de la corona, modelos de Corriente Alterna: disipación de ondas y modelos de Corriente Directa: disipación de campos magnéticos intensos.

Una característica fundamental de la corona son los hoyos coronales. Los hoyos coronales son regiones de plasma de baja densidad en el Sol, que tienen líneas de campo magnético que se abren libremente en el espacio interplanetario. Cuando existe baja actividad solar, los hoyos coronales cubren las capas polares del norte y sur del Sol. Durante los períodos más activos, los hoyos coronales pueden existir en todas las latitudes solares, pero sólo pueden mantenerse durante algunas rotaciones solares, antes de evolucionar a una configuración magnética diferente. Los iones y electrones fluyen a lo largo de las líneas de campo magnético abiertas en los hoyos coronales, para formar la componente de alta velocidad del viento solar.

#### **1.2.2.4 Viento solar**

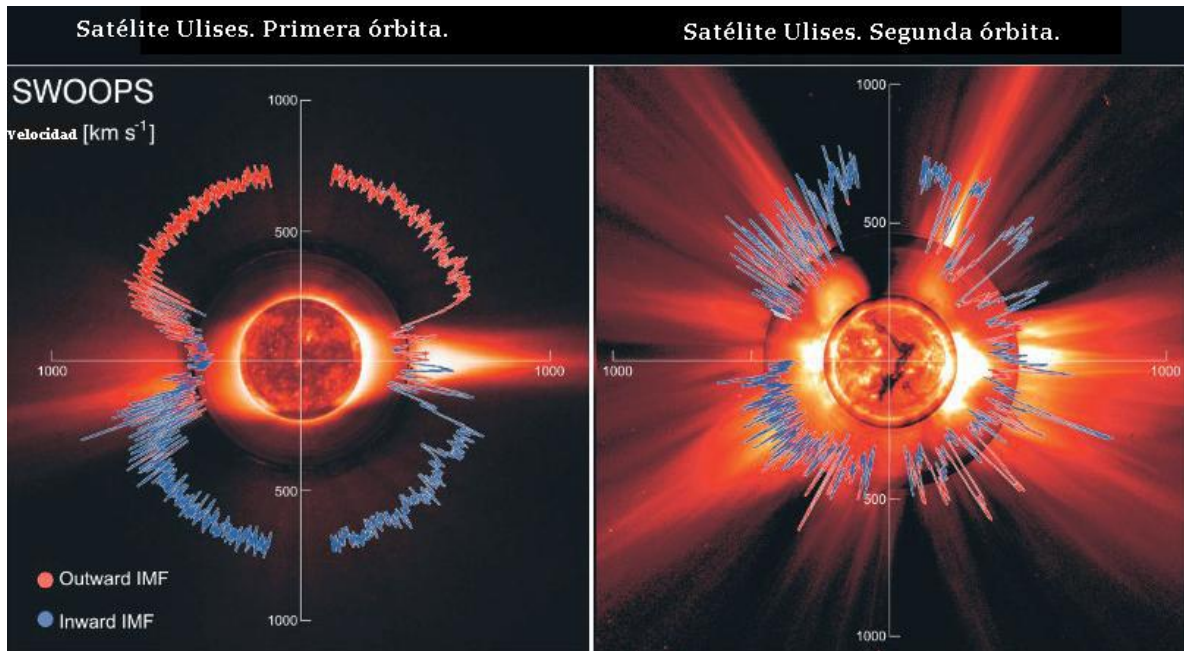
A pesar de la elevada temperatura, la energía total almacenada en la corona es muy pequeña, debido a que el gas es muy tenue y la energía es liberada gradualmente bajo la forma de viento solar. El proceso está acompañado por una pérdida de gas que es reemplazado con el que emerge de la cromósfera.

Hay dos componentes para el viento solar. Una componente rápida, con velocidades de ~800 km/s y una lenta, con velocidades de 300 km/s. Con base en observaciones con satélites, se sabe que el viento solar rápido se origina, en su mayoría, en las regiones de baja densidad, conocida como hoyos coronales (figura 1.6).

Cerca de la Tierra la densidad del viento solar es típicamente de 5 a 10 partículas/cm<sup>3</sup> y la velocidad ~400 km/s. La pérdida de masa debido al viento solar es inferior a la 1/10<sup>9</sup> parte de la masa total del Sol.

La distribución espacial del viento solar varía dramáticamente durante el ciclo solar. El cinturón de flujo es la fuente del viento solar lento y los hoyos coronales son la fuente del viento solar rápido. El viento lento puede originarse en los hoyos coronales polares, los cuales se reducen en tamaño. La figura 1.6 muestra la velocidad del viento solar en todas las latitudes, donde el viento de alta velocidad está confinado a la región de alta latitud y el viento solar lento está confinado a las regiones ecuatoriales.





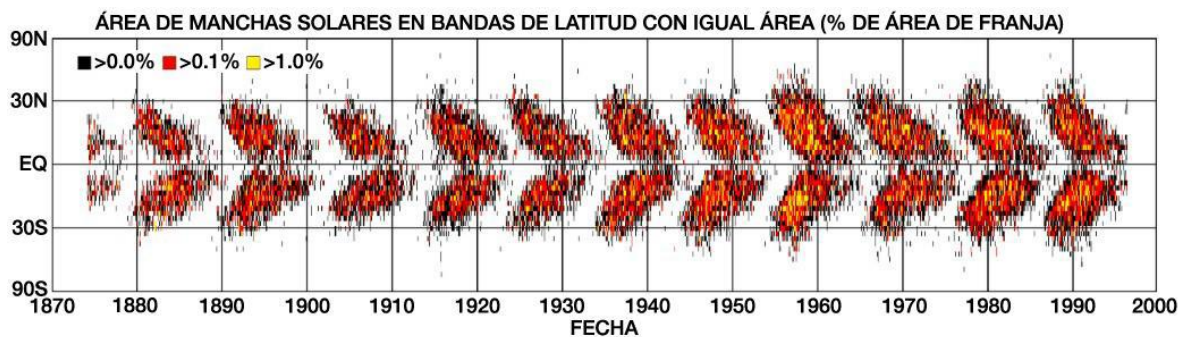
**Figura. 1.6.** Izquierda: distribución de la velocidad del viento solar en el mínimo.  
**Fuente:** <http://www.nasaimages.org/>. 1995.

En el mínimo de actividad solar, los hoyos coronales polares, con campos magnéticos abiertos, dan lugar a flujos de viento solar rápido y de baja densidad, mientras que los flujos de las regiones ecuatoriales, con campos magnéticos cerrados, están asociados con viento solar lento y denso. En el máximo solar, pequeños hoyos coronales de baja latitud generan viento rápido y una variedad de velocidades de vientos, que resultan en una pequeña variación latitudinal promedio de la velocidad del viento solar.

## 1.3 ACTIVIDAD SOLAR

### 1.3.1 Ciclo de manchas solares

El ciclo de manchas Solares tiene tres aspectos fundamentales, el periodo de 11 años del número de manchas, la ley de Hale-Nicholson de la polaridad de las manchas y la inversión del campo magnético global. El surgimiento de las manchas se da abruptamente y decae lentamente. Las primeras manchas de un ciclo aparecen cerca de las latitudes de  $30^{\circ}$  N y  $30^{\circ}$  S y las últimas cercanas al ecuador. Esta variación es llamada ley de Spörer, pero su regularidad se aprecia mediante el diagrama de mariposa de Maunder:



**Figura 1.7.** Diagrama de mariposa, donde se muestra la posición de las manchas solares durante los años de 1870 a 1995.

**Fuente:** <http://www.physics.unlv.edu/>. 1995.

En el diagrama de mariposa, las alas se superponen parcialmente, esto es que las manchas del viejo ciclo permanecen en el ecuador cuando el nuevo ciclo surge, cada ciclo dura entre 9 y 14 años.

El número de manchas solares en cada hemisferio puede ser muy diferente y uno de ellos puede dominar por varios años. En 1919 se descubrió que cada ciclo Solar sucesivo estaba marcado por una polaridad magnética distinta. Esto es conocido como la ley de Hale-Nicholson.

El grupo de manchas Solares, que son bipolares, aparecen en la dirección de rotación Este-Oeste con la misma polaridad magnética dirigida a un hemisferio y opuesta en el otro. La inversión del campo magnético es un semiciclo de 11 años, por lo tanto, el ciclo magnético del Sol es de 22 años. Todos los campos magnéticos intensos emergen a través del proceso de manchas Solares (rompimiento) y se difunden a través de la superficie para producir un campo global dipolar.

### 1.3.2 Regiones altamente activas

En 1919 se introdujo la clasificación magnética que permanece como la más importante:

- $\alpha$ : Una mancha dominante, normalmente conectada a una playa de polaridad magnética opuesta.
- $\beta$ : Un par de manchas dominantes de polaridad opuesta.
- $\gamma$ : Grupos complejos con una distribución irregular de polaridades.
- $\beta\gamma$ : Grupos bipolares sin una línea de inversión norte-sur marcada.
- $\delta$ : Umbral de polaridad opuesta en una penumbra.

En 1960 Künzel mostró que los grupos  $\delta$  producen mucho más fulguraciones que los demás tipos de estructuras magnéticas. La alta actividad asociada con las manchas  $\delta$  es debido a que dos polos con fuertes campos opuestos son presionados.

Posteriormente en 1987 Zirin y Liggett encontraron que las manchas  $\delta$  se forman de tres modos:

1. Surgimiento de dipolos entrelazados y polaridades inversas. **Esto genera a las regiones más activas.**
2. Grandes dipolos emergen alrededor de las manchas ya existentes, tal que la expansión de la región de flujo emergente presiona a una mancha p dentro de una mancha f o a la inversa.
3. Un grupo de manchas dipolares surgiendo choca con otro dipolo.

El tercer tipo es el más frecuente, ocurriendo siempre que una región de flujo emergente surge en el lugar adecuado.

### **1.3.3 Fulguraciones solares**

Las fulguraciones solares se descubrieron entre 1859 y 1860 por Carrington y Hodgson debido a las observaciones de intensos brillantamientos en el espectro visible cercano a un grupo de manchas solares.

Las fulguraciones solares están asociadas con la rápida liberación de energía magnética almacenada. Una gran fulguración puede liberar hasta  $10^{34}$  de energía en algunos minutos, sobre un área aproximada de  $10^{22}$  m<sup>2</sup>.

Esta liberación de energía está asociada con la eyección<sup>5</sup> de  $10^{15}$  gr de materia a velocidades que pueden exceder los 100 Km/s.

La energía es almacenada en un periodo de horas a días en la configuración del campo magnético en las llamadas regiones activas, aunque aún no está determinado si estas regiones activas ocurren antes o después de que emerja el flujo de debajo de la superficie Solar. Todas las fulguraciones muestran incrementos en las líneas de emisión atómicas.

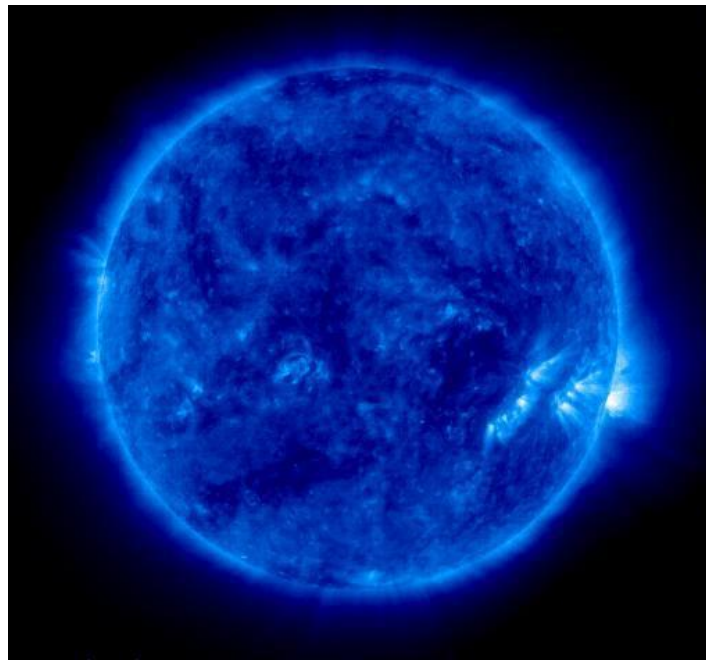
Las observaciones ópticas y en rayos X, son las técnicas clásicas para ver fulguraciones Solares, debido a la alta resolución y corta exposición, además que la mayor parte de la energía es liberada en la frecuencia del óptico. Otra técnica usada para observar fulguraciones es con filtro en H $\alpha$ . El H $\alpha$  es la línea más sensitiva al calentamiento atmosférico por las partículas energéticas que resultan de las fulguraciones, además, muestra el fondo magnético de la fulguración, especialmente da aspectos como los filamentos.

---

<sup>5</sup> Onda hecha de radiación y viento solar que se desprende del Sol en el periodo llamado Actividad Máxima Solar.

Las fulguraciones energéticas muestran la fase de destello, una explosión de extrema brillantez. Si estas fulguraciones están sobre el borde de una gran mancha, una erupción y onda de choque aparecerán, con una muy intensa emisión de rayos X duros y baja emisión de rayos X suaves (Foukal, 2004).

En una gran fulguración como se muestra en la figura 1.8, el crecimiento continúa y la energía es liberada sobre una gran área y puede ocurrir una violenta eyección de material. El tiempo de evolución de las fulguraciones solares se puede dividir en 4 fases: Precursor, Impulsiva, Explosiva y Gradual.



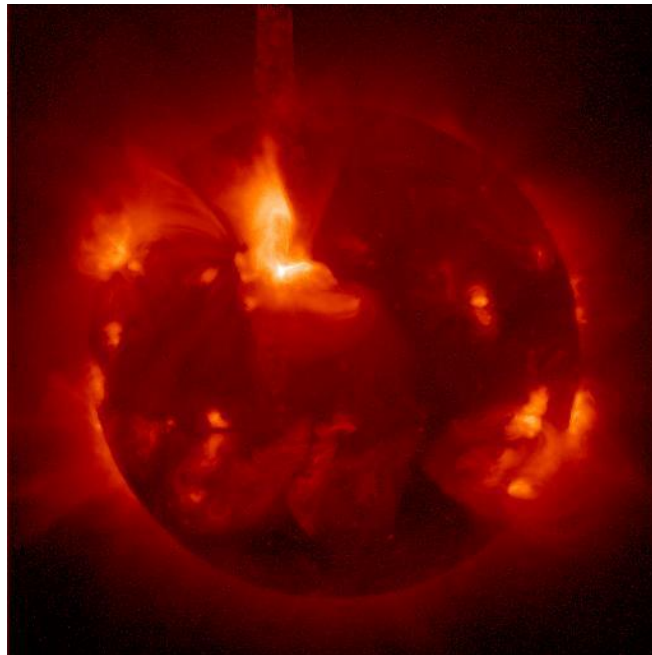
**Figura 1.8.** Fulguración solar del 10 de Noviembre de 2005. Tomada por el satélite SOHO.  
**Fuente:** <http://sohowww.nascom.nasa.gov/>. 2008.

La fase del precursor dura entre 10 y 30 minutos y se observa una acumulación gradual de rayos X suaves y radiación UV extrema. La fase impulsiva dura entre 10 y 1000 segundos y se observa en micro-ondas, rayos X duros y, eventualmente, en rayos  $\gamma$ . En la fase explosiva, el flujo en  $H\alpha$  es máximo sobre un periodo del orden de minutos. Finalmente, la fase gradual puede durar una a varias horas y libera la mayor parte de la energía total de la fulguración.

### 1.3.4 Emisiones de las fulguraciones

#### 1.3.4.1 Rayos X Suaves

Los rayos X suaves contienen un rango de longitud de onda más larga y tienen una fracción importante de energía radiada en las fulguraciones. Las emisiones de rayos X suaves están entre 1-10 Å (ångström<sup>6</sup>) y consisten de una mezcla del Bremsstrahlung<sup>7</sup> y líneas espectrales, debido a las especies altamente ionizadas de metales. Los electrones responsables son electrones térmicos con una temperatura alrededor de los 10<sup>7</sup> K. Como esta temperatura es típica de la corona solar, las líneas de rayos X suaves proveen un diagnóstico importante de las condiciones (temperatura, velocidad, densidad) de la atmósfera, (Schmelzm 1992). La figura 1.9 muestra una fulguración solar en rayos x suaves.



**Figura 1.9.** Fulguración solar en rayos X suaves.

**Fuente:** <http://solar.physics.montana.edu/sxt/>. 2008.

---

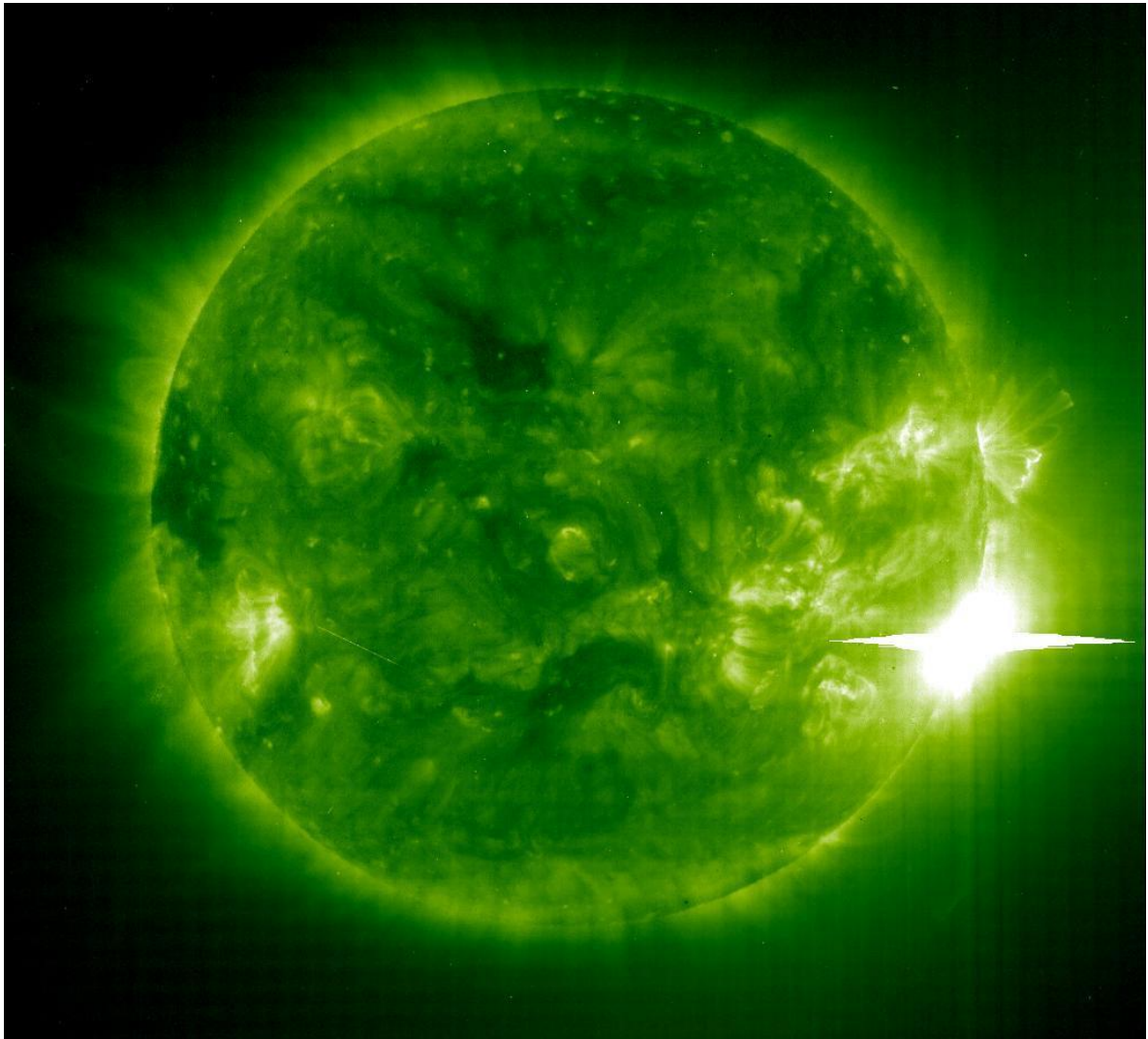
<sup>6</sup> Unidad de longitud empleada principalmente para expresar longitudes de onda, distancias moleculares y atómicas.

<sup>7</sup> Cuando un electrón se mueve rápida y libremente fuera de un átomo, es inevitable que se mueva cerca de un protón en el gas ambiente. Hay una atracción eléctrica entre el electrón y el protón. El electrón libre es desviado de su camino y cambia su velocidad, emitiendo radiación electromagnética en el proceso, conocida como Bremsstrahlung.

#### **1.3.4.2 Rayos X Duros**

La emisión tiene un rango de energías de 10-100 keV. Por debajo de esta energía son considerados rayos X suaves. Por encima de esta energía son considerados como rayos gama, debido a que el mecanismo productor cambia. Esta emisión se produce mediante el Bremsstrahlung de los electrones, (Longair, 1992).

La figura 1.13 muestra la emisión de rayos x duros durante la fulguración del 28 de Octubre de 2003.



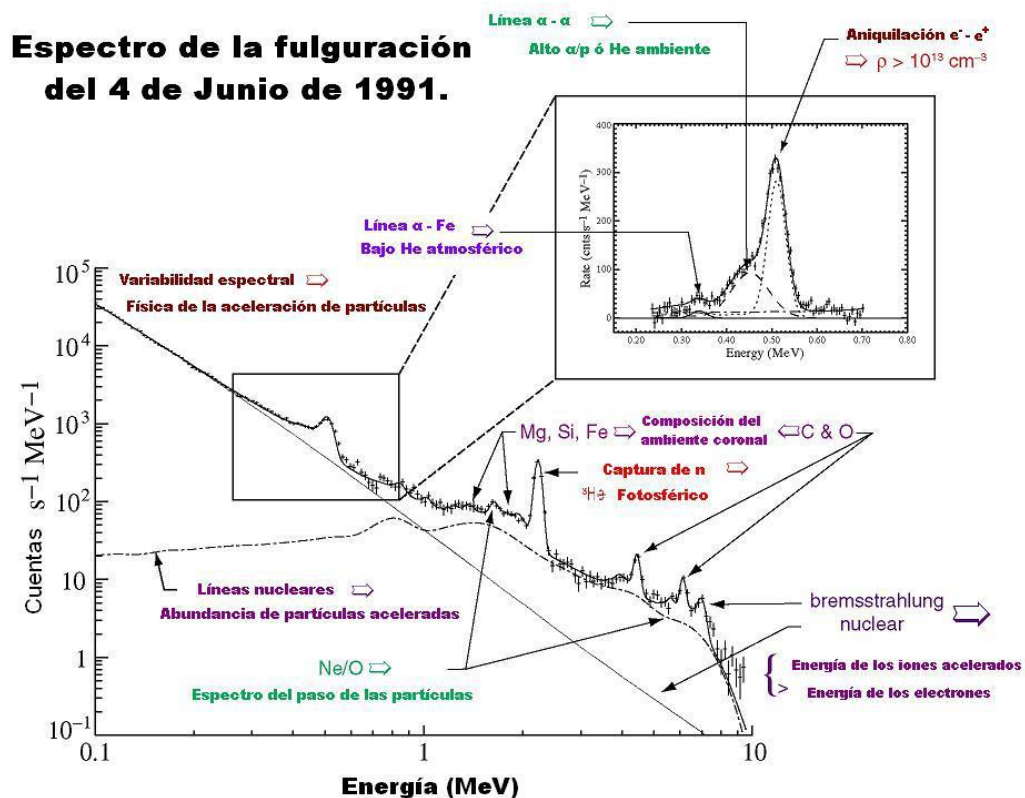
**Figura 1.10.** Fulguración solar del 4 de Noviembre de 2003, la más energética registrada en la historia. Tomada por el satélite SOHO.

**Fuente:** <http://sohowww.nascom.nasa.gov/>. 2008.

### 1.3.4.3 Rayos Gama

La emisión de rayos gama ( $\gamma$ ) en el Sol quieto está totalmente ausente, tal que algún fenómeno en  $\gamma$  es una emanación asociada con una fulguración.

Las líneas de emisión de distintos elementos al igual que la del continuo han sido constantemente observadas; estas son debidas a procesos nucleares (desexcitación de núcleos y reacciones nucleares que producen  $\pi^0$  que decaen en dos gamas), la aniquilación del par  $e^-e^+$  y la línea de formación de Deuterio<sup>8</sup> a 2.223 MeV, producida por la absorción de neutrones lentos por los protones del ambiente atmosférico Solar. El continuo es debido al Bremsstrahlung de electrones relativistas. La figura 1.14 muestra un espectro en rayos gama de una fulguración.



**Figura 1.11.** Espectro en rayos gama de la fulguración del 04 de Junio de 1991. Fuente: (Murphy, 1991). 2008.

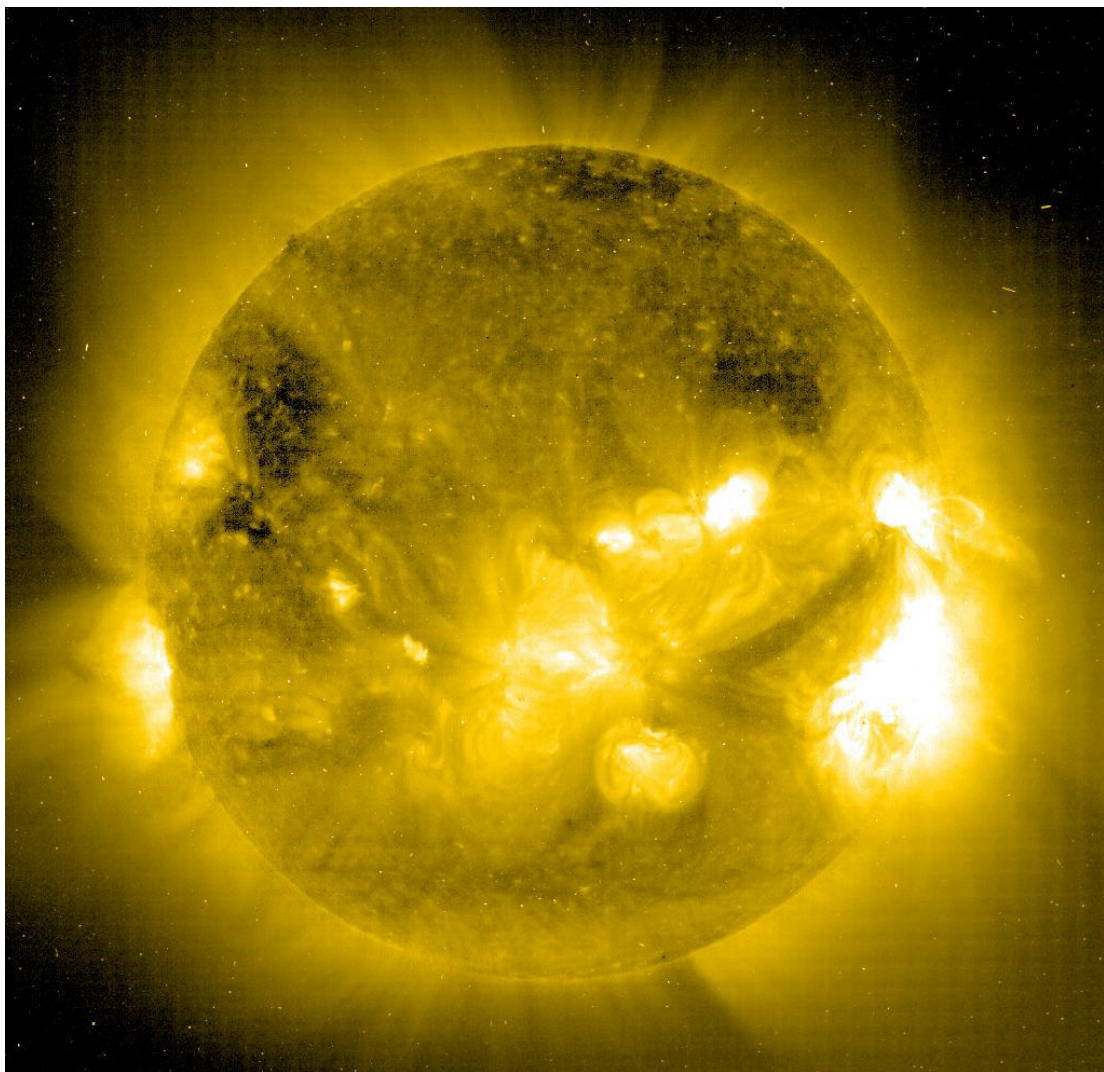
<sup>8</sup> Isótopo estable del hidrógeno que se encuentra en la naturaleza con una abundancia del 0,015 % átomos de hidrógeno.



#### **1.3.4.4 Ultravioleta**

Emisiones térmicas en temperaturas del orden de 104-105 K, que se emiten en las regiones de transición entre la densa corona y la cromósfera.

Esta región del espectro provee un diagnóstico sobre el rápido cambio de la temperatura y densidad de las atmósferas donde se presenta una fulguración, (Schmelzm 1992). La figura 1.12 muestra la emisión en ultravioleta de una fulguración.

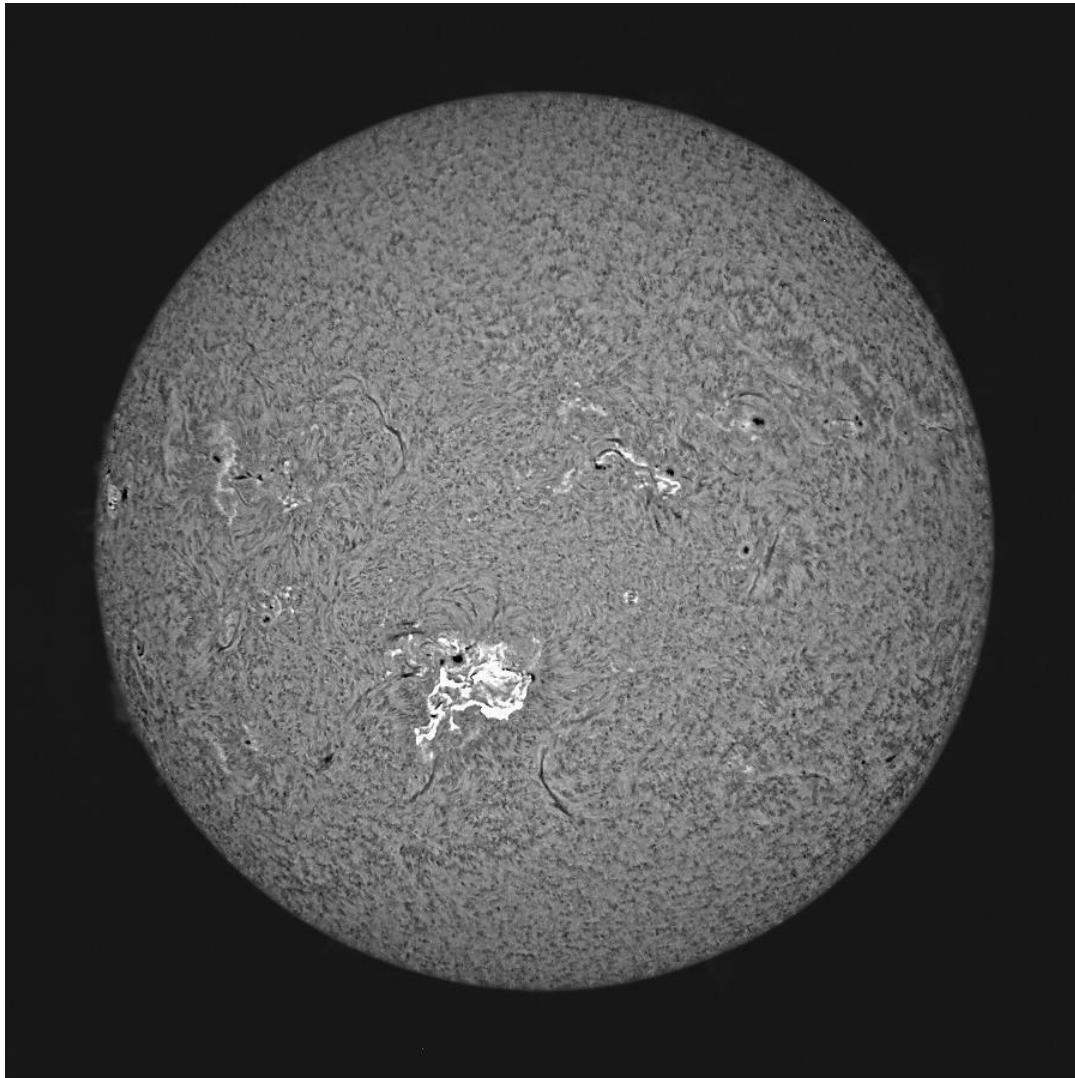


**Figura 1.12.** Fulguración solar del 2 de Noviembre de 2003 en ultravioleta (gran emisión mostrada en la extrema derecha).

**Fuente:** <http://sohowww.nascom.nasa.gov/gallery/images.html>. 2008.

#### 1.3.4.5 Óptico

El óptico muestra una variedad de líneas y el continuo, constituyendo una fracción importante de la energía total liberada. Aquí se encuentra la emisión de líneas que normalmente se observan en el Sol quieto, por ejemplo el H $\alpha$ , que se encuentra en el ancho de la línea de Balmer<sup>9</sup> del Hidrógeno (656.5 nm).



**Figura. 1.13.** Fulguración solar del 19 de Julio de 2000 en H $\alpha$ . El gran abrillantamiento de la parte central fue visto por el satélite SOHO.

**Fuente:** <http://www.nasaimages.org/>. 2008.

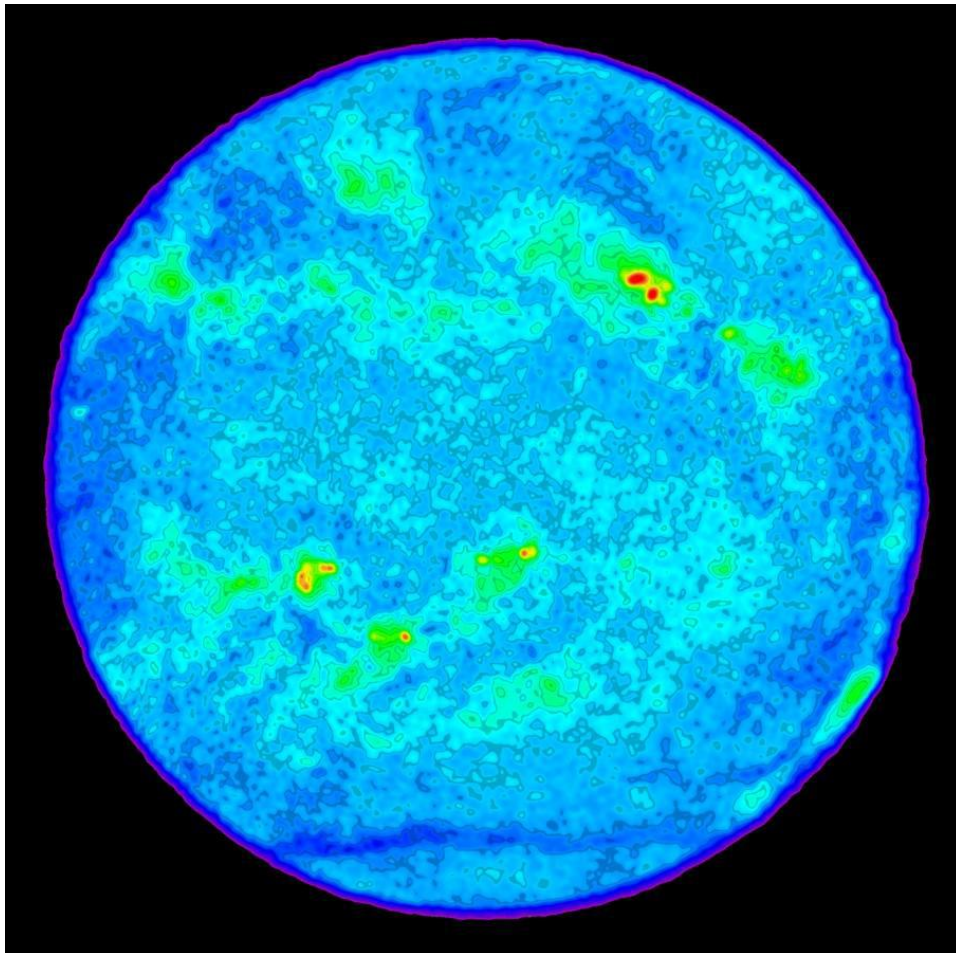
---

<sup>9</sup> Conjunto de rayas que resultan de la emisión del átomo de hidrógeno cuando un electrón transita desde un nivel  $n \geq 3$  a  $n = 2$ .

#### 1.3.4.6 Radio

Las ondas de radio constituyen una radiación de baja energía. Proveen un diagnóstico de algunos de los fenómenos más energéticos en la fulguración.

Las ondas de radio son insignificantes en energía, pero su emisión es rica en detalles y complejidad y permite un diagnóstico de los electrones energéticos, debido a que estos son la fuente emisora de tal radiación, y de la intensidad del campo magnético. Los principales mecanismos de emisión son la radiación giro-sincrotrón<sup>10</sup>, el Bremsstrahlung y los procesos del plasma colectivo, (Schmelzm 1992). La figura 1.14 muestra una fulguración solar vista en radio.



**Figura. 1.14.** Fulguración solar vista en radio. La zona roja es la zona de mayor emisión en radio.

**Fuente:** <http://www.nasaimages.org/>. 2008

---

<sup>10</sup> Electrones moviéndose a velocidades cercanas a la luz emiten un haz de radiación al girar alrededor de una línea de campo magnético.

### 1.3.5 Tipos de fulguraciones y su clasificación

Históricamente las fulguraciones han sido clasificadas por su apariencia en las líneas de emisión en H $\alpha$ . Se usa un esquema de clasificación de dos parámetros; el primer parámetro es el área de brillo en H $\alpha$ , el segundo es la intensidad del incremento de la emisión (débil, normal o brillante). Así, por ejemplo, una pequeña y muy brillante región H $\alpha$  puede ser clasificada como una fulguración 1b.

La nueva instrumentación generó que este esquema de clasificación no fuera el mejor indicador de la cantidad de energía liberada. Como una gran fracción de radiación en una fulguración aparece como rayos X radiados por el plasma coronal caliente, producido por las fulguraciones, una nueva clasificación fue generada, basada en el flujo de energía en la banda de 1-8 Å, como es observado por el satélite GOES. Este esquema usa una letra para denotar el orden de magnitud del flujo y un número para denotar el múltiplo de la cantidad base. La clasificación es la siguiente:

$$A \ 10^{-8} \text{ W m}^{-2}$$

$$B \ 10^{-7} \text{ W m}^{-2}$$

$$C \ 10^{-6} \text{ W m}^{-2}$$

$$M \ 10^{-5} \text{ W m}^{-2}$$

$$X \ 10^{-4} \text{ W m}^{-2}$$

De este modo, una fulguración tipo X5 tiene un flujo de  $5 \times 10^{-4} \text{ W m}^{-2}$  en el canal de 1-8 Å del satélite GOES.

Existen criterios para clasificar eventos solares en impulsivos, graduales y térmicos. Las fulguraciones en rayos X duros han sido diferenciadas en estas tres clases. El criterio más importante es el perfil temporal de la emisión de rayos X. Las fulguraciones son clasificadas de la siguiente forma:

**Térmicas:** que son presumiblemente eventos compactos, que exhiben una suave variación en el perfil de flujo vs tiempo, correspondiendo a la energización de un lazo. Contiene muy baja densidad de partículas.

**Impulsivas:** generalmente asociadas con lazos de fulguraciones ( $\geq 20,000$  Km.), exhibiendo un agudo perfil de flujo vs tiempo. Muestran una población de partículas energéticas enriquecidas con núcleos pesados y  $^3\text{He}$ , además de los electrones, (Reames, 1999). Las partículas escapan después de la aceleración. Las partículas producidas en estas fulguraciones pueden ser visibles sólo en fulguraciones que no están acompañadas de una eyección coronal de masa (ECM), debido a que el choque asociado produce un flujo de partículas aceleradas que opaca a las partículas provenientes de la fulguración misma (Cliver, 1996).

**Graduales:** se considera que ocurren en lazos muy largos ( $\geq 50,000$  Km.). Conlleva partículas atrapadas en una baja densidad coronal, decayendo lentamente a través de colisiones Coulombianas. El gran tamaño de los eventos graduales está asociado con eventos de mayor duración que los anteriores. Tienen composiciones similares a las abundancias solares promedio; los electrones son minoría en el flujo de partículas. Los eventos menos intensos son mucho más frecuentes que las fulguraciones muy energéticas o brillantes.

### **1.3.6 Liberación y transporte de energía en fulguraciones**

Las fulguraciones ocurren en regiones de campo magnético intenso y bajo grandes presiones, donde puede ocurrir una interacción entre líneas de campo magnético; por ejemplo, en la vecindad de las manchas Solares. Se sabe que existe suficiente energía ( $1/8\pi \int B^2 dV$ ) en el campo magnético para explicar la potencia observada en las fulguraciones Solares. Así, una de las interrogantes es conocer cómo esta energía magnética es disipada lo suficientemente rápido para tomar en cuenta los tiempos de escala observados en las fulguraciones. La razón de este problema radica en la extremadamente alta inductancia y baja resistencia del plasma coronal.

Una vez que la energía ha sido liberada, esta es transportada a través de la atmósfera para producir un incremento de la temperatura en la corona y la cromósfera, al igual que un incremento en la radiación observado en el espectro

electromagnético, (Zirin, 1972). Varios mecanismos han sido propuestos para efectuar este transporte de energía, siendo los más notables entre estos mecanismos: (1) la propagación de electrones supra térmicos acelerados y (2) la difusión de calor desde la región primaria de energía liberada.

#### **1.3.6.1 Disipación de energía en fulguraciones**

Algunas definiciones referentes a los modos de liberación de energía y producción de rayos X duros en las fulguraciones son:

**Térmica:** una fuente de rayos X es llamada térmica si la energía del Bremsstrahlung producido por los electrones es comparable a la energía del ensamble entero de electrones de la fuente. Esto implica que no hay pérdidas seculares de energía de los electrones produciendo Bremsstrahlung y, en principio, que cada una de las fuentes puede ser 100 % eficiente en la producción de rayos X.

**No Térmica:** se considera una fuente no térmica de rayos X, si su promedio de energía excede la energía de los electrones de fondo en la fuente. Los electrones que producen Bremsstrahlung ahora pierden una considerable fracción de su energía en colisiones con electrones del medio ambiente atmosférico y electrones lentos y sólo una pequeña fracción (0.001%) de su energía se pierde en pequeños impactos y colisiones emitiendo Bremsstrahlung con protones del medio ambiente. Por lo tanto, la eficiencia de la fuente es muy pequeña, requiriendo mucho más energía del electrón, (Schmelzm 1992).

La liberación de la energía magnética en la fulguración solar se produce, principalmente, durante la fase impulsiva, cuando las partículas cargadas son aceleradas y se emiten rayos X duros no térmicos. La fase de decaimiento térmico, detectado por la emisión gradual de rayos X blandos, se produce como consecuencia de la interacción de la atmósfera solar y las partículas energéticas generadas durante la fase impulsiva. Una fulguración solar se produce por una inestabilidad o reordenamiento en la configuración magnética en la corona baja.

En el siguiente capítulo se describe el diseño y funcionamiento del Monitor de Neutrones de la Ciudad de México ubicado en Ciudad Universitaria, el cual tiene como objetivo principal monitorear la intensidad de la radiación cósmica que llega continuamente a la Tierra desde el espacio.

# CAPÍTULO 2: MONITOR DE NEUTRONES DE LA CIUDAD DE MÉXICO





## 2.1 INTRODUCCIÓN

*"Podría parecer que hemos llegado a los límites alcanzables por la tecnología informática, aunque uno debe ser prudente con estas afirmaciones, pues tienden a sonar bastante tontas en cinco años"*  
**John Von Neumann, sobre 1949**

De acuerdo con, (González, 2008), los monitores de neutrones se usan para detectar la componente nucleónica producida por la radiación cósmica primaria. La radiación cósmica primaria menos energética produce menos neutrones y protones al interaccionar con los núcleos atmosféricos. Tienen alta sensibilidad, lo cual permite que tenga cuentas muy altas. Son omnidireccionales, sin embargo, no pueden diferenciar la dirección de arribo de las partículas que llegan a la Tierra. No resuelven la energía de las partículas y diferencian entre neutrones y protones.

Existen dos tipos de monitores de neutrones estandarizados, uno es el IGY (International Geophysical Year) y el otro el NM64. El monitor de neutrones IGY fue originalmente diseñado por J.A. Simpson en 1953. Para 1964 y con el propósito de mejorar el diseño de IGY, Hugh Carmichael diseñó el Súper Monitor de Neutrones NM64. El NM64 es 3 veces más eficiente que el IGY; por tal motivo es el detector normalmente usado alrededor del mundo.

En este capítulo describo la estructura y diseño del Monitor de Neutrones NM64 instalado en el Observatorio de Rayos Cósmicos de la Ciudad de México ubicado en Ciudad Universitaria, así como la forma en la que obtiene las mediciones para entender mejor las gráficas de PySMG.

## 2.2 MONITOR DE NEUTRONES (6-NM64)

Desde 1990, en Ciudad Universitaria se instaló un Monitor de Neutrones 6-NM64 dentro del Observatorio de Rayos Cósmicos a cargo del Instituto de Geofísica de la UNAM. El Monitor de Neutrones se encuentra trabajando continuamente desde esta fecha. En la figura 2.1 muestra el monitor de neutrones de Ciudad Universitaria.



**Figura 2.1.** Monitor de Neutrones 6-NM64 instalado en el Observatorio de Rayos Cósmicos en Ciudad Universitaria.

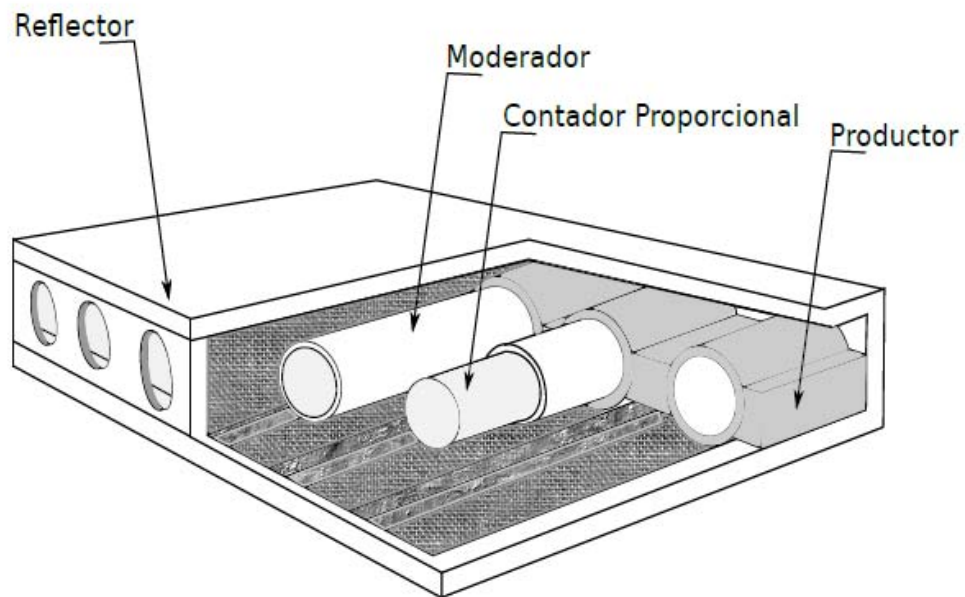
**Fuente:** [www.geofisica.unam.mx](http://www.geofisica.unam.mx). 2015.

De acuerdo con (González, 2008), el concepto básico de detección de neutrones con un monitor de neutrones NM64 consiste principalmente de cuatro (4) partes; el reflector, el productor, el moderador y el contador proporcional.

## 2.3 DISEÑO

De acuerdo con (García, 2014), los monitores de neutrones se denominan conforme al número de contadores proporcionales de trifloruro de boro ( $\text{BF}_3$ ) con los que cuente. Por lo tanto, a un monitor de neutrones con tres (3) contadores proporcionales se le denomina 3-NM64.

El monitor de neutrones de Ciudad Universitaria cuenta con seis (6) contadores proporcionales, es por esto que se le denomina 6-NM64.



**Figura 2.2.** Esquema que muestra la estructura del Monitor de Neutrones de Ciudad Universitaria 6-NM64.

**Fuente:** (García, 2014). 2014.

Como ya se comentó, la estructura del 6-NM64 consta primeramente del contador proporcional que se encuentra cubierto por el moderador, que a su vez se encuentra rodeado por el productor, finalmente todo esto cubierto por el reflector. En la figura 2.2 se describe esta estructura.

### 2.3.1 Contador proporcional de BF<sub>3</sub>

El contador proporcional de BF<sub>3</sub><sup>11</sup> es uno de los más utilizados alrededor del mundo para la detección de neutrones. De acuerdo con estándares internacionales para el NM64, estos son sus especificaciones técnicas:

Construido de acero inoxidable con grosor de 0.075 cm a 0.084 cm.

Diámetro interno de 14.85 cm.

Longitud activa de 191 cm.

Volumen efectivo de 33 lt.

Relleno de Trifloruro de Boro (96% B10) a una presión de 20 cmHg a 22 °C.

El contador proporcional de BF<sub>3</sub> enriquecido con B10 por sí solo no es capaz de detectar neutrones rápidos, por lo que se debe utilizar un *moderador* para que se detecten estos neutrones rápidos.

### 2.3.2 Moderador

El moderador está compuesto por un tubo de polietileno con espesor de 2 cm y diámetro exterior de 24.5 cm.

Como ya se comentó, el contador proporcional no es capaz por sí solo de detectar los neutrones rápidos. El moderador de polietileno sirve precisamente para que estos neutrones colisionen con otros núcleos y de esta manera desacelerarlos intercambiando su energía cinética, ya que si llegaran directamente al contador proporcional tendrían muy poca probabilidad de ser detectados.

En otras palabras, la función del moderador es reducir las energías de los neutrones incidentes, para dejarlos en el rango de las energías térmicas ~0.025eV como sea posible, (García, 2014).

---

<sup>11</sup> Trifloruro de Boro (BF<sub>3</sub>).

### **2.3.3 Productor**

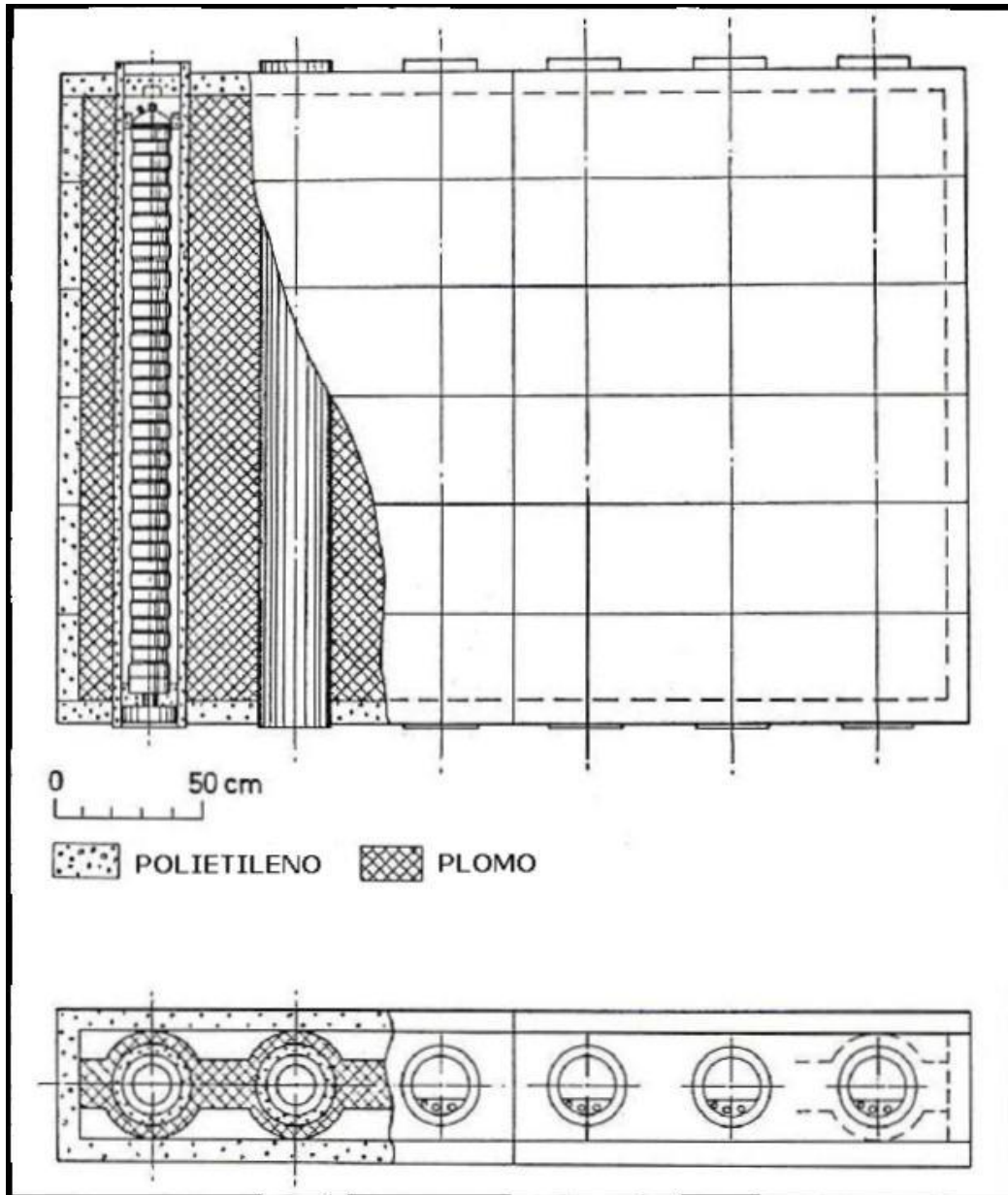
De acuerdo con (García, 2014), el productor está compuesto por anillos de plomo, con una pureza no menor a 99 %. Los anillos tienen 25.4 cm de diámetro interno y 35.6 cm de diámetro exterior.

Los protones y neutrones que inciden en el monitor, procedentes de rayos cósmicos secundarios, producen reacciones nucleares con el plomo. Las cuales dan como resultado neutrones de evaporación de baja energía. Los neutrones de evaporación están en el rango de energía entre 2 MeV y 15 MeV aproximadamente. Gracias a esta reacción nuclear, la probabilidad de detección incrementa ya que por cada partícula incidente en el monitor se generan aproximadamente 15 neutrones de evaporación. Finalmente, la probabilidad de detección de estos neutrones de evaporación por parte del tubo contador es del ~6 %.

### **2.3.4 Reflector**

El reflector tiene la forma de una caja rectangular construida de bloques de polietileno de baja densidad con un grosor de 7.5 cm y 37 cm de ancho (La figura 2.3 muestra el esquema del reflector). Los rayos cósmicos secundarios que llegan al monitor, atraviesan el reflector sin pérdidas de energía significativas.

La tarea de este sistema es reflejar los neutrones de evaporación producidos en el plomo hacia los contadores proporcionales. Además aísla y absorbe neutrones de baja energía producidos en el entorno externo del monitor, (García, 2014).



**Figura 2.3.** Muestra la composición del reflector del 6-NM64.

**Fuente:** (González, 2008). 2008.

## 2.4 ADQUISICIÓN DE DATOS

De acuerdo con (García, 2014), en el Observatorio de Rayos Cósmicos de Ciudad Universitaria, el Sistema Adquisidor de Datos se encuentra en una etapa de actualización. El nuevo Sistema de Adquisición es un diseño de aplicación específica desarrollado en un dispositivo lógico programable. El software que controla la adquisición y despliega la información también está programado en Python. Además el sistema cuenta con una interfaz de comunicación basada en el protocolo de comunicación I<sup>2</sup>C (Inter-Integrated Circuit).

El Observatorio de rayos cósmicos de Ciudad Universitaria detecta las partículas que llegan desde el espacio y que diariamente llegan a la tierra desde todas las direcciones. El sistema almacena la información y tiene la capacidad de mostrarlos en tiempo real.

Estos datos son vertidos en un archivo (\*.txt) con razón de conteo mínima de 1 minuto (~1350) en formato ASCII.

```
Mexico City, Neutron Monitor  
Data resolution: 1 minute  
From 27-1-2015 to 27-1-2015
```

<b>Date</b>	<b>Time</b>	<b>Data</b>
2015-01-27	00:00:00	1327
2015-01-27	00:01:00	1328
2015-01-27	00:02:00	1345
2015-01-27	00:03:00	1316
2015-01-27	00:04:00	1310
2015-01-27	00:05:00	1353
2015-01-27	00:06:00	1318
2015-01-27	00:07:00	1355
2015-01-27	00:08:00	1323

Este es un ejemplo de los datos que arroja el Observatorio del día 27 de enero de 2015 con resolución de 1 minuto y corregidos por presión. Estos son los datos con los que trabaja PySMG, el cual está desarrollado en Python y que en el siguiente capítulo se describe el lenguaje a profundidad.

**CAPÍTULO 3:**  
**LENGUAJE DE**  
**PROGRAMACIÓN**  
**PYTHON**





## 3.1 HISTORIA

*"Hay sólo dos clases de lenguajes de programación:  
Aquellos de los que la gente está siempre quejándose  
y aquellos que nadie usa".  
Bjarne Stroustrup*

El lenguaje Python tiene sus principios a finales de los ochenta, creado por el investigador holandés Guido Van Rossum del centro de investigación CWI (Centro para las Matemáticas e Informática) de Ámsterdam.

Guido Van Rossum fue asignado a un proyecto que consistía en el desarrollo de un sistema operativo distribuido llamado Amoeba. En el CWI se utilizaba un lenguaje de programación llamado ABC, por lo que Guido decide crear un nuevo lenguaje para desarrollar el proyecto Amoeba con el fin de superar las limitaciones y problemas que se había encontrado en proyectos anteriores trabajando con el lenguaje ABC.

En 1991 se publica la primera versión de Python (0.9.0), sin embargo, es hasta 1994 cuando se publica la versión 1.0 en conjunto con el primer foro de discusión del lenguaje. En el año 2001 se creó la Python Software Foundation License que es compatible con la GPL (GNU General Public License). Esta nueva licencia se diferencia de la GPL al ser no copyleft, por lo que es posible modificar el código fuente y desarrollar nuevo código sin la necesidad de hacerlo open source.

En la actualidad existen sólo tres versiones principales, sin embargo, para las versiones 2.x y 3.x se da mantenimiento por separado, esto quiere decir que la actualización 2.7 y la 3.3 se consideran estables, esto debido a que las características entre ambas versiones las hace incompatibles.

Con la creación de la versión 3.x Guido Van Rossum pretende mejorar la versión 2.x en todos los aspectos, esto con el objetivo de lograr un mayor rendimiento y lograr una mayor coherencia con la sintaxis del lenguaje, por estas razones la versión 3.x se vuelve incompatible con la versión 2.x.

## 3.2 CARACTERÍSTICAS

Python es un lenguaje de programación interpretado de alto nivel, entre sus principales características están:

**Lenguaje de alto nivel:** Este tipo de lenguajes se caracterizan principalmente por contar con una estructura sintáctica y semántica legible, acorde a las capacidades cognitivas humanas, (Bahit, 2012).

**Es un lenguaje interpretado:** A diferencia de los lenguajes compilados Python está diseñado para ser ejecutado por medio de su intérprete. Un intérprete funciona de manera similar a un compilador, con la diferencia que ejecuta el código de manera directa y no necesita ejecutar primeramente un ejecutable para el programa.

**Multiplataforma:** Python está diseñado para poder ser ejecutado en diferentes plataformas informáticas como por ejemplo: Microsoft Windows, GNU Linux, Mac OS X, etc.

**Filosofía:** la filosofía de Python hace especial énfasis en una sintaxis clara y limpia que favorezca un código legible.

**Multiparadigma:** Se dice que Python es multiparadigma ya que soporta diferentes estilos de programación como: la orientación a objetos, programación imperativa y, en menor medida, programación funcional.

**Tipado dinámico:** Esto quiere decir que Python nos da la facilidad de no definir el tipo al declarar una variable, por lo que esta puede tomar valores de distinto tipo en diferentes momentos.

### 3.3 FILOSOFÍA

Python es un lenguaje cuya filosofía de trabajo busca desarrollos legibles y transparentes. Estos principios quedan plasmados en un documento llamado **El Zen de Python** que está descrito en el **Python Enhancement Proposal N° 20 (PEP)** escrito por Tim Peters:

El Zen de Python:

Hermoso es mejor que feo.  
Explícito es mejor que implícito.  
Simple es mejor que complejo.  
Complejo es mejor que complicado.  
Sencillo es mejor que anidado.  
Escaso es mejor que denso.  
La legibilidad cuenta.  
Los casos especiales no son lo suficientemente especiales para romper las reglas.  
Lo práctico le gana a la pureza.  
Los errores nunca deberían dejarse pasar silenciosamente.  
A menos que hayan sido silenciados explícitamente.  
Frente a la ambigüedad, rechaza la tentación de adivinar.  
Debería haber una –y preferiblemente sólo una- manera obvia de hacerlo.  
Aunque esa manera puede no ser obvia al principio a menos que usted sea holandés.  
Ahora es mejor que nunca.  
Aunque “nunca” es a menudo mejor que “ahora mismo”.  
Si la implementación es difícil de explicar, es una mala idea.  
Si la implementación es fácil de explicar, puede que sea una buena idea.  
Los “namespaces” son una gran idea ¡Hay que hacer más de eso! (Peters, 2004)

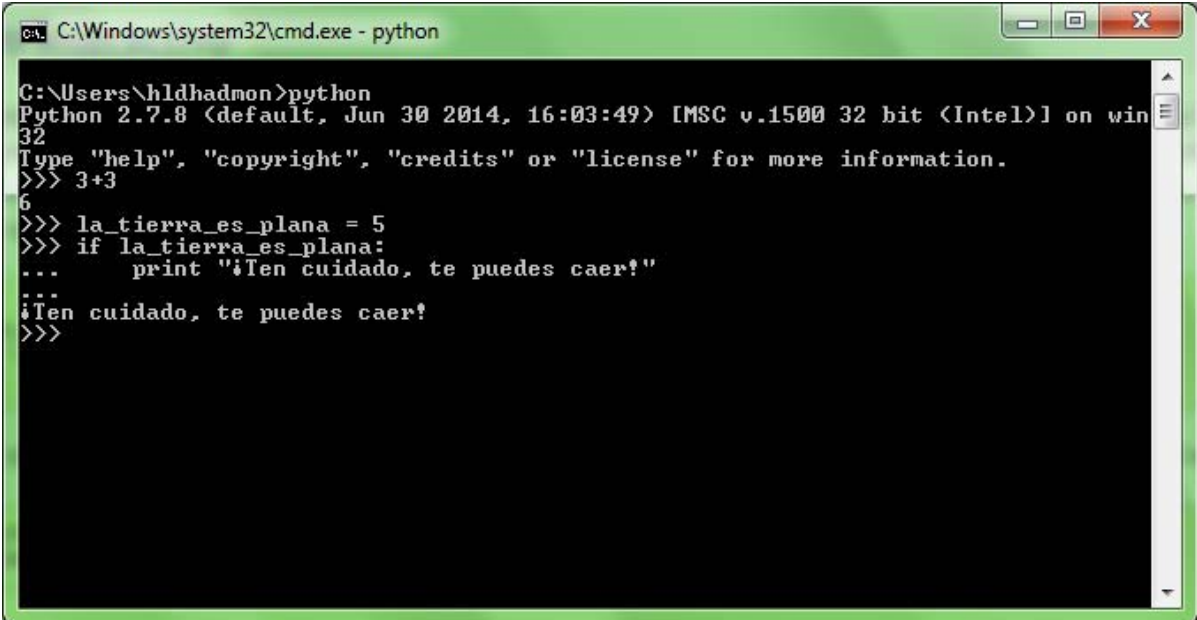
El código que se rige por los principios de legibilidad y transparencia de Python se dice que es un “código Pythonico”.

### 3.4 MODO INTERACTIVO

El intérprete de Python cuenta con una Shell interactiva. Se dice que se está en modo interactivo cuando las instrucciones se leen desde la terminal.

Las expresiones son introducidas una a una por medio del “prompt primario” (>>>) mientras que las líneas de continuación por medio del “prompt secundario” (...), arrojando inmediatamente el resultado de su evaluación, por lo que resulta ser muy útil tanto para las personas que se familiarizan con el lenguaje como para los programadores expertos al permitir probar porciones de código antes de integrarlo al programa final.

Existen programas que añaden funciones a esta Shell interactiva, como texto enriquecido, señalamiento de errores y el autocompletado de código, tales como: IDLE, bpython o iPython por mencionar algunos.



```
C:\Windows\system32\cmd.exe - python
C:\Users\hldhadmon>python
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit <Intel>] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> 3+3
6
>>> la_tierra_es_plana = 5
>>> if la_tierra_es_plana:
...     print "¡Ten cuidado, te puedes caer!"
...
¡Ten cuidado, te puedes caer!
>>>
```

**Figura 3.1.** Modo interactivo de Python.

**Fuente:** Elaboración propia (2015).

## 3.5 ELEMENTOS DEL LENGUAJE

Python como la mayoría de los lenguajes de programación cuenta con varios elementos y reglas de estilos que definen su estructura. Estas reglas de estilos se describen en el **PEP 8**, (Van Rossum, 2001).

### 3.5.1 Variables

En general una variable se define como un espacio de memoria reservado para almacenar datos. Estos datos son modificables en cualquier momento. Por lo que cada variable cuenta con un nombre, longitud arbitraria y un valor.

Python cuenta con un tipo de “variable” llamada **constante** y se utiliza para definir valores fijos, los cuales no es necesario modificar.

En Python los nombres de las variables pueden estar formados por números y letras, sin embargo, deben empezar con una letra y nunca con un número. Con base en el **PEP 8**, los nombres de las variables deben ser descriptivos y/o hacer referencia al tipo de valor que almacenan, utilizar minúsculas y para los nombres compuestos se puede separar por medio del guion bajo (`_`) o separar las palabras con letra capital.

Para las constantes se deben utilizar letras mayúsculas y separar las palabras con guion bajo (`_`) al igual que los nombres descriptivos:

#### **Correcto:**

```
mi_variable = 10
MiVariable = 10
Variable3 = 10
MI_CONSTANTE = 10
```

#### **Incorrecto:**

```
45variable = 10
Constante = 10
```

```
C:\Windows\system32\cmd.exe - python
C:\Users\osvaldo>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> # Definición correcta de variables:
...
>>> mi_variable = 10
>>> MiVariable = 10
>>> Variable = 10
>>> MI_CONSTANTE = 10
>>>
>>> #Definición incorrecta de variables:
...
>>> 45variable = 10
File "<stdin>", line 1
  45variable = 10
    ^
SyntaxError: invalid syntax
>>> $variable = 10
File "<stdin>", line 1
  $variable = 10
    ^
SyntaxError: invalid syntax
>>>
```

**Figura 3.2.** Definición de variables.  
**Fuente:** Elaboración propia (2015).

Una restricción más para formar nombres de variables y constantes son las **palabras reservadas** las cuales son:

<b>and</b>	<b>continue</b>	<b>except</b>	<b>global</b>	<b>lambda</b>	<b>raise</b>	<b>yield</b>
<b>as</b>	<b>def</b>	<b>exec</b>	<b>if</b>	<b>not</b>	<b>return</b>	
<b>assert</b>	<b>del</b>	<b>finally</b>	<b>import</b>	<b>or</b>	<b>try</b>	
<b>break</b>	<b>elif</b>	<b>for</b>	<b>in</b>	<b>pass</b>	<b>while</b>	
<b>class</b>	<b>else</b>	<b>from</b>	<b>is</b>	<b>print</b>	<b>with</b>	

### 3.5.2 Tipos de datos

Una de las ventajas de utilizar Python es su tipado dinámico, ya que no es necesario definir el tipo de dato que almacenará la variable al momento de su declaración; es decir, Python identifica el tipo de dato que se le está asignando a dicha variable sin necesidad de declararlo explícitamente.

Algunos ejemplos de los tipos de datos primitivos en Python:

**Número entero:**

```
mi_edad = 20
```

**Número octal:**

```
mi_edad = 024
```

**Número hexadecimal:**

```
mi_edad = 0x14
```

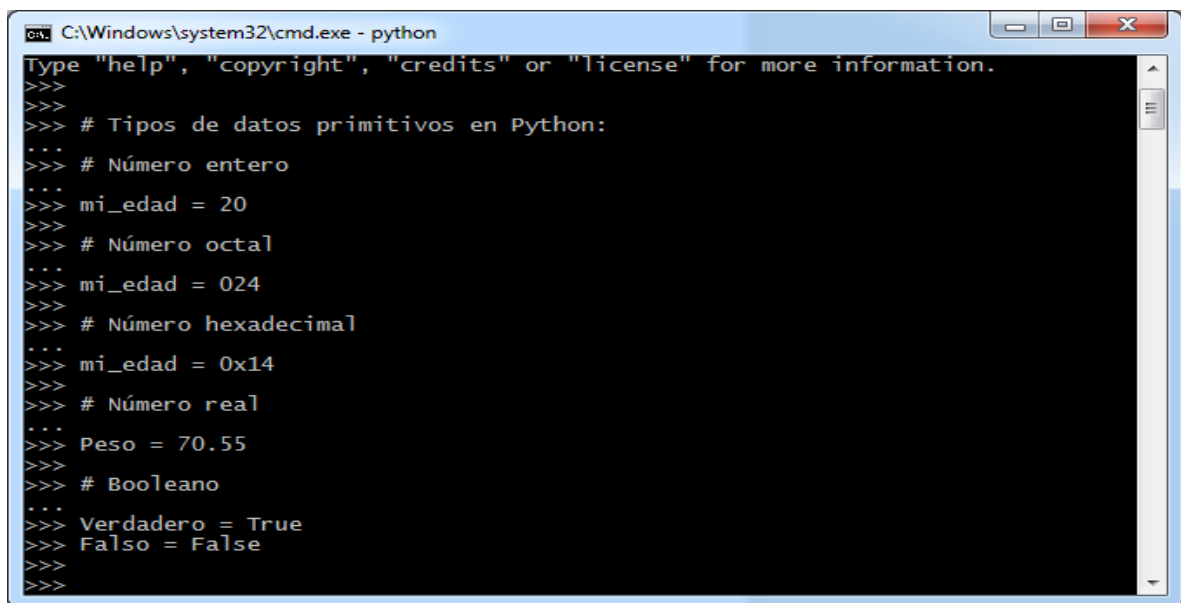
**Número real:**

```
Peso = 70.55
```

**Booleano:**

```
Verdadero = True
```

```
Falso = False
```

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe - python". The window contains Python code defining primitive data types. The code starts with a prompt for help, followed by a comment "# Tipos de datos primitivos en Python:". It then defines an integer variable "mi\_edad" with the value 20, an octal variable "mi\_edad" with the value 024, a hexadecimal variable "mi\_edad" with the value 0x14, a real variable "Peso" with the value 70.55, and a boolean variable "Verdadero" with the value True and "Falso" with the value False. The code is displayed in a monospaced font on a black background with a light blue border around the window.

```
C:\Windows\system32\cmd.exe - python
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> # Tipos de datos primitivos en Python:
>>> # Número entero
>>> mi_edad = 20
>>>
>>> # Número octal
>>> mi_edad = 024
>>>
>>> # Número hexadecimal
>>> mi_edad = 0x14
>>>
>>> # Número real
>>> Peso = 70.55
>>>
>>> # Booleano
>>> Verdadero = True
>>> Falso = False
>>>
```

**Figura 3.3.** Tipos de datos.  
**Fuente:** Elaboración propia (2015).

Además de estos tipos de datos “estándar”, también existen otros tipos de datos complejos que describiré a continuación.

### 3.5.3 Tipos de datos complejos

Los Tipos de datos complejos de Python básicamente son 3 y almacenan **colecciones de datos** de diversos tipos: Listas, Tuplas y Diccionarios. Los cuales describiré a continuación.

#### 3.5.3.1 Listas

Una lista es una variable que almacena datos de diferente tipo cuyos valores pueden ser modificados:

```
mi_lista = ['cadena de texto', 20, 850, 3.5, 'más texto', True]
```

Características:

- Las listas se delimitan por corchetes [ ].
- Permiten agregar nuevos elementos.
- No son inmutables, por lo que permite modificar sus valores.
- Al primer elemento se accede con el índice 0 y al último con -1.

#### 3.5.3.2 Tuplas

A diferencia de las listas las tuplas son colecciones de datos **inmutables**, esto quiere decir que no se puede modificar sus valores una vez declarada.

```
mi_tupla = ('cadena de texto', 20, 850, 3.5, 'más texto', True)
```

Características:

- Las tuplas se delimitan por paréntesis ( ).
- Son **inmutables** por lo que no permite modificar sus elementos.
- Al primer elemento se accede con el índice 0 y al último con -1.

#### 3.5.3.3 Diccionarios

A diferencia de las listas y las tuplas en las que se accede a sus valores mediante índices, a los datos almacenados en los diccionarios se accede mediante una palabra clave.



Los diccionarios los podemos definir como: Un conjunto NO ordenado de pares *clave:valor*.

```
mi_diccionario = {'clave 1': valor_1, 'clave 2': valor_2,...,  
'clave n': valor_n}
```

Características:

- Los diccionarios se delimitan por medio de llaves {}.

- Al igual que las listas, permiten modificar sus valores.

- Para acceder a sus valores se hace a través de su clave.

### 3.5.4 Funciones

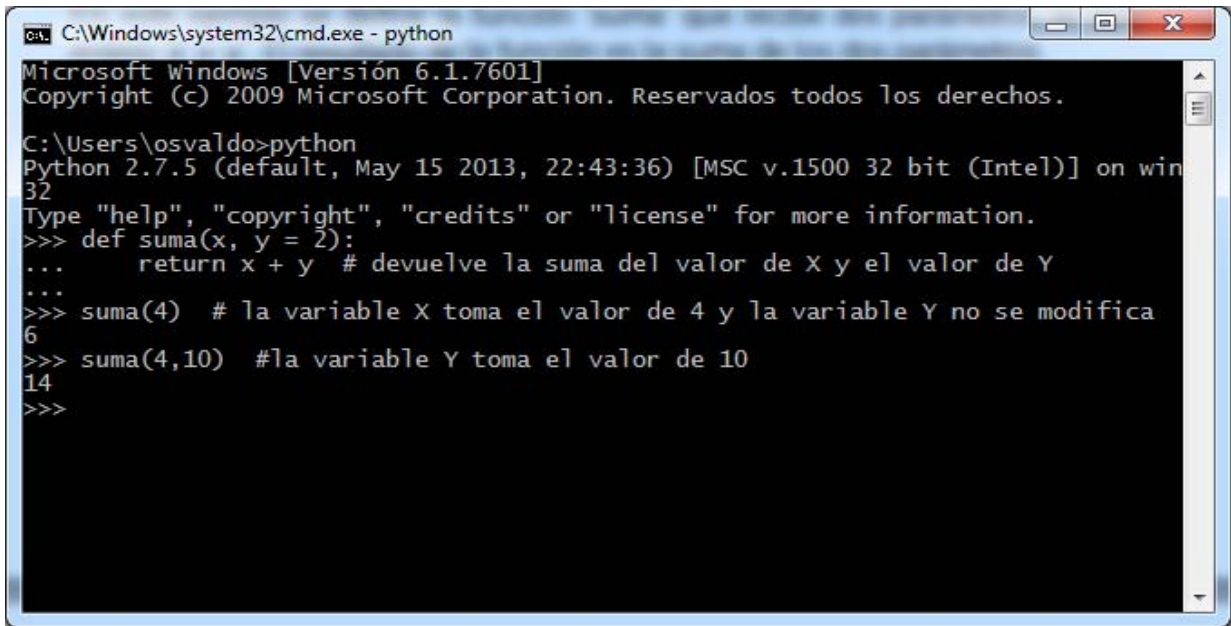
Las funciones son fragmentos de código con un nombre asociado que realizan tareas determinadas para que sean ejecutadas por el usuario cuando éste decida llamarlas y éstas a su vez regresan un valor.

Características:

- Las funciones se definen mediante la palabra reservada **def**, un nombre de la función descriptiva y sus parámetros.

- Las funciones devuelven el resultado de sus instrucciones mediante la palabra reservada **return**.

En la figura 3.4 se muestra la definición de una función en Python mediante la palabra reservada **def**.



```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\osvaldo>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> def suma(x, y = 2):
...     return x + y # devuelve la suma del valor de X y el valor de Y
...
>>> suma(4) # la variable X toma el valor de 4 y la variable Y no se modifica
6
>>> suma(4,10) #la variable Y toma el valor de 10
14
>>>
```

**Figura 3.4.** Función suma.

**Fuente:** Elaboración propia (2015).

En este ejemplo se define la función 'suma' que recibe dos parámetros de tipo entero 'x' y 'y'. El resultado de la función es la suma de los dos parámetros.

Existe otra forma para definir funciones en Python mediante la palabra reservada **lambda**. Lambda nos sirve para definir una función sobre la marcha y en pocas líneas.

Características:

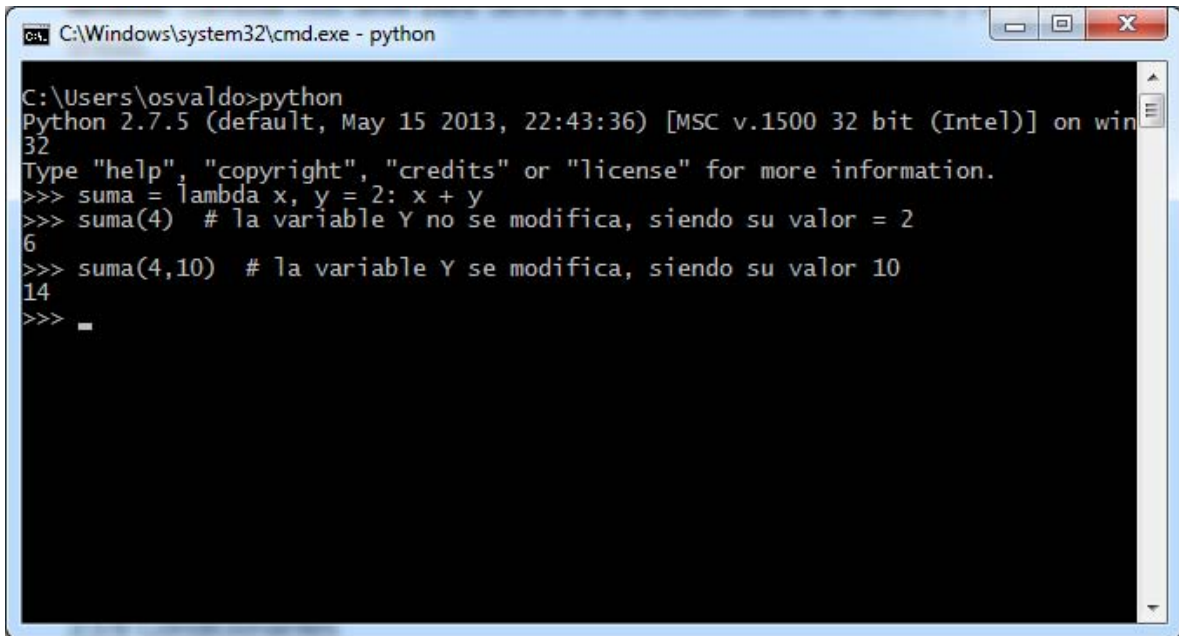
Esta forma de definir funciones es tomada de Lisp<sup>12</sup>.

Las funciones lambda arrojan el mismo resultado que la forma anterior.

En la figura 3.5 se muestra la definición de una función en Python mediante la palabra reservada **lambda**.

---

<sup>12</sup> Lisp es un lenguaje de programación multiparadigma utilizado principalmente en inteligencia artificial.



```
C:\Windows\system32\cmd.exe - python
C:\Users\osvaldo>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> suma = lambda x, y = 2: x + y
>>> suma(4) # la variable Y no se modifica, siendo su valor = 2
6
>>> suma(4,10) # la variable Y se modifica, siendo su valor 10
14
>>> _
```

**Figura 3.5.** Definición de función suma mediante lambda.

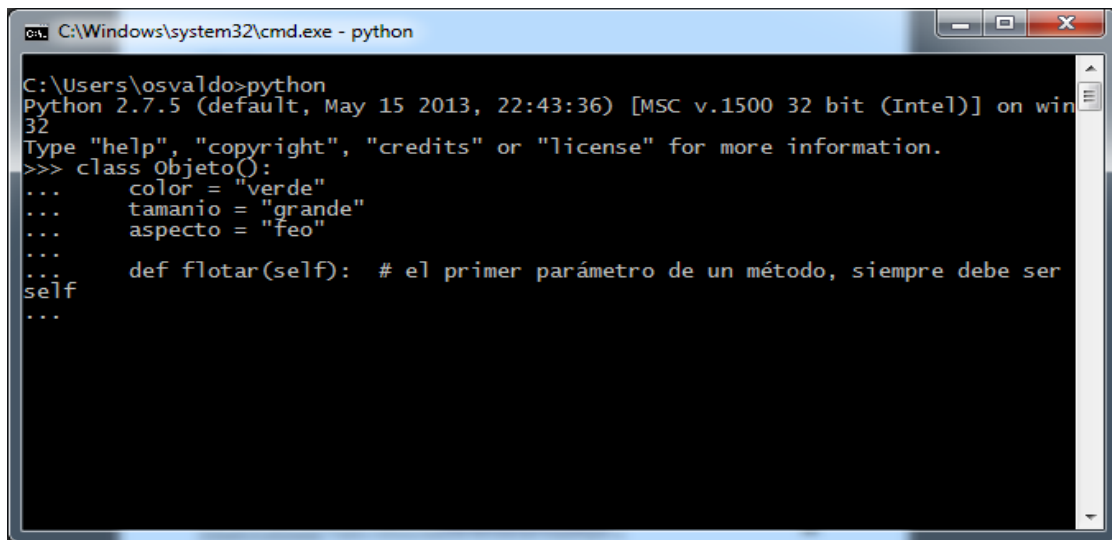
**Fuente:** Elaboración propia (2015).

### 3.5.5 Clases

En general, una clase en programación es una plantilla genérica para la creación de objetos de datos.

En Python una clase se define mediante la palabra reservada `class`, seguido del nombre de la clase y si hereda de otra clase el nombre.

En una clase, las funciones son llamadas “métodos” y las variables se les da el nombre de “propiedades”.

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe - python". The window shows the following text:

```
C:\Users\osvaldo>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> class Objeto():
...     color = "verde"
...     tamaño = "grande"
...     aspecto = "feo"
...
...     def flotar(self): # el primer parámetro de un método, siempre debe ser
self
...
>>>
```

**Figura 3.6.** Definición de la clase Objeto.  
**Fuente:** Elaboración propia (2015).

### 3.5.6 Estructuras de control de flujo

Una estructura de control, es un bloque de código que permite agrupar instrucciones de manera controlada, (Bahit, 2012). En Python las estructuras de control se clasifican de la siguiente manera: Estructuras de control condicionales y las iterativas, a continuación describiré cada una.

#### 3.5.6.1 Identación

Para entender las estructuras de control, primero hay que explicar qué es la indentación.

Con base en el **PEP 8** se usa la indentación para indicar que una serie de instrucciones pertenecen a una misma estructura de control y deben anteceder **cuatro (4) espacios en blanco**.

No en todos los lenguajes de programación es obligatoria la indentación, sin embargo, por cuestiones de estilo y para mayor legibilidad del código es altamente recomendable su uso. En Python **la indentación es obligatoria** ya que de esto depende la estructura del bloque de código.

Por lo tanto la definición de una estructura de control en Python sería:

*Inicio de la estructura de control:  
Expresiones*

### **3.5.6.2 Operadores**

Como en la mayoría de los lenguajes de programación, Python también cuenta con operadores, estos operadores son símbolos que nos permiten realizar funciones en específico. Existen tres tipos de operadores en Python, operadores aritméticos (para realizar operaciones básicas), los operadores relacionales (para realizar comparaciones) y los operadores lógicos que como su nombre lo indica realizan operaciones lógicas.

#### **3.5.6.2.1 Operadores Aritméticos**

Los operadores aritméticos nos permiten realizar operaciones básicas:

<b>Símbolo</b>	<b>Significado</b>	<b>Ejemplo</b>	<b>Resultado</b>
+	Suma	$x = 5 + 5$	$x = 10$
-	Resta	$x = 30 - 15$	$x = 15$
-	Negación	$x = -2$	$x = -2$
*	Multiplicación	$x = 3 * 5$	$x = 15$
**	Exponente	$x = 2^{**}2$	$x = 4$
/	División	$x = 10.5 / 2$	$x = 5.25$
//	División entera	$x = 10.5 / 2$	$x = 5.0$
%	Módulo	$x = 27 \% 4$	$x = 3$

#### **3.5.6.2.2 Operadores Relacionales**

Los operadores relacionales nos permiten realizar evaluaciones a sentencias, el resultado de dicha evaluación siempre será de tipo booleano:

<b>Símbolo</b>	<b>Significado</b>	<b>Ejemplo</b>	<b>Resultado</b>
==	Idéntico que	10 == 11	Falso
!=	Diferente que	X != Y	Verdadero
<	Menor que	3 < 4	Verdadero
>	Mayor que	18 > 100	Falso
<=	Menor o igual que	40 <= 40	Verdadero
>=	Mayor o igual que	8 >= 15	Falso

### 3.5.6.2.3 Operadores Lógicos

Los operadores lógicos nos permiten hacer evaluaciones de múltiples sentencias:

<b>Operador</b>	<b>Ejemplo</b>	<b>Resultado</b>
AND (Y)	3 == 3 AND 5 < 10	0 Y 0
	3 < 5 AND 8 > 3	1 Y 1
	5 < 30 AND 3 > 20	1 Y 0
OR (O)	10 == 10 OR 1 > 5	1 o 0
	4 < 6 OR 40 > 34	1 o 1
XOR (O excluyente)	5 == 5 XOR 1 < 3	1 o 1
	1 < 6 XOR 30 > 40	1 o 0

### 3.5.6.3 Estructuras de control condicionales

Las estructuras de control condicionales evalúan sentencias para poder ejecutar un fragmento de código en específico. Este tipo de estructuras nos permiten decidir la manera en que queremos que se comporte nuestro programa dependiendo del resultado de la condición que puede ser Falso o Verdadero.

## **IF**

Si la condición se cumple, el bloque será ejecutado:

```
if condición:  
    Bloque
```

## **IF ELSE**

Si la condición se cumple se ejecuta el bloque 1, de lo contrario se ejecutará el bloque 2:

```
If condición:  
    Bloque 1  
Else:  
    Bloque 2
```

Para abreviar varios **if** y **else** anidados se utiliza la siguiente estructura:

## **IF ELIF [ELSE]**

```
If condición 1:  
    Bloque 1  
Elif condición 2:  
    Bloque 2  
Elif condición 3:  
    Bloque 3  
Else:  
    Bloque 4
```

#### **3.5.6.4 Estructuras de control iterativas**

Este tipo de estructuras nos sirven para repetir una o varias veces un bloque de instrucciones mientras se cumpla una condición.

#### **FOR**

El bucle **for** ejecuta un bloque de código un número predeterminado de veces:

**For** variable **in** lista (o cadena):

Bloque

#### **WHILE**

El bucle **while** evalúa una condición y **mientras** ésta sea verdadera, el bloque de instrucciones dentro del bucle se ejecutará:

**While** condición:

Bloque

Si la condición es falsa desde el principio, el bucle no se ejecuta y continúa con las siguientes instrucciones.

#### **3.5.7 Comentarios**

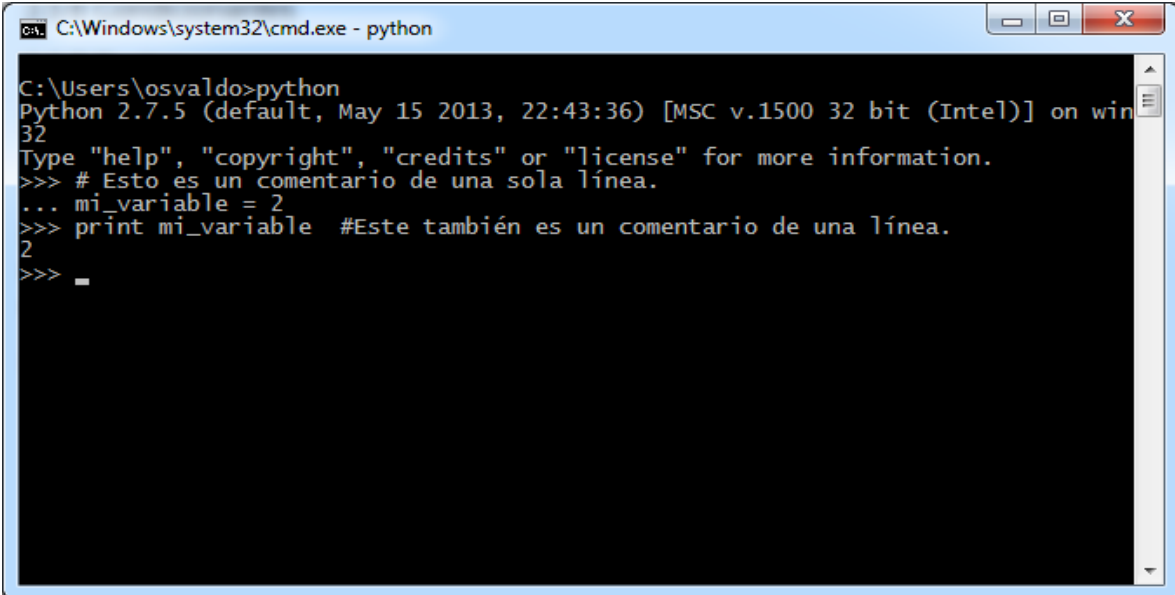
En medida que nuestros programas van creciendo, el programador necesita de notas en su lenguaje natural que expliquen el comportamiento de un segmento de código, en programación estas notas son llamadas comentarios.

El intérprete de Python omite los comentarios al momento de ejecutar el programa, por lo que se vuelve una herramienta muy útil para el programador y facilita la lectura y documentación del programa.



Hay dos formas de escribir un comentario:

Comentario de una línea:

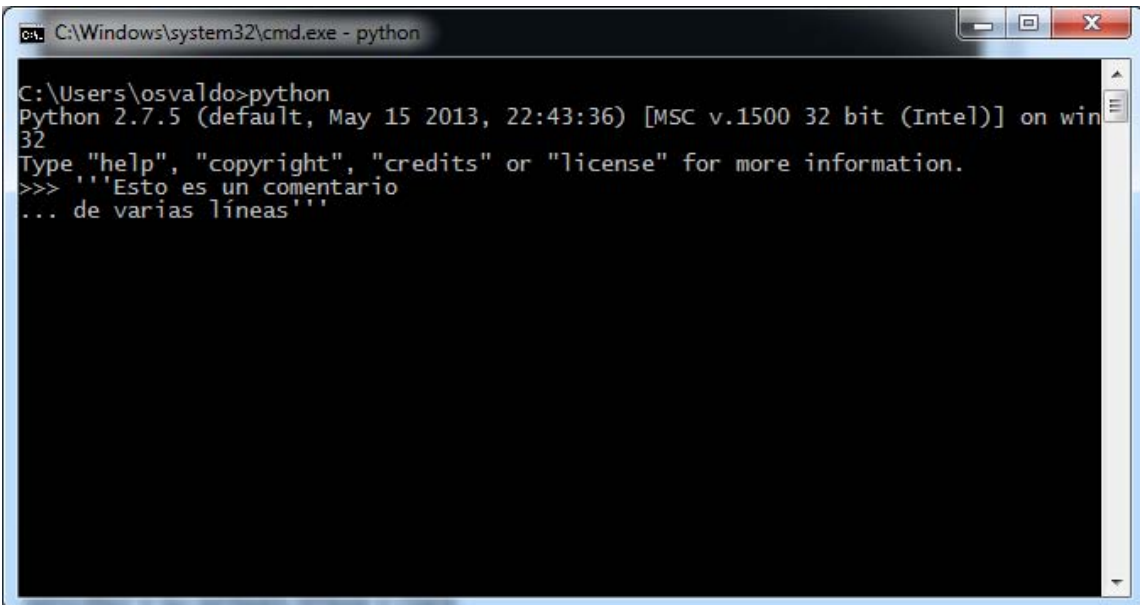


```
C:\Windows\system32\cmd.exe - python
C:\Users\osvaldo>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> # Esto es un comentario de una sola línea.
... mi_variable = 2
>>> print mi_variable #Este también es un comentario de una línea.
2
>>> _
```

**Figura 3.7.** Comentario de una sola línea.

**Fuente:** Elaboración propia (2015).

Comentario de varias líneas:



```
C:\Windows\system32\cmd.exe - python
C:\Users\osvaldo>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> '''Esto es un comentario
... de varias líneas'''
```

**Figura 3.8.** Comentario de varias líneas.

**Fuente:** Elaboración propia (2015).

### 3.6 PROGRAMACIÓN ORIENTADA A OBJETOS

Como ya se explicó en el capítulo anterior Python es uno de los lenguajes multiparadigma más poderosos que existen gracias a su sintaxis limpia y a su versatilidad ya que se pueden desarrollar aplicaciones robustas con pocas líneas de código.

La Programación Orientada a Objetos (POO o OOP por sus siglas en inglés) es el paradigma<sup>13</sup> de programación más usado en los últimos años debido a que representa una nueva forma de pensar, es una forma de descomponer problemas y por supuesto una forma diferente de afrontar la solución a un problema, aunque al principio nos resulte un poco difícil pensar en objetos empezaremos precisamente por definirlo.

¿Qué es un Objeto? Es simple, ***un objeto es una cosa***, un objeto puede ser la llave de tu casa, el auto que manejas, tu mascota también es un objeto e incluso tu vecino representa un objeto, en pocas palabras una cosa es un sustantivo y por definición todo sustantivo a la hora de programar se puede visualizar como un objeto. Pero estos objetos también tienen características intrínsecas llamados atributos, realizan acciones o tareas llamados métodos e incluso los objetos tienen la capacidad de interactuar con otros objetos.

Es importante mencionar que la POO no se debe ver como nuevas características que se le añaden al lenguaje de programación, como ya se mencionó la POO es una filosofía de programación. Pero para entender mejor lo que es el paradigma orientado a objetos es necesario precisar algunos conceptos fundamentales como clase y objeto, a lo largo de esta sección también hablaremos sobre las características y ventajas de este potente paradigma que lo ha llevado a ser el más popular.

---

<sup>13</sup> Es un estilo de programación. Es una propuesta tecnológica adoptada por una comunidad de desarrolladores.

### 3.6.2 Clases, atributos y métodos

En el capítulo pasado, se explicó que una clase es una plantilla genérica que sirve para modelar e instanciar objetos, contiene las características (atributos) y el comportamiento (métodos) del objeto. Tomaremos como ejemplo un conjunto de objetos llamados autos, mi auto tiene atributos como el color, el modelo, la marca, transmisión, año, etc. Y comportamientos como el de arrancar, conducir y avanzar por decir algunos, sin embargo, mi auto no es igual que el de mi vecino y a pesar de que estos dos objetos son diferentes, siguen perteneciendo a la misma clase.

En Python para definir una clase se utiliza la palabra reservada `class` seguido del nombre de la clase (por convención la primera letra debe ser mayúscula), el símbolo dos puntos (`:`) y si lo que continua es una cadena de texto esta se tomara como la documentación de la clase o docstring.

```
class Auto:
    '''Cadena de documentación de clase'''
    #atributos y métodos

class Mascota:
    '''Cadena de documentación de clase'''
    #atributos y métodos

class Persona:
    '''Cadena de documentación de clase'''
    #atributos y métodos

class Ojo:
    '''Cadena de documentación de clase'''
    #atributos y métodos
```

Ahora ya comentábamos sobre las características intrínsecas de los objetos, que en la POO se conocen como atributos de clase y se representan por medio de variables, continuando con el mismo ejemplo tenemos:

```

class Auto:
    marca = ""
    anio = ""
    transmisión = ""

class Mascota:
    tamaño = ""
    edad = ""
    color = ""

class Persona:
    peso = ""
    edad = ""
    tamaño = ""

class Ojo:
    forma = ""
    color = ""
    tamaño = ""

```

Esta es la forma en cómo se representan las propiedades de los objetos en la POO, pero estos mismos objetos cuentan también con acciones o tareas que pueden realizar, en la POO se denominan métodos y básicamente son funciones, en el capítulo pasado vimos cómo se definen las funciones en Python, en este caso tomaremos como ejemplo la clase Auto:

```

class Auto:
    '''Definición de la clase Auto. '''
    def __init__(self, gasolina):
        self.gasolina = gasolina
        print "Tenemos", gasolina, "litros"

    def arrancar(self):
        if self.gasolina > 0:
            print "Arranca"
        else:
            print "No Arranca"

    def conducir(self):

```

```
if self.gasolina > 0:
    self.gasolina -= 1
    print"Quedan",self.gasolina,"litros"
else:
    print "No se mueve"
```

En el código anterior solo se inicializa la propiedad *gasolina* y los métodos *arrancar* y *conducir* cada uno con un bloque de instrucciones que realizar al momento de ser llamados, sin embargo, hay un tercer método declarado en nuestra clase, el método `__init__` que como su nombre lo indica sirve para inicializar el objeto una vez que este se haya instanciado. El método `__init__` es el método constructor de una clase en Python que se ejecuta automáticamente cuando se crea un objeto y tiene como función principal inicializar el objeto.

### 3.6.3 Objetos

Anteriormente se comentó que los objetos son “cosas”, cualquier cosa que podamos imaginar, sin embargo, en la POO un objeto es la instanciación de una clase, ya que la clase por sí misma es solo una plantilla genérica y para poder materializar las propiedades y los métodos con los que cuenta se tiene que crear un objeto.

Continuando con el ejemplo de la clase *Auto*, en el siguiente ejemplo se crea una instancia:

```
mi_coche = Auto(3)
```

Ahora ya se tiene un objeto llamado `mi_coche` el cual se instancia con el nombre de la clase más el parámetro *gasolina* que en este caso es equivalente a 3, con este objeto ya podemos tener acceso a los atributos y a los métodos de la clase de la siguiente manera:

```
>>> print mi_coche.gasolina
3
```

```

>>> mi_coche.arrancar()
Arranca
>>> mi_coche.conducir()
Quedan 2 litros
>>> mi_coche.conducir()
Quedan 1 litros
>>> mi_coche.conducir()
Quedan 0 litros
>>> mi_coche.conducir()
No se mueve
>>> mi_coche.arrancar()
No arranca
>>> print mi_coche.gasolina
0

```

### 3.6.4 Herencia

Una de las características más importantes de la POO es la herencia, anteriormente comentamos que existen objetos que comparten propiedades y no necesariamente son iguales pero siguen perteneciendo a la misma clase, aunado a esto los objetos también pueden agregar ciertas características e inclusive también se pueden agregar nuevos métodos, esto es la herencia en programación: ***hacer que una clase herede de otra***, tomando el ejemplo de la clase `Auto` tenemos que: si hacemos que la clase `MiAuto` (Subclase) herede de la clase `Auto` (Superclase), automáticamente estamos haciendo que `MiAuto` contenga todos los atributos y métodos de la superclase. También al hacer que se herede de una clase se le dice que se está “extendiendo de una clase”.

```

class Auto:
    '''Definición de la clase Auto.'''
    def __init__(self, gasolina):
        self.gasolina = gasolina
        print "Tenemos", gasolina, "litros"

```

```

def arrancar(self):
    if self.gasolina > 0:
        print "Arranca"
    else:
        print "No Arranca"

def conducir(self):
    if self.gasolina > 0:
        self.gasolina -= 1
        print"Quedan",self.gasolina,"litros"
    else:
        print "No se mueve"

class MiAuto (Auto):
    pass

class TuAuto (Auto):
    pass

```

Para indicar que una clase está heredando de otra se coloca el nombre de la clase de la que se hereda dentro de un paréntesis después del nombre de la clase.

En Python a diferencia de otros lenguajes orientados a objetos, podemos hablar de **Herencia Múltiple** que como su nombre lo indica, se puede heredar de más de una clase al mismo tiempo, por ejemplo si tenemos un cocodrilo, este tiene características de animales terrestres, sin embargo, al mismo tiempo cuenta con características de animales acuáticos, para indicar que se está heredando de varias clases se hace así:

```

class Cocodrilo(Terrestre, Acuatico):
    pass

```

Cabe mencionar que aunque otros lenguajes como Java, C# y Ruby no cuentan con esta característica, tienen otras formas que arrojan el mismo resultado.

### 3.6.5 Encapsulamiento

El encapsulamiento es una forma de ocultar el estado del objeto para que sea el mismo objeto el único que pueda acceder y/o modificar sus atributos y métodos, es una forma de abstracción y se trata de una característica de la POO muy importante ya que convierte al objeto en una caja negra evitando que sus datos “sean visibles” para cualquier otro objeto.

En otros lenguajes orientados a objetos como Java existen los llamados “modificadores de acceso”: **private**, **public** y **protected** los cuales dan niveles de acceso a los atributos y métodos del objeto. En Python los modificadores de acceso no existen, para especificar que un método o atributo es privado basta con anteponer dos guiones bajos al nombre del método o atributo, para indicar que el método o atributo es público se define como ya lo hemos visto.

```
class Encapsulado:
    def publico(self):
        print "Método Público"
    def __privado(self):
        print "Método Privado"

>>> ejemplo = Encapsulado()
>>> ejemplo.publico()
Método Público
>>> ejemplo.__privado()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: Encapsulado instance has no
attribute '__privado'
```

Como se muestra en el ejemplo, el objeto no puede acceder al método privado aunque éste se encuentre correctamente definido. Otra forma de encapsular el estado de la clase es dar acceso a los métodos y/o atributos controladamente mediante los conocidos *setters* y *getters*, este mecanismo consta de declarar métodos que se dedican estrictamente a obtener y a establecer los valores de los atributos a los que queremos tener el acceso controlado.



Por convención estos métodos se les asignan el nombre anteponiendo el prefijo 'get' o 'set' dependiendo de la función que vaya a realizar.

```
class Fecha():
    def __init__(self):
        self.__dia = 1

    def getDia(self):
        return self.__dia

    def setDia(self, dia):
        if dia > 0 and dia < 31:
            self.__dia = dia
        else:
            print "Error"
```

En el ejemplo se muestra la definición de los métodos set y get, que a pesar de que el atributo `__dia` es privado, los métodos `getDia` y `setDia` tienen acceso a él, debido a que estos pertenecen a la misma clase. En resumen lo que hace el encapsulamiento es aislar el objeto del exterior con el propósito de proteger los datos contra modificaciones e incluso la eliminación por elementos que no tengan derecho a acceder a ellos por lo que no tendremos que preocuparnos por efectos secundarios.

### 3.6.6 Polimorfismo

Es la capacidad que tienen los objetos de distintas clases a responder al mismo mensaje.

Debido a que Python es un lenguaje de tipado dinámico y no restringe el tipo de dato que se le pasa a un método o función, se dice que el polimorfismo en Python no existe o no tiene mucha importancia a diferencia de los lenguajes altamente tipados o también conocidos como lenguajes de tipado estático como Java o C++, por ejemplo, cuando se llama a un método del objeto A independientemente de que este se comporte como se tiene planeado, y queremos llamar a un método X()

del objeto B que se pasó como parámetro anteriormente al objeto A, este método obviamente debe existir en el objeto B, debido a esto se dice que no existe el polimorfismo. Hay algunos textos que llaman polimorfismo en Python a la sobrecarga de métodos (overload) el cual consiste en tener dos o más métodos del mismo nombre pero con funciones diferentes, lo que los hará diferentes para el compilador independientemente de la función que realicen, será el número de parámetros que estos reciban.

### 3.7 SUBPROCESAMIENTO MÚLTIPLE

Hasta el momento hemos mostrado algunos ejemplos de programas **secuenciales**, es decir, los ejemplos hasta el momento ejecutan las instrucciones de forma ordenada (una detrás de la otra) de manera que la salida de la primer instrucción es la entrada de la siguiente y así sucesivamente hasta el final del programa, esto en programación se llama **Programación Estructurada**. La programación estructurada es un paradigma de programación el cual se basa en utilizar sólo 3 estructuras: secuencia, selección (if y switch) e iteración (for y while).

Hoy en día la mayoría de los lenguajes utilizan estas estructuras de control secuenciales que como ya comentamos solo permite al programador ejecutar una sola acción a la vez. Y ¿Qué pasa si queremos ejecutar más de una instrucción a la vez? Por ejemplo, el cuerpo humano realiza infinidad de tareas al mismo tiempo o **en paralelo** como la respiración, la digestión, el habla, los cinco sentidos, entre otras. De forma similar lo hacen las computadoras, actualmente podemos leer un archivo, descargar videos, enviar correos electrónicos y mandar un archivo a la impresora, todas estas operaciones las realiza la computadora de manera **concurrente**. Esto se logra a través de los múltiples procesadores con los que cuenta una computadora, pero con las computadoras que sólo cuentan con un solo procesador hacen la simulación de esta tarea. La concurrencia se implementa

por medio de funciones primitivas del sistema operativo a las que sólo el programador experto tiene acceso a ellas.

Python a diferencia de otros lenguajes que no cuentan con las capacidades integradas de subprocesamiento como C y C++, pone a nuestra disposición estas funciones primitivas a través del propio lenguaje y de sus bibliotecas. En esta sección hablaremos de las aplicaciones de la **programación concurrente**, sus conceptos fundamentales y por supuesto la implementación de **hilos** y **procesos** en Python, veremos también los problemas a los que se enfrenta Python para poder brindar un correcto funcionamiento con este paradigma.

Cabe mencionar que desarrollar aplicaciones concurrentes supone una tarea difícil y propensa a errores. También mencionar que el uso de hilos y procesos o como en esta sección se les dirá **Multithreading** y **Multiprocessing** respectivamente, está asociado con la velocidad y simultaneidad con la que se ejecutan los programas de computadora, por lo que se requiere de un alto conocimiento de programación e incluso conocimiento en desarrollo de hardware para alcanzar su correcto funcionamiento. Por último PySMG también hace uso de este paradigma.

### **3.7.2 Programación concurrente**

La programación concurrente o computación concurrente se define como la capacidad para ejecutar múltiples tareas interactivas al mismo tiempo, las cuales pueden ser un conjunto de **procesos** o **hilos de ejecución** creados por un único programa. Para comprender mejor este paradigma y antes de enfocarnos al subprocesamiento múltiple en Python, hay que definir algunos conceptos fundamentales:

**Concurrencia:** es un término utilizado en computación para hacer referencia a la ejecución de múltiples sucesos o eventos simultáneamente dentro del mismo contexto de un programa de computadora.

**Hilo de ejecución:** es la unidad mínima de procesamiento ejecutada o planificada por un sistema operativo, también es conocido como hebra o subproceso.

**Proceso:** en este contexto lo podemos definir como un programa en ejecución independiente a los propios del sistema operativo, cuenta con recursos propios como memoria, archivos e incluso subprocesos.

**Programa secuencial:** como ya lo comentamos anteriormente un programa secuencial es aquel que ejecuta sus instrucciones una detrás de la otra hasta su finalización.

**Programa concurrente:** es aquel que se diseña para tener dos o más contextos de ejecución activos simultáneamente, es decir, un programa que parece que varias partes del mismo se ejecutan al mismo tiempo. Se dice que estos programas son multihilo.

Como ya se mencionó los procesos son concurrentes si existen simultáneamente y la concurrencia se vuelve un punto clave para el diseño de sistemas operativos. La concurrencia comprende un gran número de cuestiones de diseño, incluyendo la comunicación entre procesos, compartición y competencia por los recursos, sincronización de la ejecución de varios procesos y asignación del tiempo de procesador a los procesos.

### **3.7.3 Multithreads en Python**

Los threads son hilos de ejecución simultánea en un mismo proceso, comparten espacio de memoria y otros recursos entre ellos, debido a esto son más ligeros, menos costosos de instanciar y se vuelven más manejables desde la aplicación. En la mayoría de los lenguajes el uso de threads se vuelve impredecible y en ocasiones suele causar algunos problemas debido a que se “pelean” por el orden en el que acceden a los recursos compartidos, es cuando se debe pensar en la sincronización para mejorar la comunicación entre los threads.

Python toma una solución drástica a este problema de sincronización con la implementación del GIL (Global Interpreter Lock por sus siglas en inglés) el cual es el encargado de la ejecución de los threads en Python. El GIL es un mecanismo utilizado por el intérprete de CPython<sup>14</sup> para asegurar que uno y solo un hilo ejecuten bytecode<sup>15</sup> a la vez, haciendo al modelo de objetos implícitamente más seguro contra el acceso concurrente. En pocas palabras cuando a un thread le toca ejecutarse, obtiene un lock (bloqueo) a nivel de intérprete que no permite la ejecución de otro thread de forma simultánea.

Esto a simple vista es bueno ya que podemos dejar de preocuparnos por la sincronización de threads, sin embargo, nos encontramos con la principal problemática que tiene CPython con la implementación de threads ya que hace que la aplicación o programa multihilo se ejecute en un solo núcleo o procesador, esto sin importar si la computadora cuente con más de un núcleo, esto provoca que el rendimiento de nuestra aplicación se reduzca con cada llamada a un nuevo thread.

Se puede decir entonces que CPython hace “trampa” creando una ilusión al usuario al suponer que las tareas se ejecutan al mismo tiempo ya que el procesador alterna la ejecución de cada una de ellas rápidamente. Esta ilusión creada por el GIL es necesaria ya que la gestión de memoria en CPython no es segura para múltiples hilos.

Dejando a un lado por un rato al GIL veamos cómo se implementan los threads en CPython. En CPython para trabajar con threads se utiliza el módulo `thread`. Este módulo brinda primitivas de **bajo nivel** para trabajar con múltiples hilos. Además de `thread`, contamos también con el módulo `threading` el cual está basado en el primero, brindándonos una API de alto nivel, más completa y orientada a objetos. El módulo `threading` está basado en el modelo de threads de Java.

---

<sup>14</sup> Es la aplicación canónica del lenguaje Python.

<sup>15</sup> Código intermedio más abstracto que el código máquina.

El módulo `threading` contiene una clase `Thread` la cual se debe extender para poder trabajar con hilos. A continuación veremos un ejemplo de cómo se trabajan los hilos en Python:

```
import threading

class MiThread(threading.Thread):
    def __init__(self, num):
        threading.Thread.__init__(self)
        self.num = num

    def ejecuta(self):
        print "Soy el hilo", self.num

print "Soy el hilo principal"

for i in range(0, 10):
    t = MiThread(i)
    t.start()
    t.join()
```

En el ejemplo se crea la clase `MiThread` heredando del módulo `threading`. El método `ejecuta()` imprime el número de hilo correspondiente. Se instancia nuestra clase y mediante al método `start()` empieza a correr el hilo. Por último se llama al método `join()` el cual sirve para bloquear al hilo mediante el cual se hace la llamada al nuevo hilo y finalice. Esto sirve para asegurar que el hilo principal no termine antes de los hilos hijos.

Es importante mencionar que existen otras implementaciones de Python que contienen GIL, como lo son Jython e Iron Python.

### 3.7.4 Multiprocessing en Python

Como se observa el uso de threads en Python puede llegar a representar problemas con el rendimiento de nuestra aplicación ya que sin importar el número de núcleos con los que cuente nuestra computadora, el GIL ejecutará los threads en un solo núcleo.

La solución a este problema la brinda la propia documentación de Python. La cual dice que si queremos que nuestra aplicación haga un mejor uso de los recursos computacionales de los ordenadores multi-núcleo, aconseja utilizar el módulo `multiprocessing` (Recursos Python, 2013).

Como su nombre lo dice `multiprocessing` utiliza procesos completos con el beneficio de la comunicación entre procesos, colas de tareas y la sincronización de los mismos ya que su API está basado en el módulo de `threading`.

Con todas estas ventajas la programación multiproceso se convierte en una solución óptima, limpia y eficiente (Recursos Python, 2013). Veamos un ejemplo:

```
from multiprocessing import Process
import os
import time

class MiProceso():
    def dormir(self, nombre, segundos):
        print 'empezando el proceso hijo con id: ',
            os.getpid()
        print 'proceso padre: ', os.getppid()
        print 'durmiendo por %s' % segundos
        time.sleep(segundos)
        print 'termino de dormir'

    def correr(self):
        print 'proceso padre (id %s)' % os.getpid()
        p = Process(target=self.dormir,
                    args=('osvaldo', 5))
        p.start()
        print "proceso padre despues de iniciar
            proceso hijo"
```

```

        print "proceso padre a punto de unirse a
                proceso hijo"
    p.join()
    print "el proceso padre e hijo se unieron"
    print "proceso padre con id: ", os.getpid()
    print "proceso principal (padre de los
            padres): ", os.getppid()

if __name__ == '__main__':
    proceso = MiProceso()
    proceso.correr()

```

El código anterior ejemplifica el uso de los multiprocesos, muy similar al uso de Threads. La idea es muy simple, el método `correr()` crea el proceso padre, al igual que en el ejemplo pasado se ejecuta mediante el método `start()`, sin embargo en este ejemplo primero se “duerme” al proceso hijo por un lapso de cinco (5) segundos, después se bloquea el proceso padre mediante `join()` para que no termine antes que el proceso padre.

### 3.7.5 Conclusión

En resumen estas dos técnicas son bastante útiles cuando queremos optimizar nuestra aplicación ya que podemos ejecutar varias tareas a la vez y como ya comenté están asociadas a la velocidad de ejecución y al a simultaneidad de procesos. Por ningún motivo se debe pensar que estas dos técnicas están peleadas, por el contrario, se debe tener un amplio conocimiento de ellas para lograr un óptimo rendimiento de nuestras aplicaciones en cualquier computadora donde sea ejecutada.

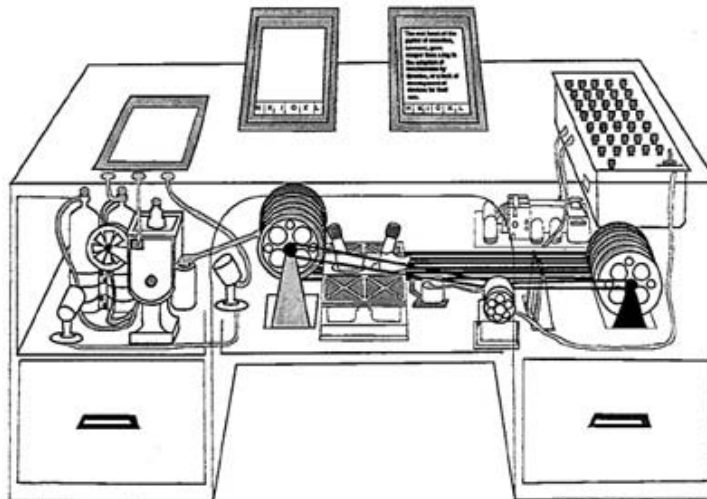
Podemos decir que el Multithreading es el indicado para aplicaciones pequeñas y medianas. Debemos empezar a pensar que el GIL es una herramienta que evita que nuestra aplicación se vuelva inestable y olvidarnos de las “inconveniencias” que presenta. Por otra parte el Multiprocessing es el indicado si queremos desarrollar aplicaciones que aprovechen al máximo los recursos computacionales de nuestro ordenador.



### 3.8 INTERFACES GRÁFICAS DE USUARIO (GUI)

Las Interfaces Gráficas de Usuario (GUI por sus siglas en inglés) son herramientas que sirven para crear interfaces gráficas, es decir, elementos gráficos con los cuales el usuario puede interactuar con la aplicación.

La historia de las GUI es muy extensa y se remonta hacia los años de 1930 cuándo Vennevar Bush diseña el *MEMEX* el cual tenía aspecto de un escritorio que contaba con dos pantallas, un escáner y un teclado. El objetivo de este dispositivo era crear una base de conocimiento y que el usuario tuviera acceso a esta información con unos cuantos “clics”.



**Figura 4.2.** Diseño de Memex.

**Fuente:** <https://www.etsisi.upm.es>. 1945.

Las GUI representan un parteaguas en la historia de la computación ya que cambió la forma de pensar, de ver y hasta la forma en que el usuario se relaciona con la computadora. Es hasta la década de los 80's que aparecen las primeras interfaces gráficas completas como GEM y WINDOWS 1.0.

Actualmente los usuarios no se tienen que enfrentar a descifrar códigos binarios de programas de computadora para poder entender el contenido o interactuar directamente con la máquina y esto se lo debemos a las interfaces gráficas de usuario.

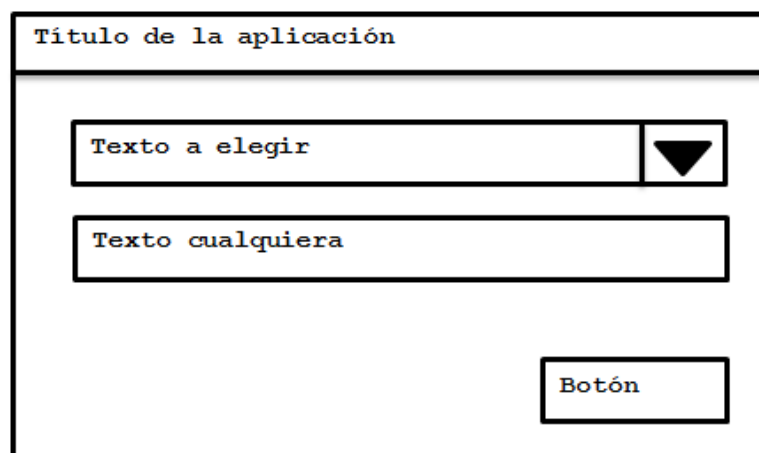
### 3.8.2 Las GUI en Python

Durante el capítulo pasado se habló sobre las características del lenguaje, de la potencia que tiene por ser de alto nivel y lo importante que significa el ser multiplataforma, también en este capítulo ya vimos algunos ejemplos de Python utilizando el paradigma orientado a objetos y evidentemente en esta sección vamos a hablar de las librerías gráficas que tiene Python.

Existen muchas librerías gráficas en Python que brindan herramientas para desarrollar nuestras propias GUI's, incluso también podemos encontrar IDE's<sup>16</sup> que tienen alguna de estas librerías preinstaladas y que facilitan aún más el hecho de armar nuestra interfaz. En nuestro caso no utilizaremos ningún IDE para nuestro software, sin embargo, nos encontramos con la difícil tarea de elegir que librería escoger.

A continuación veremos las cuatro librerías gráficas de Python más importantes: Tkinter, wxPython, PyQt y PyGTK.

Describiendo algunas de sus ventajas y desventajas que tiene cada librería, además de mostrar un ejemplo sencillo de interfaz con algunos elementos básicos como botones, cajas de texto, etiquetas y por supuesto la ventana. Este es un maqueta de cómo se verán los programas.



**Figura 4.3.** Maqueta de las aplicaciones con librerías gráficas.

**Fuente:** Elaboración propia (2015).

<sup>16</sup> IDE o Entorno de Desarrollo Integrado es una aplicación que facilita al programador el desarrollo de software.

### 3.8.2.1 Tkinter

Es la librería gráfica de Python basada en Tcl/Tk<sup>17</sup>, esta librería gráfica viene instalada por default en la mayoría de las versiones, es multiplataforma y es excelente para proyectos ligeros y para el aprendizaje del lenguaje.

Ventajas:

Está preinstalada en la mayoría de las versiones.

Es Multiplataforma.

Documentación completa.

Fácil aprendizaje e ideal para proyectos ligeros.

Desventajas:

Tiene una pequeña selección de widgets (elementos gráficos).

Lenta debido a que dibuja cada elemento.

Apariencia extraña ya que no es nativa de ninguna plataforma.

No se tiene control total de la interface.

Ejemplo:

```
import Tkinter as tk

class Ejemplo Tkinter(tk.Frame):
    ''' Ejemplo de app con Tkinter. Instancia
        y llama al método run para ejecutar. '''
    def __init__(self, master):
        # Se crea la ventana principal
        tk.Frame.__init__(self,
                           master,
                           width=300,
                           height=200)
```

---

<sup>17</sup> Lenguaje de script utilizado para el desarrollo de interfaces gráficas.

```

# Se establece el titulo
self.master.title('Ejemplo de Interfaz con TkInter')

# Permite que el tamaño de la ventana se modifique
self.pack_propagate(0)

self.pack()

# Selector de saludos
# Se crea una variable para los saludos
self.greeting_var = tk.StringVar()
self.greeting = tk.OptionMenu(self,
                               self.greeting_var,
                               'Hola',
                               'Adios',
                               'Que Onda?')

self.greeting_var.set('Hola')

# Se crea el campo de texto
self.recipient_var = tk.StringVar()
self.recipient = tk.Entry(self,

textvariable=self.recipient_var)
self.recipient_var.set('Cualquier Texto')

# Se crea boton
self.go_button = tk.Button(self,
                            text='Boton',
                            command=self.print_out)

# Se colocan los elementos
self.go_button.pack(fill=tk.X, side=tk.BOTTOM)
self.greeting.pack(fill=tk.X, side=tk.TOP)
self.recipient.pack(fill=tk.X, side=tk.TOP)

def print_out(self):
    ''' Se imprime el saludo completo
        con la seleccion hecha por el usuario. '''
    print('%s, %s!' % (self.greeting_var.get().title(),
                      self.recipient_var.get()))

def run(self):

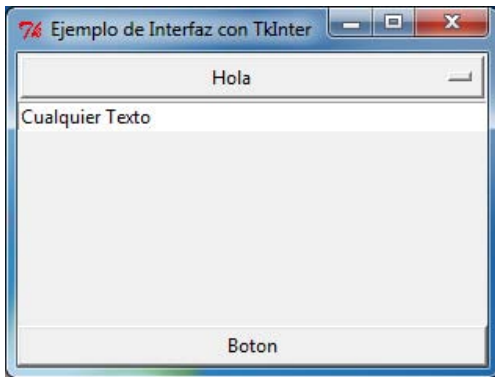
```

```
''' Ejecuta la app '''  
self.mainloop()
```

```
app = EjemploTkinter(tk.Tk())  
app.run()
```

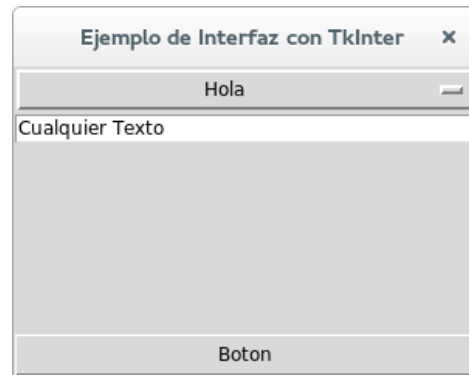
Interfaz:

Windows



**Figura 4.4.** Tkinter en Windows  
**Fuente:** Elaboración propia (2015).

Linux



**Figura 4.5.** Tkinter en Linux  
**Fuente:** Elaboración propia (2015).

### 3.8.2.2 wxPython

Es la librería gráfica basada en wxWidgets<sup>18</sup> para Python, en palabras del propio Guido Van Rossum wxPython hubiera sido la librería gráfica de Python si no existiera Tkinter. Cuenta con una considerable cantidad de elementos gráficos nativos en todas las plataformas, probablemente la librería gráfica más popular junto con pyGTK.

Ventajas:

Es multiplataforma.

Elementos nativos en todas las plataformas.

Documentación completa y múltiples ejemplos.

Control sobre el comportamiento de la interfaz.

---

<sup>18</sup> Bibliotecas multiplataforma y libres que sirven para el desarrollo de interfaces gráficas programadas en C++.

De fácil instalación en Windows y Linux.

Desventajas:

No está preinstalada en Python.

Su API <sup>19</sup> es anti-pythonica.

Relativamente más difícil de aprender.

Mala documentación y soporte escaso.

Ejemplo:

```
import wx

class EjemploWxpython(wx.Frame):
    def __init__(self):
        # Todas las apps de wx deben crear un objeto
        # Antes de hacer cualquier cosa.
        self.app = wx.App()

        # Configuramos la ventana principal
        wx.Frame.__init__(self,
                           parent=None,
                           title='Ejemplo de Interfaz con
                                   wxPython',
                           size=(300, 200))

        # Saludos disponibles
        self.greetings = ['Hola', 'Adios', 'Que onda?']

        # Layout del panel
        self.panel = wx.Panel(self, size=(300, 200))
        self.box = wx.BoxSizer(wx.VERTICAL)

        # ComboBox de saludos
        self.greeting = wx.ComboBox(parent=self.panel,
                                    value='Hola',
                                    size=(280, -1),
                                    choices=self.greetings)
```

---

<sup>19</sup> Interfaz de Programación de Aplicaciones, contiene los procedimientos y métodos que contiene cierta biblioteca para ser usada en otro software.

```

# Agregamos el ComboBox al panel
self.box.Add(self.greeting, 0, wx.TOP)
self.box.Add((-1, 10))

# Creamos la caja de texto
self.recipient = wx.TextCtrl(parent=self.panel,
                             size=(280, -1),
                             value='Cualquier Texto')

# Agregamos la caja de texto
self.box.Add(self.recipient, 0, wx.TOP)

# Creamos la posicion del boton
self.box.Add((-1, 100))

# Creamos el boton
self.go_button = wx.Button(self.panel, 10, 'Boton')

# Creamos un evento para el boton
self.Bind(wx.EVT_BUTTON, self.print_result,
self.go_button)

# se asigna el evento al boton
self.go_button.SetDefault()

# Agregamos el boton al panel
self.box.Add(self.go_button, 0,
flag=wx.ALIGN_RIGHT|wx.BOTTOM)

# agregamos todos los elementos
self.panel.SetSizer(self.box)

def print_result(self, *args):
    ''' Se imprime el saludo
        con la seleccion del usuario. '''
    print('%s, %s!' % (self.greeting.GetValue().title(),
                    self.recipient.GetValue()))

def run(self):
    ''' Ejecuta la app '''

```

```

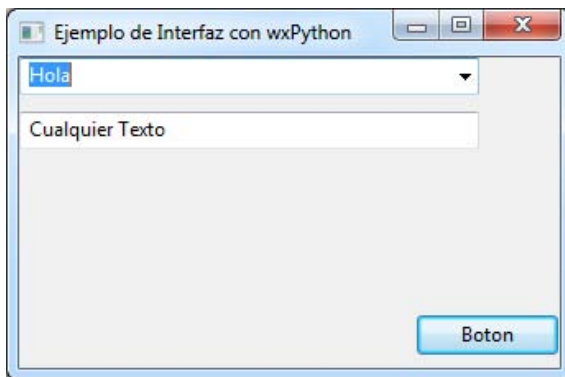
self.Show()
self.app.MainLoop()

# Se crea una instancia y se ejecuta
app = EjemploWxpython()
app.run()

```

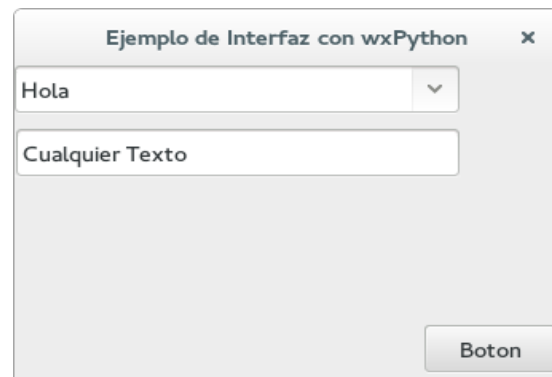
Interfaz:

Windows



**Figura 4.6.** wxPython en Windows  
Fuente: Elaboración propia (2015).

Linux



**Figura 4.7.** wxPython en Linux  
Fuente: Elaboración propia (2015).

### 3.8.2.3 PyQT

Es la librería gráfica basada en QT<sup>20</sup> desarrollada en C++, probablemente la menos popular de estas cuatro librerías, es más robusto y complicado de instalar, aunque cuenta con una gran cantidad de elementos gráficos los cuales en Windows y Linux son nativos.

Ventajas:

- Completo conjunto de elementos gráficos.
- Apariencia nativa en Windows y Linux.
- Control sobre la interfaz.

<sup>20</sup> Biblioteca usada para desarrollar aplicaciones GUI.



Desventajas:

- Es complicado de instalar.
- Poca documentación para Python.
- Código poco pythonico.
- Relativamente más complicado de aprender.

Ejemplo:

```
import sys
from PySide.QtCore import *
from PySide.QtGui import *

class EjemploPyqt(QDialog):
    ''' Ejemplo de interfaz con PyQt. Instancia
        y llama al metodo run para ejecutar. '''
    def __init__(self):
        # crea un Qt application -- todas las PyQt app lo
        # necesitan hacer
        self.qt_app = QApplication(sys.argv)

        # Saludos disponibles
        self.greetings = ['Hola', 'Adios', 'Que Onda?']

        # hace una llamada al constructor
        QDialog.__init__(self, None)

        # Configura la ventana principal
        self.setWindowTitle('Ejemplo de Interfaz con PyQt')
        self.setMinimumSize(300, 200)

        # Crea un Layout
        self.vbox = QVBoxLayout()

        # Combo de saludos
        self.greeting = QComboBox(self)
        # Agrega los saludos
        list(map(self.greeting.addItem, self.greetings))
```

```

# Crea la caja de texto
self.recipient = QLineEdit('Cualquier texto', self)

# Crea el boton
self.go_button = QPushButton('Boton')
# Crea el evento del boton
self.go_button.clicked.connect(self.print_out)

# agrega los elementos
self.vbox.addWidget(self.greeting)
self.vbox.addWidget(self.recipient)
self.vbox.addStretch(100)
self.vbox.addWidget(self.go_button)
self.setLayout(self.vbox)

def print_out(self):
    ''' Imprime el saludo creado
        por el usuario. '''
    print('%s, %s!' % (self.greetings[self.greeting.currentIndex()].title(),
                      self.recipient.displayText()))

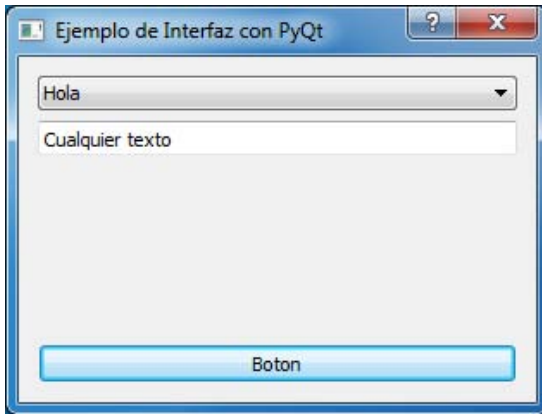
def run(self):
    ''' Ejecuta la app. '''
    self.show()
    self.qt_app.exec_()

app = EjemploPyqt()
app.run()

```

Interfaz:

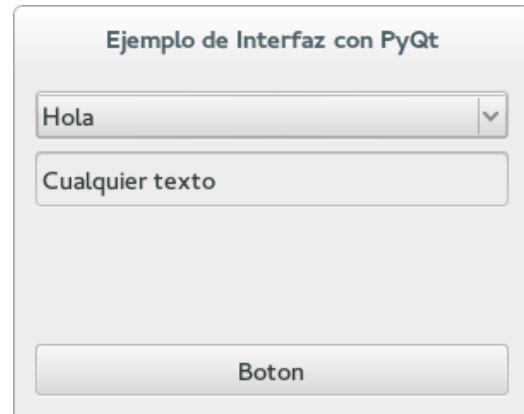
Windows



**Figura 4.8.** PyQt en Windows

**Fuente:** Elaboración propia (2015).

Linux



**Figura 4.9.** PyQt en Linux

**Fuente:** Elaboración propia (2015).

### 3.8.2.4 PyGTK

Es la librería basada en GTK<sup>21</sup> y como ya lo comentamos es una de las más populares en Python, es una de las más potentes y su API es bastante amplia, cuenta con una colección completa de elementos gráficos y en Linux tiene una apariencia nativa.

Ventajas:

Completo conjunto de elementos gráficos.

Apariencia nativa en Linux.

Probablemente es la que otorga mayor control sobre el comportamiento de la interfaz.

Es muy estable y facilita la depuración de código al mandar mensajes de error concretos.

---

<sup>21</sup> Es una biblioteca grafica utilizada para el desarrollo de GNOME.

Desventajas:

No viene preinstalado con Python.

Cuenta con la peor documentación de las cuatro.

En Windows cuenta con varias dependencias que se deben instalar por separado.

Relativamente más complicado de aprender.

Ejemplo:

```
import gtk

class EjemploPygtk(gtk.Window):
    ''' Ejemplo de aplicacion con PyGTK. Instancia
        y llama al metodo run para ejecutar. '''
    def __init__(self):
        # Inicializa la ventana
        gtk.Window.__init__(self)
        self.set_title('Ejemplo de Interfaz con pyGTK')
        self.set_size_request(300, 200)
        self.connect('destroy', gtk.main_quit)
        self.vbox = gtk.VBox()

        # Se crea combo dependiendo de la version
        if (gtk.gtk_version[1] > 24 or
            (gtk.gtk_version[1] == 24 and gtk.gtk_version[2]
> 10)):
            self.greeting = gtk.ComboBoxText()
        else:
            self.greeting = gtk.combo_box_new_text()
            self.greeting.append = self.greeting.append_text

        # Saludos.
        map(self.greeting.append, ['Hola', 'Adios', 'Que
Onda'])

        # Se crea caja de texto
        self.recipient = gtk.Entry()
        self.recipient.set_text('Cualquier Texto')
```

```

# Se crea boton
self.go_button = gtk.Button('Boton')
self.go_button.connect('clicked', self.print_out)

# Visualiza los elementos
self.vbox.pack_start(self.greeting, False)
self.vbox.pack_start(self.recipient, False)
self.vbox.pack_end(self.go_button, False)
self.add(self.vbox)

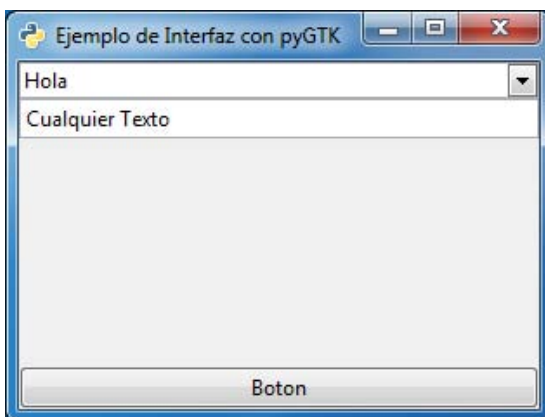
def print_out(self, *args):
    ''' Imprime el saludo creado. '''
    print('%s, %s!' % (self.greeting.get_active_text().title(),
                      self.recipient.get_text()))

def run(self):
    ''' Ejecuta la app. '''
    self.show_all()
    gtk.main()

app = EjemploPygtk()
app.run()

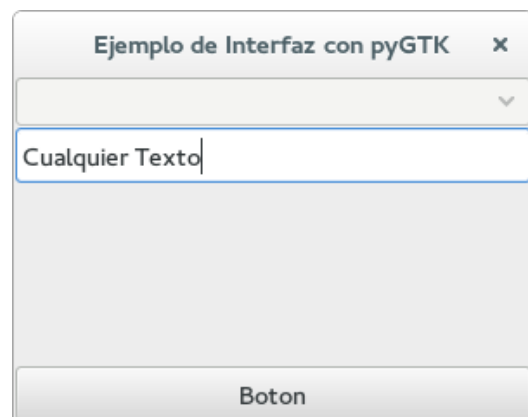
```

Interfaz:  
Windows



**Figura 4.10.** pyGTK en Windows  
**Fuente:** Elaboración propia (2015).

Linux



**Figura 4.11.** pyGTK en Linux  
**Fuente:** Elaboración propia (2015).

### **3.9 APLICACIONES DESARROLLADAS CON PYTHON**

En 2011, Linux Journal otorgó a Python por tercera vez el premio al mejor lenguaje de programación, votado por desarrolladores y usuarios, la revista señaló las ventajas del uso del lenguaje sobre sus competidores como la facilidad de uso, sencillez y su sintaxis limpia y clara.

La popularidad de Python ha crecido tanto que lo podemos encontrar en tantas aplicaciones como podamos imaginar y está presente en servicios y aplicaciones que utilizamos de manera habitual, tal es el caso del cliente oficial de Dropbox, el Software Center de Ubuntu, el gestor de e-books Calibre, el servicio de streaming Flomotion y también como ya se comentó el Sistema de Adquisición de Datos del Monitor de Neutrones 6-NM64. Gracias a que Guido Van Rossum actualmente trabaja en Google, YouTube y algunas aplicaciones de Google también utilizan Python.

El uso de Python en las grandes empresas del internet como Google, Yahoo! y las corporaciones tecnológicas como la NASA es una de las principales referencias de este lenguaje de programación.

Por supuesto el software descrito en este documento está desarrollado en Python, en su versión 2.7, como será descrito en el capítulo siguiente.

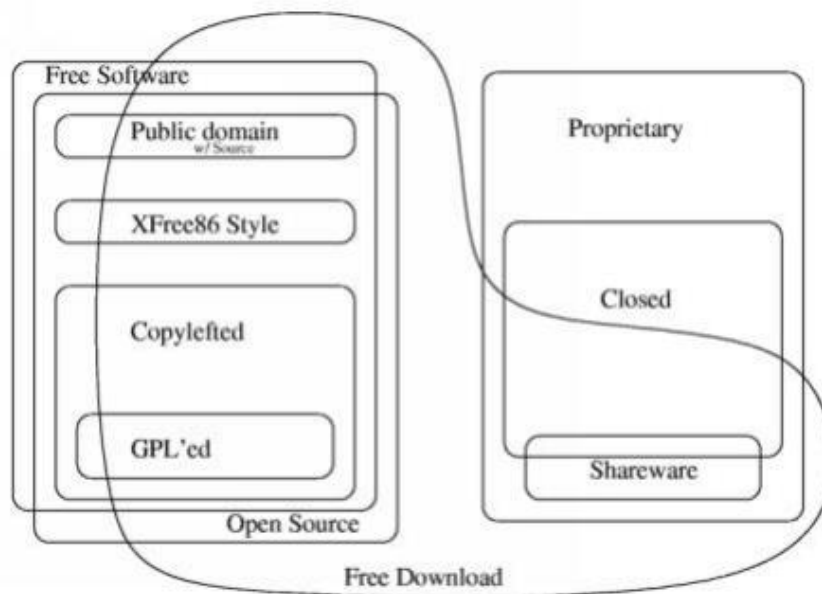
**CAPÍTULO 4:**  
**PYSMG**  
**(SOLAR MONITORS**  
**GRAPHER)**



## 4.1 CATEGORÍAS DEL SOFTWARE LIBRE Y NO LIBRE

*"Las obras de conocimiento deben ser libres,  
No hay excusas para que no sea así."  
Richard Stallman*

Uno de los puntos más importantes en el desarrollo de software es el licenciamiento, ya que básicamente se trata de un contrato entre el desarrollador y/o autor del programa y el usuario de computadora. Todos los programas de computadora se rigen bajo una *Licencia*<sup>22</sup> ya sean comerciales, libres o gratuitos.



**Figura 4.1.** Diagrama que explica las diferentes categorías del software.

**Fuente:** [http://doc.ubuntu-es.org/Licencias\\_de\\_software](http://doc.ubuntu-es.org/Licencias_de_software). 2015.

El cuadro anterior muestra las diferentes tipos de Licencia que podemos encontrar, en nuestro caso solo nos enfocaremos a dos categorías en especial: (1) Free Software y (2) Open Source. Ya que estos dos movimientos enfocan sus condiciones para que el usuario final de computadora sea beneficiado en todo momento.

<sup>22</sup> Es un contrato entre el autor de un software y el usuario que contiene una serie de términos y condiciones que se deben cumplir para su uso.



#### 4.1.1 Free Software

La *Free Software Foundation (FSF)* es una organización no lucrativa creada en 1985 por Richard Stallman principalmente para promover el uso, desarrollo y el movimiento del software libre además de eliminar las restricciones de copia, redistribución, entendimiento y modificación de software.

En 1986 la FSF lanza por primera vez el concepto de *software libre* la cual hace énfasis en el significado de la palabra *libre* ya que esta no se refiere a “gratis” sino a “libertad”. Lo define como un asunto de libertad para el usuario para poder copiar, redistribuir y modificar un programa y para lograr esto es indispensable que el código fuente esté disponible.

A través del tiempo se han ido realizando modificaciones a la definición de software libre en cuestiones legales, sociales y de filosofía del propio movimiento. Actualmente el software libre se define cómo:

Software libre es el software que respeta la libertad de los usuarios y la comunidad. En grandes líneas, significa que **los usuarios tienen la libertad para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software**. Es decir, el software libre es una cuestión de libertad, no de precio. Para entender el concepto, piense en libre como en libre expresión, no como en barra libre (Cliver, 1996).

También comprende los criterios que un software debe cumplir para que pueda ser considerado “software libre”, los cuales denomina como “libertades”:

**Libertad 0:** La libertad de ejecutar el programa como se desea, con cualquier propósito.

**Libertad 1:** La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que usted quiera.

**Libertad 2:** La libertad de redistribuir copias para ayudar a su prójimo.

**Libertad 3:** La libertad de distribuir copias de sus versiones modificadas a terceros. Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones.

Para que la libertad uno y tres se cumplan, el acceso al código fuente es una condición necesaria, por lo tanto, si el programa de computadora brinda al usuario/desarrollador estas cuatro libertades se puede decir que es software libre.

#### 4.1.2 Open Source Software

Sin embargo, el término de “*free software*” sigue siendo ambiguo sobre todo en el idioma inglés ya que “*free*” también significa “gratis” y como ya vimos en el punto anterior el término “*free software*” hace referencia a la libertad y no al precio.

Debido a esto en 1990 se empieza a utilizar el término de “*open source*” para tratar de eliminar la ambigüedad y para 1998 se crea la *Open Source Initiative (OSI)* y es lanzado el sitio: <http://www.opensource.org>.

La Open Source Initiative es una organización no lucrativa a nivel mundial creada por Eric Raymond y Bruce Perens a finales de febrero de 1998, la cual se encarga de promover el movimiento y dar mantenimiento a la definición de open source (OSD) la cual dice:

El código abierto no sólo significa el acceso al código fuente. Los términos de distribución de software de código abierto deben cumplir con los siguientes criterios:

**Libre redistribución:** La licencia no debe restringir a un tercero el vender o entregar el programa como parte de una distribución mayor que contiene programas de diferentes fuentes. La licencia no debe requerir una regalía u otras comisiones para esta venta.

**Código fuente:** El programa debe incluir el código fuente y debe permitir la distribución de código fuente, así como en forma compilada.

**Trabajos derivados:** La licencia debe permitir modificaciones y trabajos derivados y debe permitir que estos se distribuyan bajo los mismos términos que la licencia del software original.

**Integridad del código fuente del autor:** La licencia puede restringir que el código fuente se distribuya en forma modificada solo si la licencia permite la distribución de “archivos de revisión”.

**No discriminación hacia personas o grupos:** La licencia no debe discriminar a ninguna persona o grupo de personas.

**No discriminación hacia áreas de iniciativa:** La licencia no debe restringir a nadie a hacer uso del programa en un campo específico de actividad.

**Distribución de la licencia:** Los derechos asociados al programa deben aplicarse a todos aquellos a quienes se redistribuya el programa.

**La licencia no debe restringir otro software:** La licencia no debe poner restricciones sobre otros programas que se distribuyan junto con el software licenciado.

**La licencia debe ser tecnológicamente neutral:** Ninguna disposición de la licencia puede basarse en cualquier tecnología o el estilo de interfaz individual, (Open Source Initiative, 2015).

Al igual que el software libre, si un programa de computadora cumple con éstos criterios, entonces se puede considerar al programa dentro del movimiento Open Source.

### 4.1.3 Diferencia entre Free Software y Open Source

Actualmente se suele usar indistintamente el término Free Software y Open Source para referirnos al software que podemos obtener en la red y que para nada debe ser confundido con el llamado “freeware”<sup>23</sup>, sin embargo, ambos movimientos tienen sutiles diferencias que bien vale la pena mencionar.

Estos dos movimientos lucen a simple vista idénticos pero, ¿en realidad lo son? El movimiento Free Software y el Open Source tienen similitudes filosóficamente hablando ya que ambos defienden el derecho de un usuario que adquirió un producto de computadora para usar, copiar, modificar y redistribuir el software.

Para poder entender mejor las diferencias entre estos dos movimientos, a continuación se muestra un cuadro con los criterios que se deben cumplir para que un producto pueda pertenecer a estos movimientos:

**Tabla comparativa de filosofías**

Libertades del Free Software	Criterios del Open Source
Ejecutar el programa con cualquier propósito.	Libre redistribución: el software debe poder ser regalado o vendido libremente.
Estudiar y modificar el programa.	Código fuente: debe estar incluido u obtenerse libremente.
Distribuir el programa para ayudar al prójimo.	Trabajos derivados: la redistribución de modificaciones debe estar permitida.
Distribuir copias de las versiones modificadas.	Integridad del código fuente del autor: las licencias pueden requerir que las modificaciones sean distribuidas solo como parches.
	No discriminación a personas o grupos.
	No discriminación hacia áreas de iniciativa.
	Distribución de la licencia: deben aplicarse los mismos derechos a todo el que reciba el programa.

<sup>23</sup> El freeware significa literalmente “software gratis” y es un tipo de software que se distribuye sin costo y puede ser usado por tiempo ilimitado.

	La licencia no debe restringir otro software.
	La licencia debe ser tecnológicamente neutral.

**Tabla 4.1.** Tabla comparativa de filosofías FSF y OSI.

**Fuente:** Elaboración propia. 2015.

Con base en estos criterios podemos decir que una de las diferencias más importantes entre estos dos movimientos radica en sus filosofías ya que como podemos apreciar el Open Source se enfoca más en el aspecto técnico, por el contrario el Free Software tiene como prioridad el aspecto ético y moral del programa de computadora.

Otra de las diferencias importantes respecto al ámbito comercial es que cuando se desarrolla un software bajo la licencia y estándares del Free Software se le pueden obtener ingresos por conceptos de desarrollo, brindando soporte técnico e incluso en la implementación del producto si y solo si se entrega el código fuente al usuario final, lo que no sucede con aplicaciones desarrolladas bajo licencias de Open Source. Por lo tanto los productos de Free Software, sus modificaciones e incluso trabajos derivados del original deben ser libres (libertad), a diferencia del Open Source.

Por último y no menos importante cabe destacar que actualmente estos dos movimientos son aliados y no enemigos, ya que defienden y persiguen los mismos objetivos, aunque como ya vimos tengan diferencias filosóficas.

En el Apéndice A de esta tesis se puede consultar la Licencia de Software libre de PYSMG.

## 4.2 DISEÑO E IMPLEMENTACIÓN DE PYSMG

En esta sección se describirá la forma en que se diseñó el software y cómo se implementó. Se empieza por realizar un análisis de los requerimientos del software, es decir, las funciones que necesita realizar PYSMG, después una serie de diagramas (casos de uso, de paquetes, de clases, de actividades y de secuencia) que ayudaron a programar el sistema, por último se muestra la codificación de algunas clases que son cruciales para el correcto funcionamiento del sistema.

### 4.2.1 Análisis de requerimientos del software

Anteriormente ya se mencionó la necesidad de crear un software para la interpretación y la generación de gráficas con los datos que arroja el Monitor de Neutrones de la Ciudad de México, estos datos son vertidos en un archivo con las siguientes características:

- Datos en formato ASCII<sup>24</sup>.
- Archivo en formato de texto (\*.txt).
- Razón mínima de conteo de 1 minuto.
- Datos con razón de conteo de ~1350.
- Datos en tres columnas (fecha, hora, datos).

Con base en lo anterior, se necesita generar gráficas no sólo de los datos con la resolución mínima (1 minuto), sino a partir de la mínima obtener las siguientes resoluciones:

- Mínima (1 minuto).
- 5 minutos.
- 30 minutos.
- 1 hora.
- 1 día.

Por supuesto esto se obtiene sumando las cantidades del conteo de resolución mínima hasta obtener el resultado esperado, de esta manera se obtiene lo siguiente:

- Para la resolución de 1 minuto, se grafican todos los puntos.
- Para la resolución de 5 minutos, se suman intervalos de 5 datos.
- Para la resolución de 30 minutos, se suman intervalos de 30 datos.
- Para la resolución de 1 hora, se suman intervalos de 60 datos.
- Para la resolución de 1 día, se suman intervalos de 1440 datos.

---

<sup>24</sup> Código Estándar Estadounidense para el Intercambio de Información.

Con todo esto, para poder generar una gráfica se le solicita al usuario final la siguiente información:

- Archivo de datos valido.
- Fecha y hora de inicio.
- Fecha y hora de fin.
- Resolución.

Adicionalmente para generar la estadística básica se solicitará al usuario dos rangos de fechas para ser comparadas y arrojar los siguientes resultados:

- La media en datos.
- El mínimo y el máximo de cada rango.
- La media en porcentaje.

La media en porcentaje se obtiene al realizar un comparativo entre el primer rango de fechas introducido por el usuario y el segundo. En este caso también se podrá si así lo desea el usuario generar la gráfica de los intervalos que ingresó para el cálculo de la estadística y su respectiva resolución.

La presentación de los resultados también requiere de una presentación institucional y las correctas etiquetas a los ejes de la gráfica, a continuación se indican los datos en orden de aparición:

- Universidad (UNAM).
- Institución (Instituto de Geofísica).
- Detector de partículas (Monitor de Neutrones)
- Etiqueta con el rango de fechas a la que pertenece la gráfica.
- Eje Y: Resolución de la gráfica.
- Eje X: Tiempo local.

Finalmente, de ser necesario el usuario podrá exportar los datos utilizados para realizar la gráfica en un archivo de texto con formato idéntico del original, para que puedan ser utilizados nuevamente.

#### 4.2.2 Diagrama de casos de uso

De acuerdo con Gutiérrez (2011a), un diagrama de caso de uso es una descripción de un proceso fina-fin, relativamente largo, que incluye varias etapas o transacciones y en el que uno o varios actores interaccionan con el sistema.

Primero se empieza por definir los actores del sistema, en éste caso sólo se define uno:

**Usuario:** Es el actor cuyo papel es desarrollado por la persona que hará uso del software, es decir, cualquiera que desee generar graficas ya sea un estudiante, practicante o investigador del Instituto de Geofísica.

Ahora se continúa listando los posibles casos de uso del sistema, en esta ocasión se listan los más importantes:

**Generar gráficas de todas las fechas:** Se utiliza cuando el usuario quiere graficar todas las fechas (datos) contenidas en el archivo que ha seleccionado.

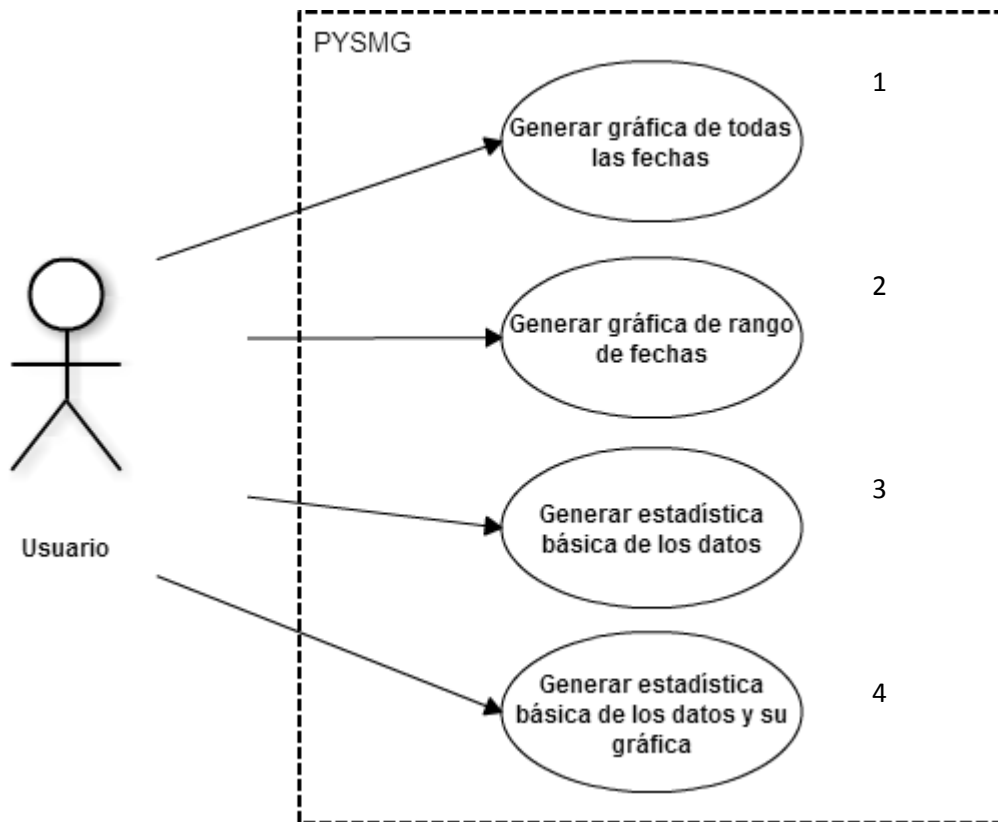
**Generar gráficas de rango de fechas:** Este caso de uso necesita que el usuario delimite e ingrese las fechas que quiere graficar, esto mediante la interfaz gráfica.

**Generar estadística básica de los datos:** Al igual que el anterior caso de uso, se requiere que el usuario ingrese dos rangos de fechas para que sean comparados por el sistema y muestre el resultado en una nueva ventana.

**Generar estadística básica de los datos y su gráfica:** Finalmente, aquí el usuario al ingresar los dos rangos de fechas solicitados por el sistema para ser comparados, permite generar su gráfica.

A continuación se muestra el diagrama de uso antes descrito:





**Figura 4.2.** Diagrama de casos de uso de PYSMG.

**Fuente:** Elaboración propia. 2015.

Para entender mejor este punto, mostraré la descripción de los casos 1 y 3 ya que los otros son similares:

#### Descripción del Caso de Uso 1

<b>Nombre:</b>	Generar gráfica de todas las fechas
<b>Autor:</b>	Oswaldo Trujillo
<b>Fecha:</b>	05/05/15
<b>Descripción:</b>	Permite generar una gráfica con todas las fechas contenidos en el archivo de datos ingresado por el usuario.
<b>Actores:</b>	Usuario
<b>Precondiciones:</b>	El usuario debió haber ejecutado PYSMG sin ningún error.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El actor selecciona el archivo de datos a graficar.</li> <li>2. El actor carga los datos al sistema.</li> <li>3. El sistema lee e interpreta los datos del archivo y los muestra al actor.</li> <li>4. El actor quiere graficar todas las fechas.</li> <li>5. El actor ingresa la resolución que quiere para la gráfica.</li> <li>6. El actor presiona el botón graficar.</li> <li>7. El sistema muestra la gráfica solicitada.</li> </ol>

**Flujo Alternativo:**

1.A. El sistema evalúa el archivo de datos elegido por el usuario, si el formato del archivo no es correcto, avisa al actor para que elija uno válido.

6.A. El sistema evalúa los datos que el actor quiere graficar, si hay algún error avisa al actor para que los corrija.

**Poscondiciones:**

El sistema ha interpretado gráficamente los datos ingresados por el actor.

**Tabla 4.2.** Descripción del caso de uso para graficar todas las fechas.

**Fuente:** Elaboración propia. 2015.

### Descripción del Caso de Uso 3

<b>Nombre:</b>	Generar estadística básica de los datos
<b>Autor:</b>	Oswaldo Trujillo
<b>Fecha:</b>	05/05/15
<b>Descripción:</b>	Permite generar estadística básica de los datos ingresados por el usuario.
<b>Actores:</b>	Usuario
<b>Precondiciones:</b>	El usuario debió haber ejecutado PYSMG sin ningún error.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El actor selecciona el archivo de datos a graficar.</li> <li>2. El actor carga los datos al sistema.</li> <li>3. El sistema lee e interpreta los datos del archivo y los muestra al actor.</li> <li>4. El actor ingresa los dos rangos de fecha que quiere comparar.</li> <li>5. El actor presiona el botón calcular.</li> <li>6. El sistema muestra los cálculos generados con los datos que proporcionó el actor.</li> </ol>
<b>Flujo Alternativo:</b>	<p>1.A. El sistema evalúa el archivo de datos elegido por el usuario, si el formato del archivo no es correcto, avisa al actor para que elija uno válido.</p> <p>5.A. El sistema evalúa los rangos de fechas que desea comparar el usuario, si existe alguna inconsistencia con las fechas que ingresó el usuario, el sistema avisa para que este los corrija.</p>
<b>Poscondiciones:</b>	El sistema ha interpretado los datos y realizado los cálculos de estadística para mostrarlos al usuario.

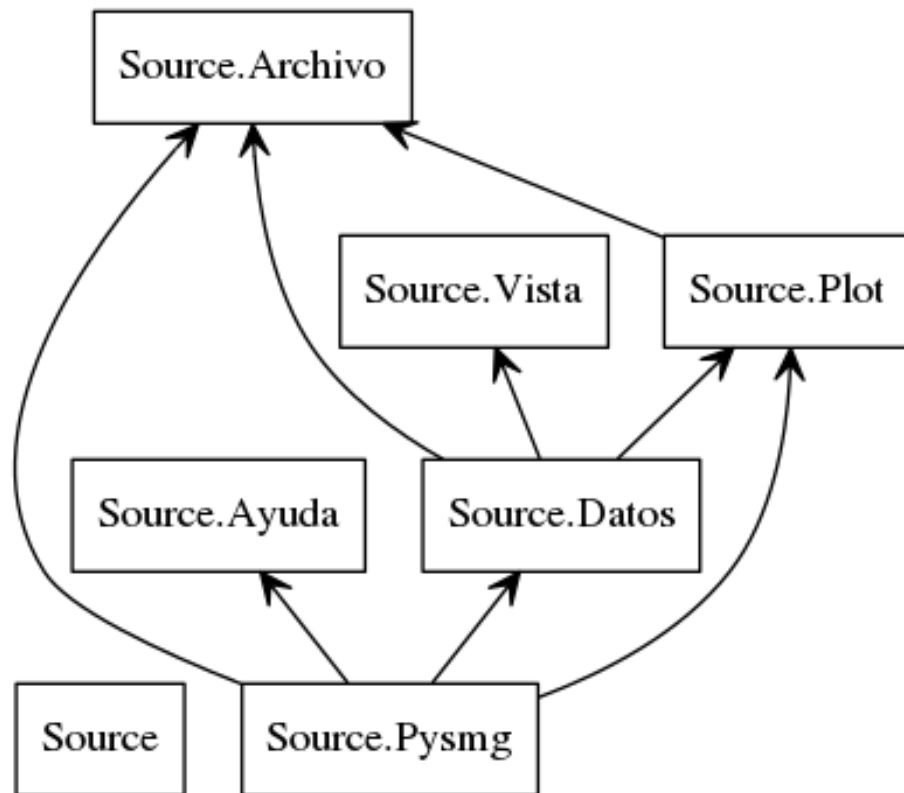
**Tabla 4.3.** Descripción del caso de uso para generar la estadística de los datos.

**Fuente:** Elaboración propia. 2015.

### 4.2.3 Diagrama de paquetes

En programación un paquete se refiere a mecanismo utilizado para agrupar elementos de UML<sup>25</sup> el cual permite agrupar los elementos modelados, facilitando de ésta forma el manejo de los modelos de un sistema complejo, (Gutiérrez, 2009).

El diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones, (Ambler, 2003). En este caso PYSMG solo cuenta con un paquete `Source` el cual contiene todos los archivos de código fuente, por lo que el diagrama de paquetes queda de la siguiente forma:



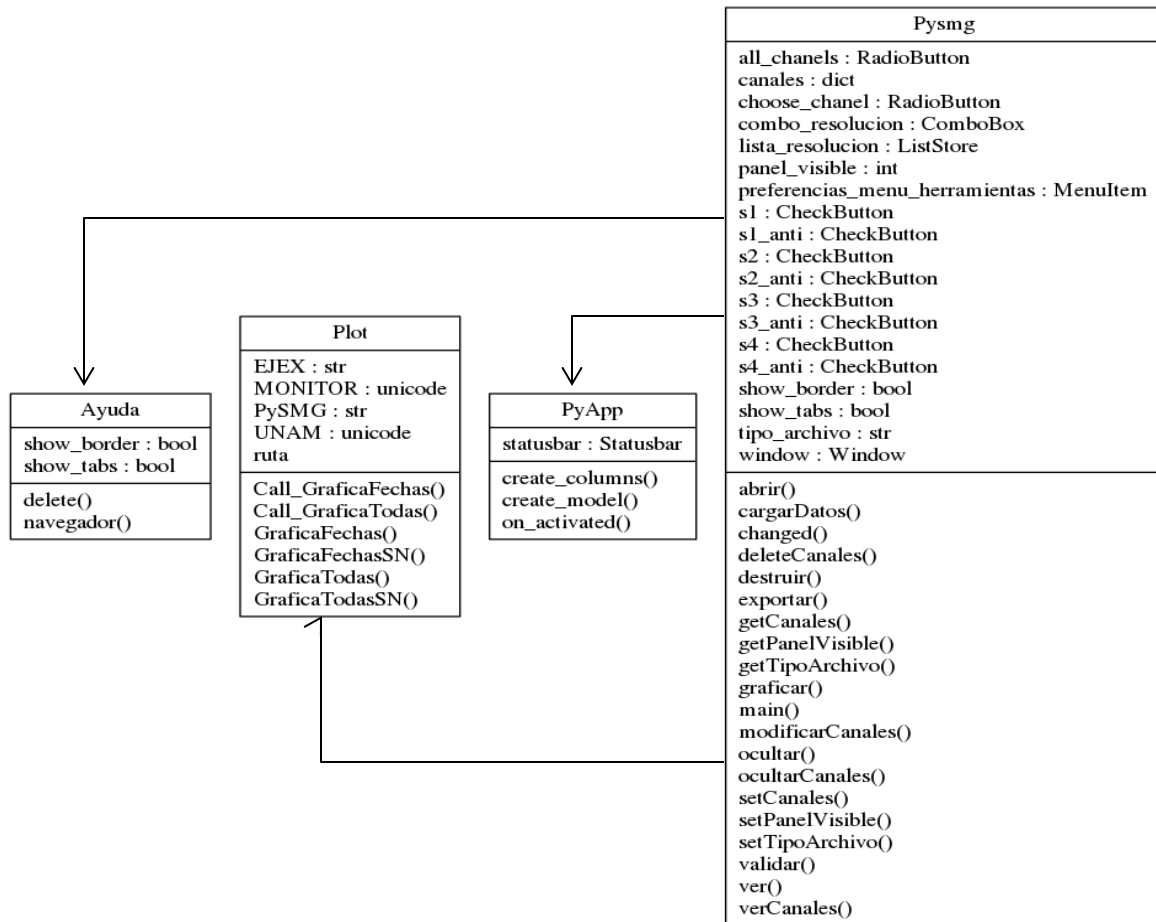
**Figura 4.3.** Diagrama de paquetes de PYSMG.

**Fuente:** Elaboración propia. 2015.

<sup>25</sup> Lenguaje Unificado de Modelado.

### 4.2.3 Diagrama de clases

En el capítulo 3 se definió y explicó lo que es una clase. El diagrama de clases sirve para observar y entender las relaciones entre las clases que conforman el sistema, la figura 4.x muestra el diagrama de clases de PYSMG.



**Figura 4.4.** Diagrama de clases de PYSMG.

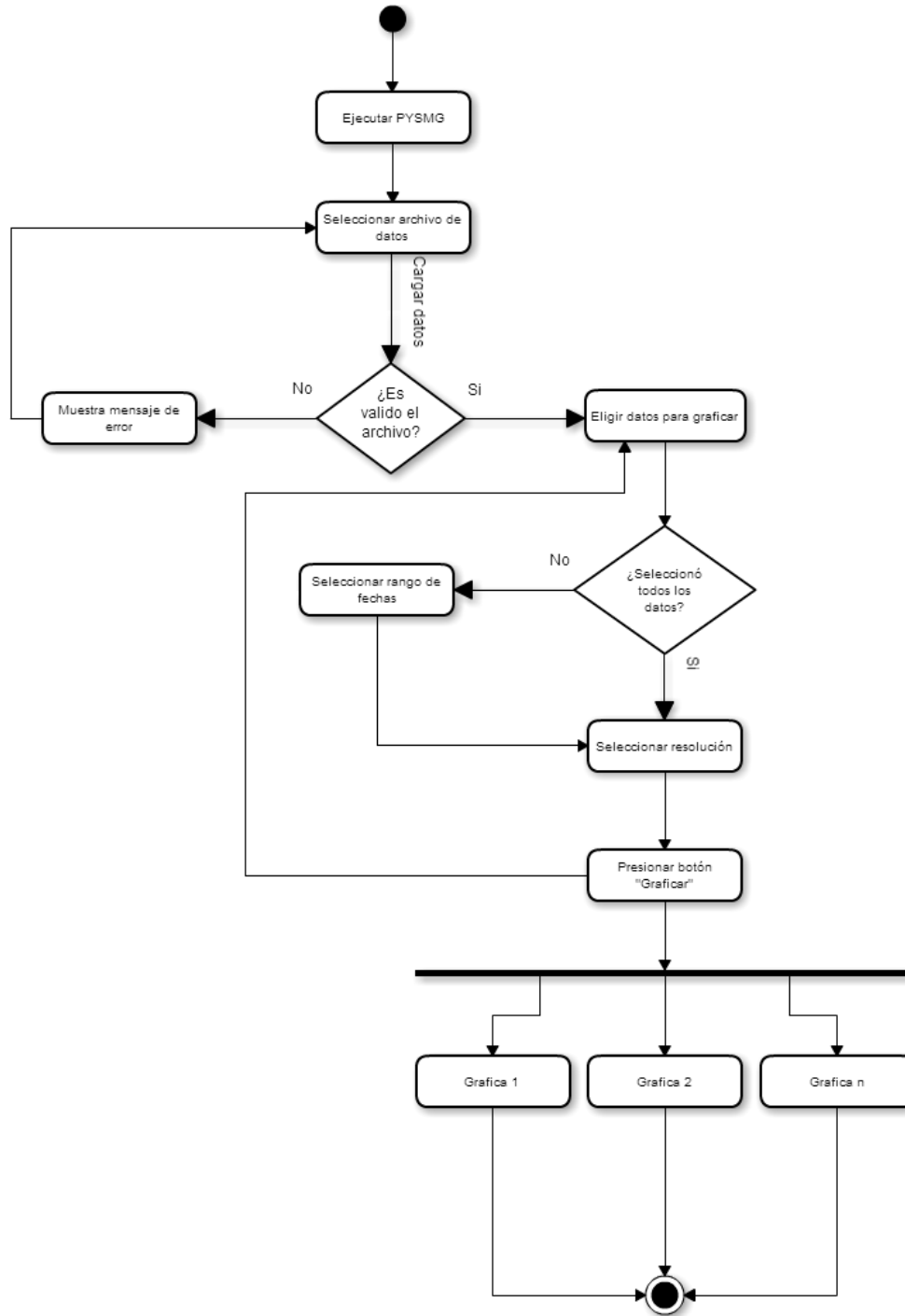
**Fuente:** Elaboración propia. 2015.

En el diagrama anterior muestra la relación que hay entre las clases principales de PYSMG. En cada clase (rectángulo) se muestra el título, en el renglón de abajo se describen los atributos de la clase y al final las operaciones o métodos de la clase.

Claramente se puede observar que la clase principal **Pysmg** interactúa con las demás clases haciendo la llamada correspondiente cuando el usuario así lo indica.

#### 4.2.4 Diagrama de actividades

Los diagramas de actividades muestran el flujo de trabajo desde el punto de inicio hasta el punto final, detallando muchas de las rutas de decisiones que existen en el progreso de eventos contenidos en la actividad.



**Figura 4.5.** Diagrama de actividades de PYSMG.

**Fuente:** Elaboración propia. 2015.

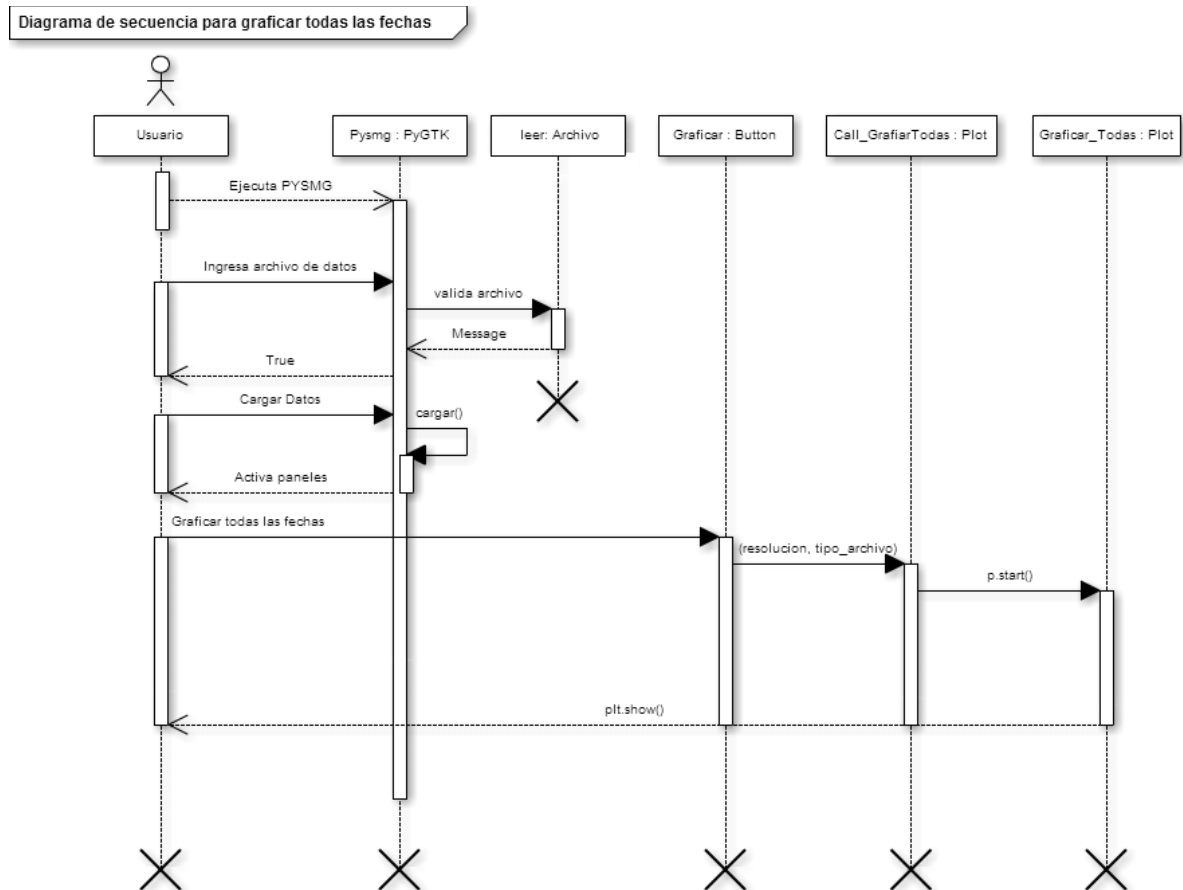
La figura 4.x muestra el diagrama de actividades (las más importantes) de PYSMG. Inicia con la ejecución del software, una vez mostrada la interfaz gráfica, el usuario ingresa el archivo de datos que quiere graficar, posteriormente se valida el archivo, si el archivo no contiene el formato de datos o la extensión no es la correcta, arroja un mensaje de error y se tendrá que elegir un archivo de datos valido.

De lo contrario, si el archivo de datos contiene el formato correcto, se cargan los datos al sistema para que el usuario pueda interactuar con ellos. El siguiente paso es elegir que datos se quiere graficar, si se eligió graficar un rango de fechas se continua con delimitar los datos.

A continuación se selecciona la resolución para la gráfica y se procede a presionar el botón “Graficar” para obtener la salida deseada, es importante mencionar que se pueden obtener las gráficas que sean debido a la concurrencia en este paso.

## 4.2.5 Diagrama de secuencia

Gutierrez (2011b) dice que los diagramas de secuencia muestran la forma en que un grupo de objetos se comunican (interactúan) entre sí a lo largo del tiempo, consta de objetos, mensajes entre estos objetos y una línea de vida del objeto representada por una línea vertical.



**Figura 4.6.** Diagrama de secuencia para graficar todas las fechas.

**Fuente:** Elaboración propia. 2015.

## 4.2.6 Codificación

Finalmente a continuación se mostrará la codificación de las clases y/o funciones críticas del sistema ya que debido al buen funcionamiento de éstas el software arrojará los resultados esperados por el usuario final. Se describe desde cómo se interpreta el archivo de datos, la obtención de las fechas del archivo hasta la forma en que se generan las gráficas solicitando al Sistema Operativo a través de Python la instanciación de un nuevo proceso.

#### 4.2.6.1 Lectura e interpretación del archivo de datos

La lógica de programación de PYSMG consiste en obtener los datos a graficar por medio de un archivo de datos que el usuario introduce al sistema, una vez que el usuario carga los datos, el sistema interpreta el contenido del archivo (el formato de los archivos de datos se describió en el capítulo 2) y vierte los datos en la interfaz gráfica para que estén disponibles para el usuario, a continuación se muestra el código de la función `setFecha` cuyo objetivo es obtener las fechas contenidas en el archivo:

```
1     def setFecha(ruta, tipo_archivo):
2         fechas = ["Seleccione una fecha"]
3
4         if tipo_archivo == "txt":
5             archivo = open(ruta, "r")
6             lineas = archivo.readlines()
7             archivo.close()
8             datos = lineas[6:]
9             indice = 0
10
11            for i in datos:
12                temp = i[0:10]
13                if fechas[indice] != temp:
14                    fechas.append(temp)
15                    indice += 1
16            return fechas
```

Se define un método llamado `setFecha` que recibe dos parámetros (1) `ruta` el cual contiene la ruta del archivo que el usuario introduce y (2) el tipo de archivo de datos. En la línea 2 se inicializa una lista `fechas` con el elemento inicial para mostrar al usuario, posteriormente se hace la validación del tipo de archivo evaluando el segundo parámetro en la línea 4.

En la línea 5 se abre el archivo para su lectura, en la línea 6 se define la lista `lineas` la cual contiene todo el contenido del archivo para que posteriormente en la línea 7 se cierre el archivo y se trabaje con variables locales.

Recordando el formato de los archivos de datos, en la línea 8 se eliminan las cabeceras y se define la lista `datos` con la información útil mediante `lineas[6:]`.

A partir de la línea 11 se lee línea por línea la lista `datos`, se almacena la cadena definida entre el carácter 0 a 10 la cual contiene la fecha de cada dato y se guarda



en la variable `temp`, después se compara el contenido del último elemento de la lista `fechas`, si es diferente del contenido de la variable `temp`, el contenido de esta variable se agrega a la lista `fechas`.

Finalmente la función regresa la lista con las fechas contenidas en el archivo.

Ya que se tienen las fechas hay que obtener los tiempos que contiene el archivo, en teoría se debería agregar cada minuto de las 24 horas del día, sin embargo, se hace la validación con la siguiente función:

```
1     def setHora(ruta, tipo_archivo):
2         horas = ['Seleccione una hora']
3         indice = 0
4
5         if tipo_archivo == "txt":
6             archivo = open(ruta, "r")
7             lineas = archivo.readlines()
8             archivo.close()
9
10            datos = lineas[6:]
11
12            for i in datos:
13                temp = i[11:16]
14                if horas[indice] != temp:
15                    horas.append(temp)
16                    indice += indice
17                    if temp == "23:59":
18                        break
19            return horas
```

La función `setHora` es prácticamente la misma que la anterior, con la diferencia que ésta obtiene la cadena comprendida entre el carácter 11 y 16 de la línea, una vez que se llega a 23:59 sale del ciclo y finalmente regresa la lista con las horas contenidas en el archivo.

Finalmente estas dos funciones arman los elementos correspondientes en la interfaz gráfica para que el usuario interactúe con ellos.

#### 4.2.6.2 Creando nuevos procesos

En el capítulo anterior se habló sobre la diferencia entre un thread (hilo) y un proceso. También se describen las diferentes bibliotecas gráficas en Python, esto es importante ya que particularmente la integración de PyGTK y los threads se vuelve inestable y produce algunos errores e interrupciones inesperadas.

Debido a esto se decide utilizar el API de Python Multiprocessing la cual como ya se comentó anteriormente brinda la posibilidad al programador de generar nuevos procesos con la ventaja de incluir por default la interacción entre ellos.

De acuerdo con lo anterior, a continuación se muestra el código de la función en la que se implementa el Multiprocessing y manda llamar a otra función en la cual propiamente se genera la gráfica:

```
1 def Call_GraficaTodas(self, resolucion, tipo_archivo):
2     if tipo_archivo == "txt":
3         p = Process(target=self.GraficaTodas,
4                     args=(self.ruta, resolucion))
5         p.start()
```

La función `Call_GraficaTodas` es muy sencilla, primeramente esta función se encarga de generar un proceso nuevo `p` el cual recibe como parámetros (1) la función objetivo y (2) los parámetros (si así se requiere) de esa función.

En este caso llama a la función que se encarga de graficar todas las fechas contenidas en el archivo de datos, finalmente se inicia el proceso nuevo mediante la llamada al método: `p.start()`.

Para el caso de querer graficar un rango de fechas contenidas en el archivo, la función es prácticamente la misma:

```
1     def Call_GraficaFechas(self, FechaDe, FechaHasta, resolucion,
2                             tipo_archivo):
3         if tipo_archivo == "txt":
4             p = Process(target=self.GraficaFechas, args=(self.ruta,
5                                                         FechaDe, FechaHasta, resolucion))
6             p.start()
```

Esta función recibe dos parámetros más (1) `FechaDe` que es la fecha de inicio y la (2) `FechaHasta` que es la fecha de fin del rango a graficar, su funcionamiento es igual al anterior al generar un nuevo proceso y llamar a la función para graficar un rango de fechas.

#### 4.2.6.3 Generando las gráficas

Ahora que ya se generó el nuevo proceso, se identificó el conjunto de instrucciones que se ejecutarán y se han pasado los parámetros correspondientes, solo queda realizar la gráfica con las directrices antes mencionadas.

Antes de mostrar el código, cabe mencionar que para este caso se hace uso de una de las librerías más poderosas en Python, Numpy es el paquete fundamental para el cómputo científico. Numpy cuenta a su vez con varios componentes, uno de ellos es Matplotlib que es una biblioteca para la generación de gráficos a partir de datos contenidos en listas (o arrays), (The Matplotlib Development Team, 2002). Para hacer uso de este recurso, se debe importar de la siguiente forma:

```
1 import matplotlib.pyplot as plt
2 import matplotlib.dates as mdates
```

Para realizar la gráfica de todos los datos contenidos en el archivo el código es el siguiente:

```
1 def GraficaTodas(self, *args):
2     path = args[0]
3     resolucion = args[1]
4
5     if((resolucion==1)or(resolucion==5)
6         or(resolucion==30)):
7         EJEY = "Cuentas / "+ str(resolucion) + " min"
8     elif(resolucion == 60):
9         EJEY = "Cuentas / 1 hora"
10    else:
11        EJEY = u"Cuentas / 1 día"
12
13    datos = leer(path)
14    x = []
15    y = []
16    fechas = []
17    contador = 0
18    lista_fechas = []
19    lista_horas = []
20    lista_cuentas = []
21
22    for dato in datos:
23        lista_fechas.append(dato[0:10])
24        lista_horas.append(dato[11:16])
```

```

24         lista_cuentas.append(int(dato[20:]))
25
26     while contador < len(datos):
27         i = 0
28         cuentas = 0
29         while i < resolucion:
30             if contador < len(datos):
31                 cuentas += lista_cuentas[contador]
32                 i += 1
33                 contador += 1
34             else:
35                 break
36         y.append(cuentas)
37         tmp=lista_fechas[contador-1]+
38             lista_horas[contador-1]
39         fecha = datetime.datetime.strptime(tmp, "%Y-
40             %m-%d%H:%M")
41         x.append(mdates.date2num(fecha))
42         fechas.append(tmp)
43
44     strFechas = str(fechas[0][0:10])+" a "+str(fechas[-
45         1][0:10])
46     plt.figure(self.PySMG)
47     plt.xticks(size = 'small', rotation = 25)
48     plt.grid(which = 'major', axis = 'both')
49     plt.xlabel(self.EJEX, size = 22)
50     plt.ylabel(EJEY, size = 22)
51     plt.suptitle(self.UNAM, size = 15)
52     plt.title(self.MONITOR, size = 12)
53     plt.plot_date(x, y, ",", label = strFechas, xdate =
54         True, linestyle = "-",
55         color = "g", drawstyle = "steps")
56     plt.minorticks_on()
57     plt.show()

```

Como se mencionó, el código de arriba es el conjunto de instrucciones a ejecutar en el proceso creado anteriormente. La función recibe un arreglo de parámetros, los cuales en la línea 2 y 3 se asignan a las variables locales correspondientes. De la línea 5 a la 10 se define la etiqueta para el eje Y dependiendo de la resolución que haya elegido el usuario. En la línea 12 a la 19 se definen las variables que se ocuparan el transcurso de la función. Se continúa separando cada columna del

archivo en tres listas diferentes (una para la fecha, hora y cuentas) a través de un `for` en la línea 21. En la línea 26 se empieza a organizar los datos y fechas a graficar con base en la resolución que eligió el usuario, se forman las listas `x` y `y` que son los datos de cada eje correspondiente. La parte más importante de la función es generar la gráfica, en la línea 45 se encuentra la instrucción para generar una nueva ventana:

```
45 plt.figure(self.PySMG)
```

La cual recibe como parámetro el título de la misma, se inicializan las etiquetas de la gráfica (título, subtítulo, eje x, eje y y anotaciones). En la línea 52 se da la instrucción de dibujar la gráfica en la ventana anteriormente creada por medio de:

```
52 plt.plot_date(x, y, ",", label = strFechas, xdate =
    True, linestyle = "-", color = "g", drawstyle
    = "steps")
```

Se pasan como parametros los datos que se graficarán (`x`, `y`), su etiqueta, se indica que el eje x es de formato fecha, el estilo de la línea, el color y finalmente el estilo de la gráfica que en este caso es por pasos. Finalmente se muestra la ventana con la gráfica con `plt.show()`.

El procedimiento para graficar un rango de fechas es prácticamente el mismo:

```
1 def GraficaFechas(self,*args):
2     fechade = args[1]
3     fechahasta = args[2]
4     .
5     .
6     .
10    datos = leer(path)
11    indice_FechaDe = 0
12    indice_FechaHasta = 0
13    x = []
14    y = []
15    fechas = []
16
17    for elemento in datos:
18        if fechade in elemento:
19            break
20        indice_FechaDe = indice_FechaDe + 1
```

```

21
22     for elemento in datos:
23         if fechahasta in elemento:
24             break
25         indice_FechaHasta = indice_FechaHasta + 1
26
27     lista=datos[indice_FechaDe:(
    indice_FechaHasta + 1)]
    .
    .
    .

```

La diferencia es que la función GraficaFechas recibe dos parámetros más (1) fecha de inicio y (2) la fecha de fin. Antes de realizar las funciones del código anterior, primero busca en la lista de datos completos la fecha de inicio y si lo encuentra guarda el índice en la variable indice\_FechaDe, después busca la fecha de fin en la lista y si lo encuentra nuevamente guarda el índice ahora en indice\_FechaHasta. De esta manera en la línea 27 se genera una nueva lista con los datos delimitados por las fechas que indicó el usuario. Finalmente continua con las mismas funciones que el código anterior.

## 4.3 USO DE PYSMG

Dentro de esta sección se verá el uso de PySMG y se verán los requerimientos técnicos necesarios para su ejecución. También se verá una pequeña introducción de la adquisición de datos con los que trabaja PySMG.

En el capítulo anterior se describió el lenguaje de programación Python, específicamente en su versión 2.7, el cual se eligió para el diseño y desarrollo de PySMG debido a su versatilidad multiplataforma, su potencia y su facilidad de aprendizaje.

En este capítulo se empezó por describir las dos grandes corrientes del Software Libre y de Código Abierto (FOSS por sus siglas en inglés). En este caso se decidió que PySMG contara con una Licencia de Free Software (ver apéndice A) ya que nuestros objetivos coinciden mejor con esta corriente.

También continuamos con la descripción del paradigma de la POO el cual se utilizó en el diseño y desarrollo de PySMG debido a la reusabilidad de código el cual nos permite evitar el repetir fragmentos de código, también nos brinda un mejor mantenimiento ya que es mucho más fácil de leer y como consecuencia también es mucho más fácil la modificación del código pensando a futuro.

Seguimos con las GUI, una parte muy importante del programa ya que es el medio por el cual el usuario interactúa con PySMG. En este caso se eligió pyGTK, las ventajas ya están descritas en la sección anterior: apariencia, una lista completa de elementos gráficos, facilidad de implementación e instalación son los elementos por los que se eligió esta GUI.

Finalmente, el Multiprocesamiento fue la técnica elegida para la optimización de PySMG, ya que nos brinda una API con las características del Multihilo y además nos permite aprovechar al máximo los recursos de las computadoras multinúcleo (Hoy en día la mayoría de las computadoras lo son).

Antes de poder ver en que consiste el funcionamiento de PySMG explicaré un poco los datos con los que trabaja y la forma en que se obtienen.

### 4.3.1 Requerimientos del sistema

Sistemas basados en Windows

Hardware:

**Procesador:** 1 GHZ de 32 o 64 bits.

**Memoria:** 2 GB de RAM

**Almacenamiento:** 100 MB disponibles

Software:

**Python:** versión 2.7, disponible en: <http://www.python.org>

**Matplotlib:** versión más actual en: <http://www.matplotlib.org>

**Pycairo:** versión más actual en: <http://www.cairographics.org>

**Numpy:** versión más actual en: <http://www.numpy.org>

**PyGTK:** versión más actual en: <http://www.pygtk.org>

**PyGObject:** versión más actual en: <http://www.wiki.gnome.org>

**PyParsing:** versión más actual en: <http://pyparsing.wikispaces.com>

**Python-dateutil:** versión más actual en: <https://labix.org>

Sistemas basados en GNU Linux:

Hardware:

**Procesador:** 1 GHZ de 32 o 64 bits.

**Memoria:** 2 GB de RAM

**Almacenamiento:** 100 MB disponibles

Software:

**Python:** versión 2.7

**Matplotlib:** instalar mediante:

Debian/Ubuntu:

```
sudo apt-get install python-matplotlib
```

Fedora/RedHat:

```
sudo yum install python-matplotlib
```

NOTA: La mayoría de los sistemas basados en GNU Linux tienen pre-instalado Python, de no ser así instalar mediante:



**Debian/Ubuntu:**

```
sudo apt-get install python2.7
```

**Fedora/RedHat:**

```
sudo yum install Python
```

### 4.3.2 Ejecutando PySMG

PySMG es el software desarrollado para analizar y graficar datos del Observatorio de rayos cósmicos de la Ciudad de México. A lo largo de ésta sección mostraré cada una de sus opciones para familiarizarnos con el funcionamiento del mismo.

Una vez que se hayan cumplido todas las dependencias y que Python 2.7 esté instalado en nuestra computadora, basta con hacer doble clic al archivo “PySMG.py” para que se ejecute el programa:

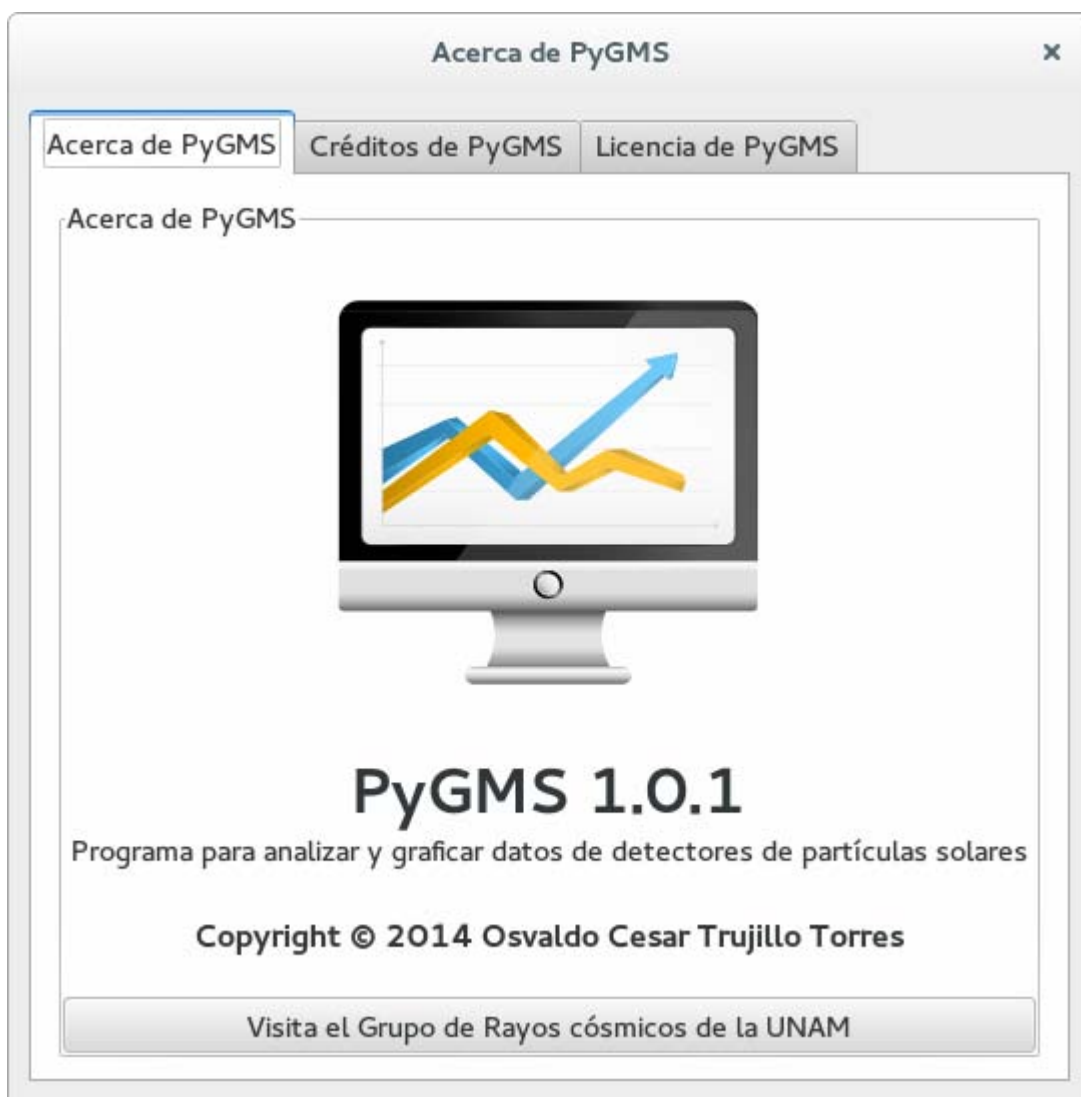
The screenshot shows the main interface of the PyGMS software. The window title is "PyGMS - Instituto de Geofísica". The interface is organized into several sections:

- Archivo de Datos:** Contains an "Examinar:" text input field, a "Cargar" button, and an "Examinar..." button.
- Fechas:** Features two radio buttons: "Graficar todas las fechas" (which is selected) and "Seleccionar fechas:". Below these are "De:" and "Hasta:" labels, each followed by a date selection widget consisting of a text box and a dropdown arrow.
- Resolución:** Includes a "Resolución:" label and a dropdown menu currently set to "5 minutos".
- Estadística:** Contains a "Rango:" section with "De:" and "Hasta:" date pickers, a "comparativo:" section with similar date pickers, a "Generar gráfica" checkbox, and a dropdown menu set to "Minima (1 minuto)". A "Calcular" button is located at the bottom right of this section.
- Bottom:** Features "Graficar" and "Salir" buttons.

**Figura 4.7.** Interfaz principal de PySMG.

**Fuente:** Elaboración propia. 2015.

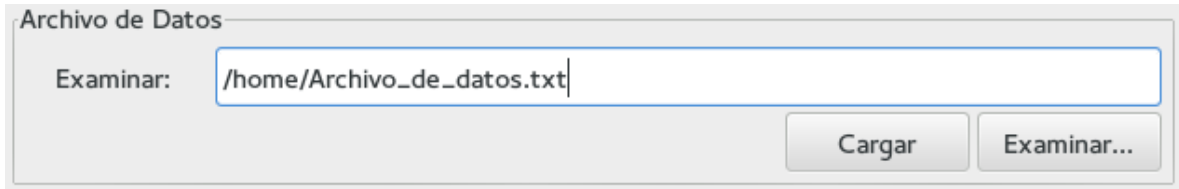
En la figura anterior se muestra la interfaz gráfica de PySMG, vamos a empezar a describir los menús: (1) Archivo, el cual cuenta con el submenú “Abrir...” para elegir el archivo de datos que vamos a graficar, el submenú “Exportar...” que sirve para guardar un rango determinado de datos del cuál realizamos alguna gráfica, por último el submenú “Salir”, obviamente cierra el programa. (2) Ayuda, solo cuenta con el submenú “Acerca de...” que contiene información de la versión del producto, créditos y la licencia del software.



**Figura 4.8.** Acerca de PySMG.

**Fuente:** Elaboración propia. 2015.

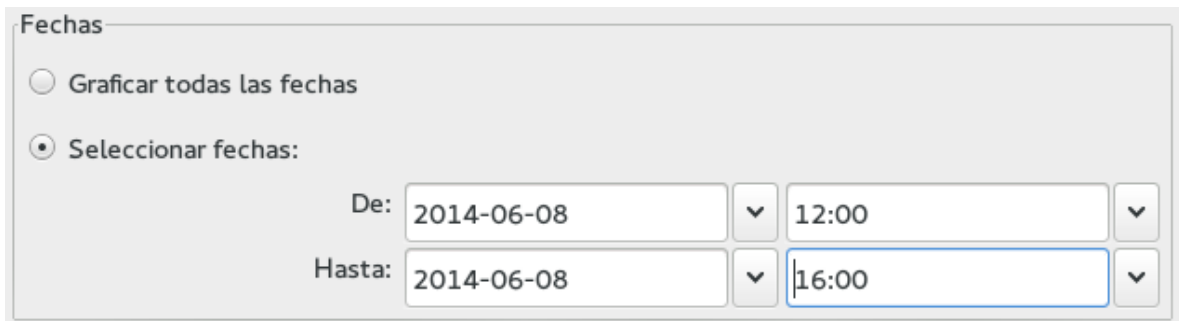
El primer panel activo es el de “Archivo de datos” mediante el cual escogemos el archivo del cual queremos realizar el análisis para que PySMG cargue los datos y los prepare para su graficación:



**Figura 4.9.** Panel para cargar el archivo de datos.  
**Fuente:** Elaboración propia. 2015.

Cuando PySMG valide el archivo y cargue correctamente los datos, automáticamente se activarán los cuatro paneles, iré describiendo uno por uno así como las funciones que realizan.

El siguiente panel disponible es el de “Fechas”:



**Figura 4.10.** Panel de fechas.  
**Fuente:** Elaboración propia. 2015.

En esta sección nos da la opción de “Graficar todas las fechas”, evidentemente si se selecciona graficará todos los datos contenidos en el archivo. La siguiente opción es “Seleccionar fechas:” que nos permite seleccionar un rango determinado de fechas contenidas en el archivo así como la hora exacta para generar nuestro análisis. En la figura 4.16 se muestra el ejemplo del 08 de junio del 2014 de las 12:00 hrs. hasta las 16:00 hrs.

Un aspecto muy importante para el funcionamiento y el análisis que realiza PySMG es la resolución de las gráficas, es decir, la cantidad de cuentas que se muestran en determinado rango de tiempo. Por ejemplo, continuando con el archivo de datos del día 08 de junio del 2014, tenemos cuentas de por minuto (~1350), sin embargo al momento de graficar requiero ver los datos con una resolución de 1 hora, la gráfica se mostrará con 24 puntos. PySMG cuenta con las siguientes resoluciones:

Mínima (1 minuto).

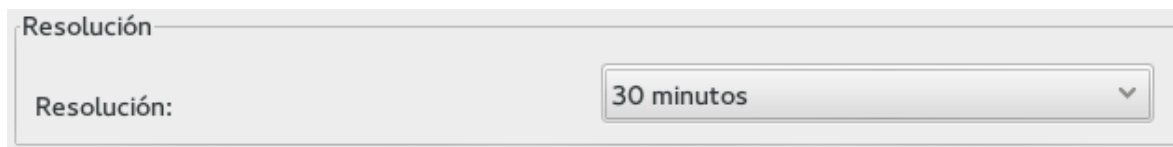
5 minutos.

30 minutos.

1 hora.

1 día

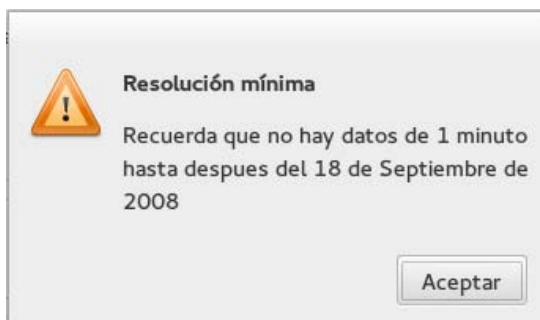
Esto permite tener mayor control sobre el resultado que queremos obtener y sobre todo brinda diferentes puntos de análisis de un mismo objetivo.



**Figura 4.11.** Panel para seleccionar la resolución para la gráfica.

**Fuente:** Elaboración propia. 2015.

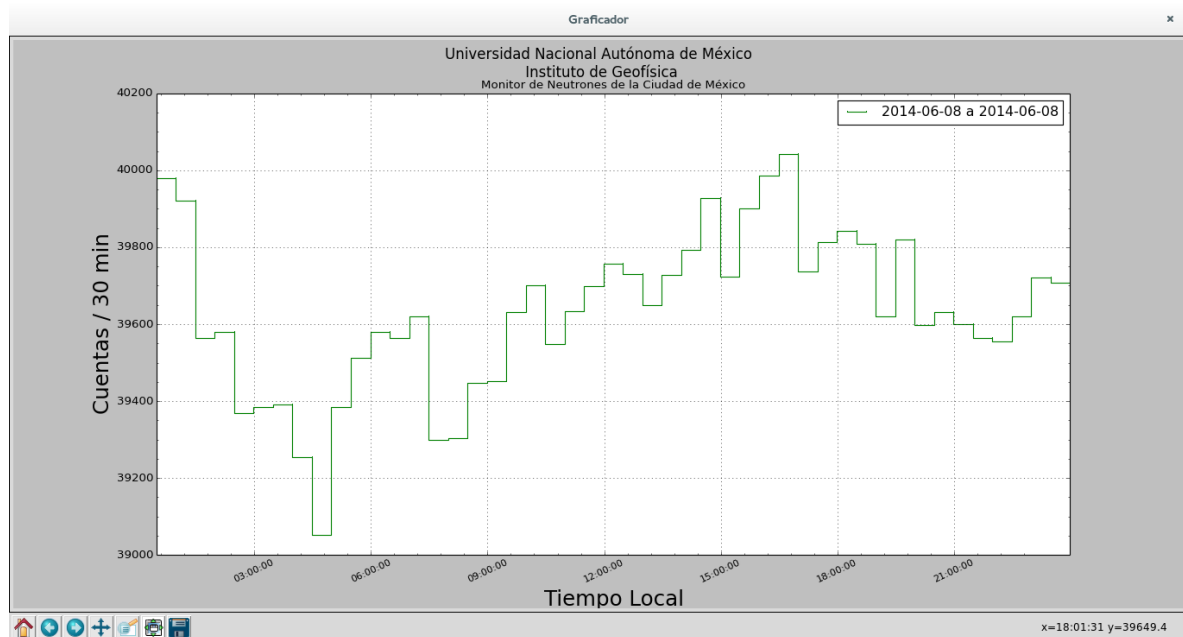
Cabe mencionar que el Observatorio de rayos cósmicos de la Ciudad de México empezó a recolectar datos con resolución de 1 minuto a partir del 18 de septiembre de 2008.



**Figura 4.12.** Alerta que indica la fecha donde se empiezan a obtener datos de 1 minuto.

**Fuente:** Elaboración propia. 2015.

Estos datos son suficientes para empezar a generar gráficas con PySMG. Se Habilita el botón “Graficar” y al presionarlo se muestra:



**Figura 4.13.** Gráfica hecha por PySMG.

**Fuente:** Elaboración propia. 2015.

Se genera una gráfica por pasos de cuentas/tiempo, en este caso con una resolución de 30 minutos.

PySMG da otra opción que es el panel de “Estadística” el cual nos permite realizar el cálculo de estadística básica para dos rangos de fechas, dando como resultado:

Rango de las fechas comparativas.

Media de los rangos de fechas.

Mínimos y Máximos de los rangos de fechas.

El porcentaje que representa la media de los rangos de fechas.

Estadística

Rango:  Generar gráfica

De: 2014-06-07 12:00 5 minutos

Hasta: 2014-06-07 20:59

comparativo:

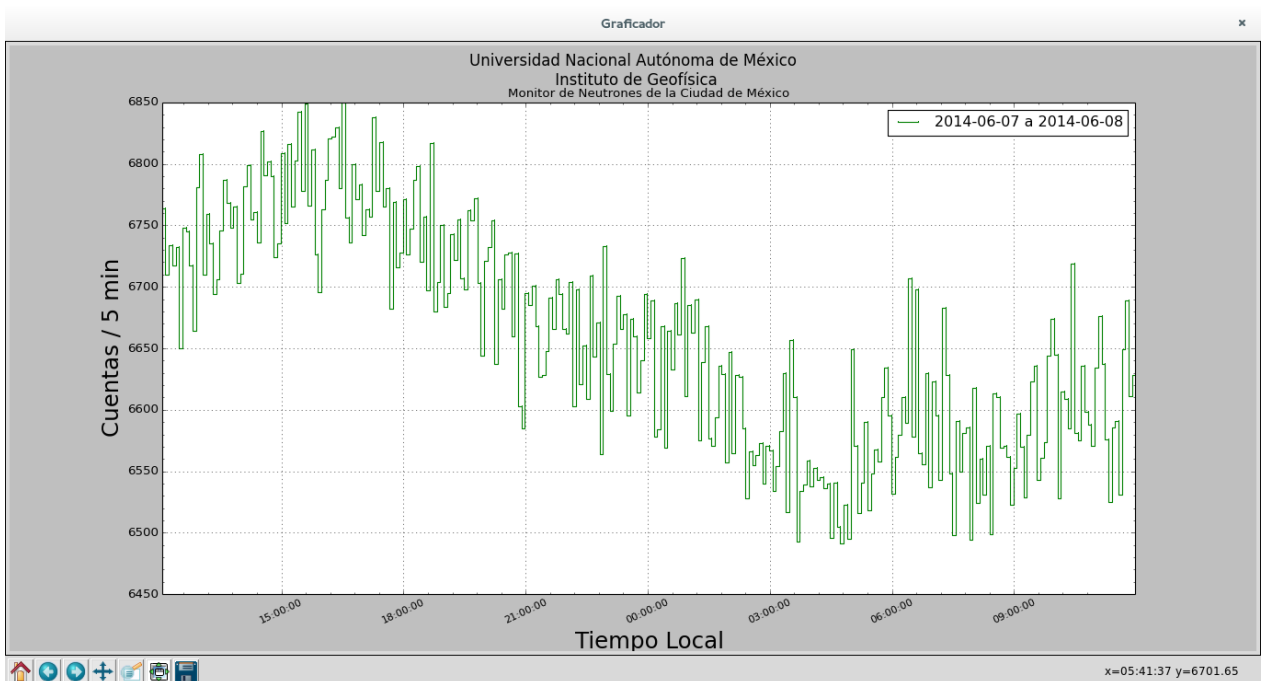
De: 2014-06-08 03:00

Hasta: 2014-06-08 11:59

Calcular

**Figura 4.14.** Panel para realizar cálculo estadístico.  
**Fuente:** Elaboración propia. 2015.

Primeramente el panel nos pide los rangos de fechas a comparar, posteriormente nos ofrece la posibilidad de generar la gráfica del comparativo de fechas y también la resolución, éste es el resultado:



**Figura 4.15.** Gráfica realizada con cálculo estadístico.  
**Fuente:** Elaboración propia. 2015.

Resultados				
	Rangos	Media	Min/Max	Media (%)
1	2014-06-07 a 2014-06-07 12:00 a 20:59	6747.81	6585/6850	100%
2	2014-06-08 a 2014-06-08 03:00 a 11:59	6578.36	6491/6719	-2.51%

**Figura 4.16.** Ventana con los cálculos realizados.

**Fuente:** Elaboración propia. 2015.

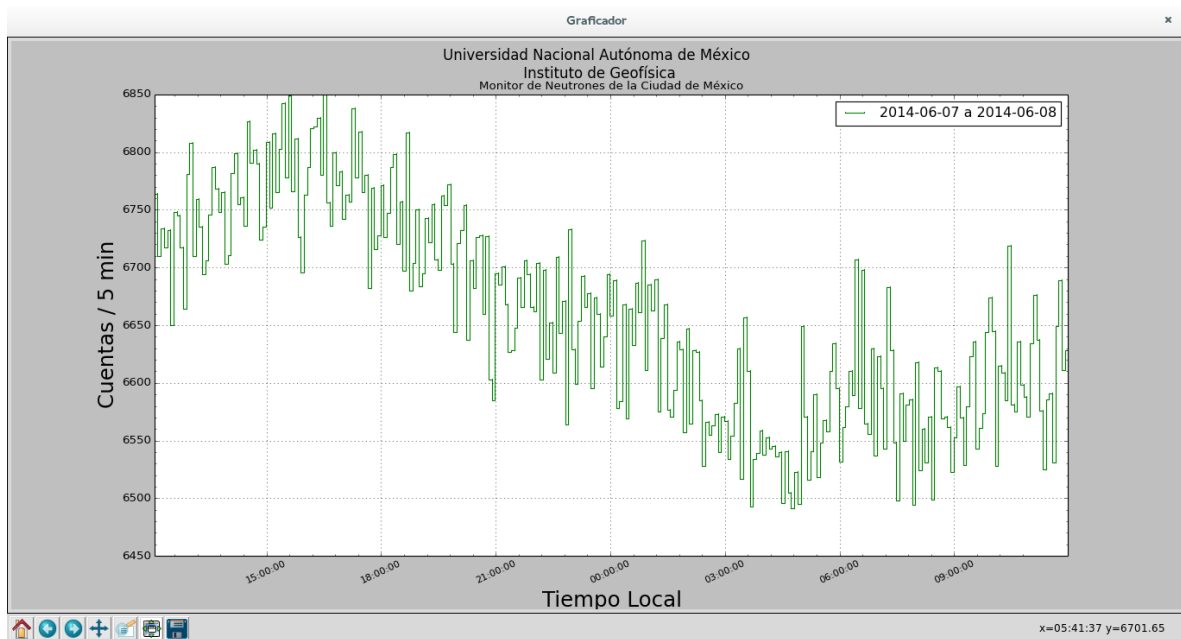
La figura 4.22 muestra los cálculos realizados, en el número 1 muestra el primer rango de fechas introducido, la segunda columna arroja la media (en cuentas/min), la tercera columna muestra el punto mínimo y el punto máximo del rango de fechas y por último la media en porcentaje. Para el número 2 es lo mismo a excepción de la media en porcentaje, ya que en este caso es comparado con el primer rango de fechas introducido, por lo que el segundo rango de fechas está 2.51 % por debajo de la media del primer rango.

En el Apéndice B de esta tesis se puede consultar el Manual de Usuario completo de PYSMG.

En el siguiente capítulo veremos más a fondo el análisis que se lleva a cabo con PySMG.



# CAPÍTULO 5: ANÁLISIS DE RESULTADOS



## 5.1 INTRODUCCIÓN

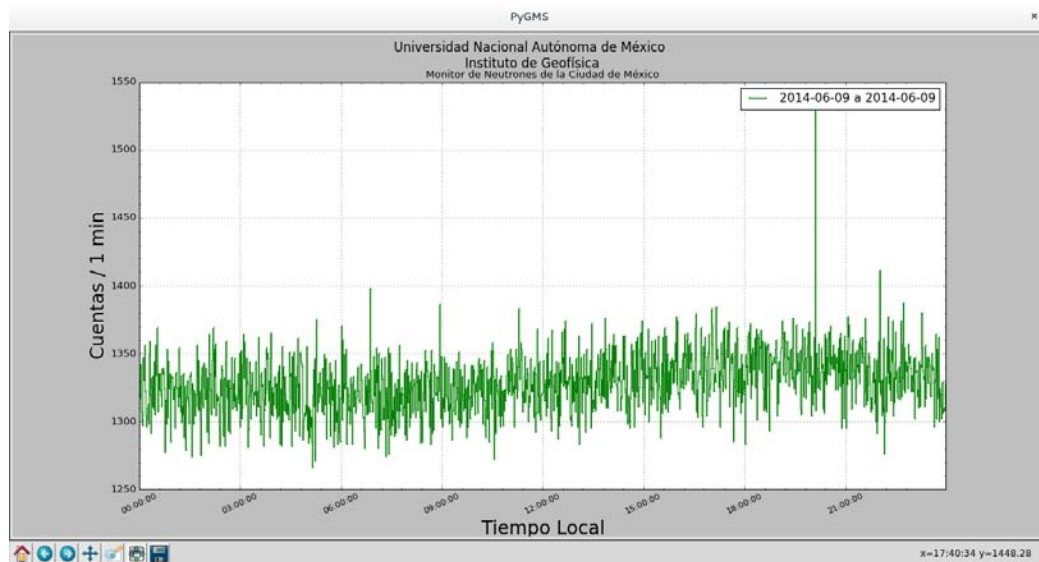
Cómo ya se comentó, el Monitor de Neutrones de Ciudad Universitaria detecta las partículas solares que llegan en todo momento a nuestro planeta. Arroja datos de ~1350 cuentas por minuto.

En el capítulo anterior se mostraron todas las opciones de PySMG así como sus respectivas ventanas, también de manera general se describió el funcionamiento de PySMG, se realizaron a manera de ejemplo algunas gráficas e incluso se mostró la comparación de dos rangos de fechas para realizar el cálculo de estadística.

En este capítulo describiré más a fondo la generación de graficas con PySMG. Se describe la forma de realizar estas gráficas con diferente rate (resolución) para poder identificar algunas variaciones al detectar neutrones.

## 5.2 GRÁFICAS CON RATE DE 1 MINUTO Y 5 MINUTOS

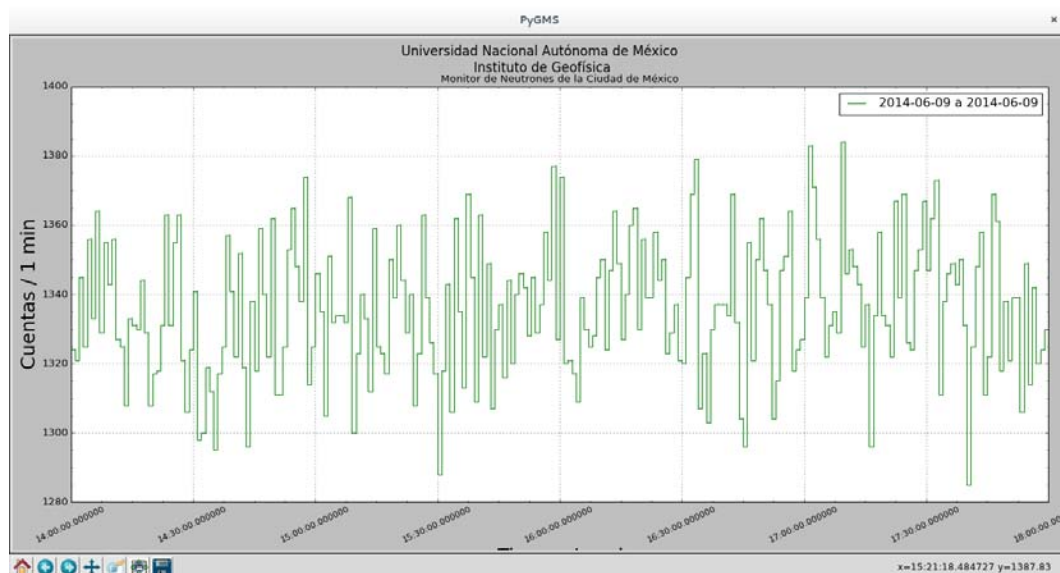
A continuación se muestra la gráfica del día 09 de junio de 2014 para ejemplificar la resolución de 1 minuto:



**Figura 5.1.** Gráfica con rate de 1 minuto para el día 9 de junio de 2014.

**Fuente:** Elaboración propia. 2015.

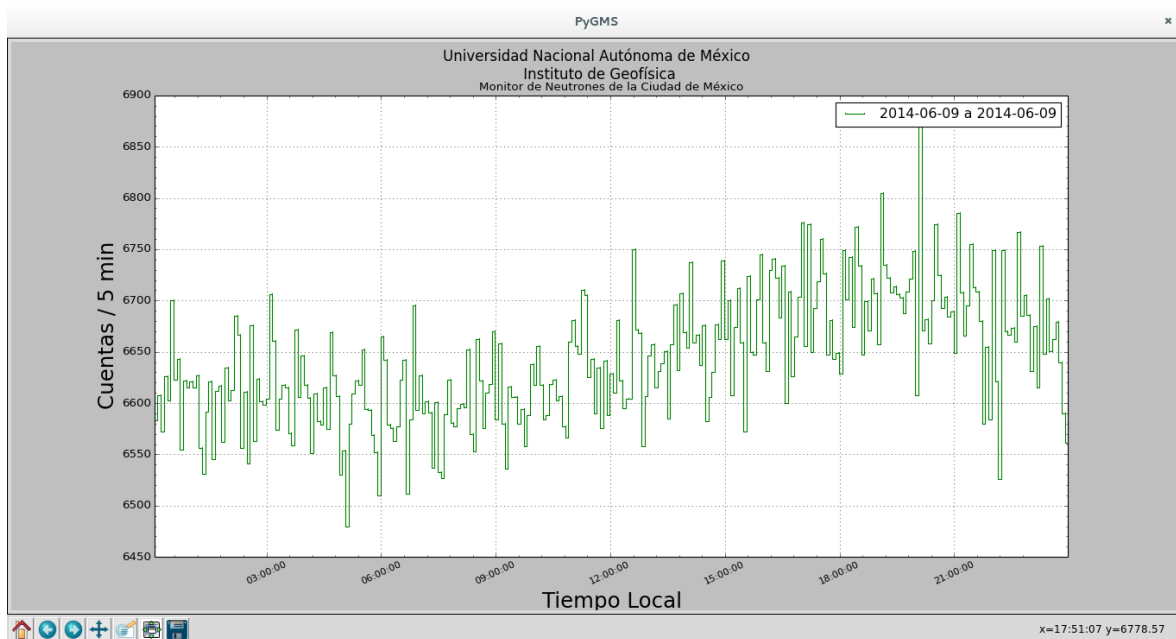
Cómo se observa en la figura 5.1, se grafican los 1440 minutos del día, sin embargo, esta resolución está pensada para visualizar intervalos de tiempo más cortos como se muestra en la figura 5.2.



**Figura 5.2.** Gráfica con rate de 1 minuto para el intervalo de las 14:00 a las 18:00 del 9 de junio de 2014.

**Fuente:** Elaboración propia. 2015.

Continuando con el mismo ejemplo, la figura 5.3 muestra la variación diurna<sup>26</sup> con rate de 5 minutos.



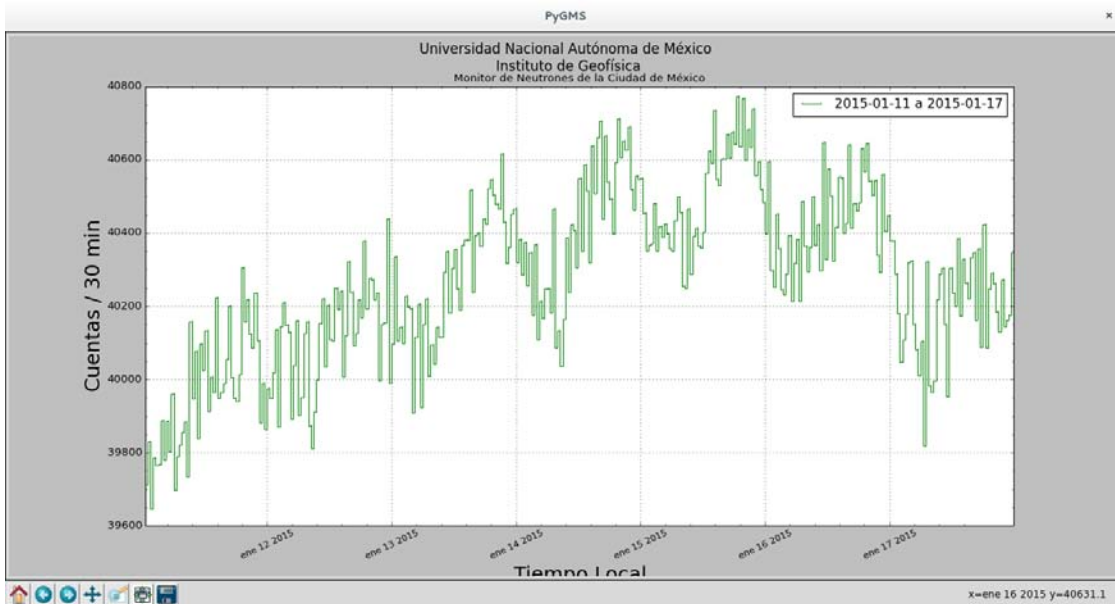
**Figura 5.3.** Gráfica con rate de 5 minutos del día 9 de junio, donde se puede visualizar la variación diurna (en el intervalo de las 12:00 a las 21:00), típica en este tipo de gráficas.

**Fuente:** Elaboración propia. 2015.

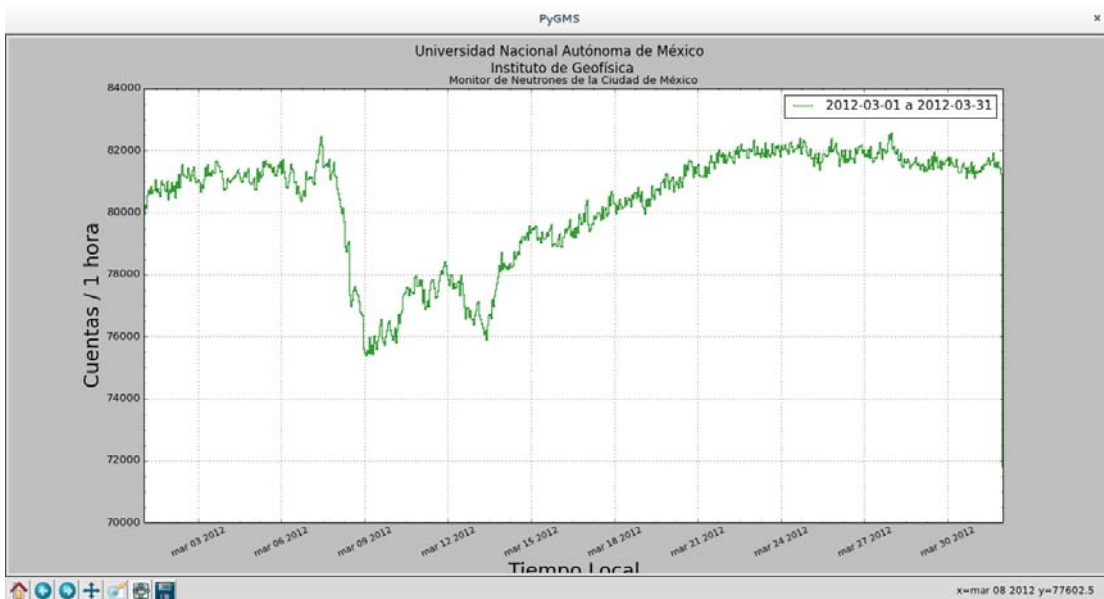
<sup>26</sup> Se define como el cambio de temperatura entre el día y la noche, producido por la rotación de la Tierra. Durante el día la radiación solar es en general mayor que la terrestre, por lo tanto la superficie de la Tierra se torna más caliente. Durante la noche, en ausencia de la radiación solar, sólo actúa la radiación terrestre, y consecuentemente, la superficie se enfría. Dicho enfriamiento continúa hasta la salida del sol. Por lo tanto la temperatura mínima ocurre generalmente poco antes de la salida del sol.

### 5.3 GRÁFICAS CON RATE DE 30 MINUTOS Y 1 HORA

Continuando con los resultados, la figura 5.4 muestra la gráfica con rate de 30 minutos para la semana del 11 al 17 de enero de 2015.



**Figura 5.4.** Gráfica de la semana del 11 al 17 de enero de 2015 con rate de 30 minutos.  
**Fuente:** Elaboración propia. 2015.



**Figura 5.5.** Gráfica del mes de marzo de 2012, con rate de 1 hora.  
**Fuente:** Elaboración propia. 2015.

Resultados				
	Rangos	Media	Min/Max	Media (%)
1	2012-03-01 a 2012-03-06 00:00 a 23:59	80996.88	74352/81684	100%
2	2012-03-10 a 2012-03-14 00:00 a 23:59	77539.46	75843/79531	-4.27%

**Figura 5.6.** Muestra el cálculo de la estadística para la figura 5.5.

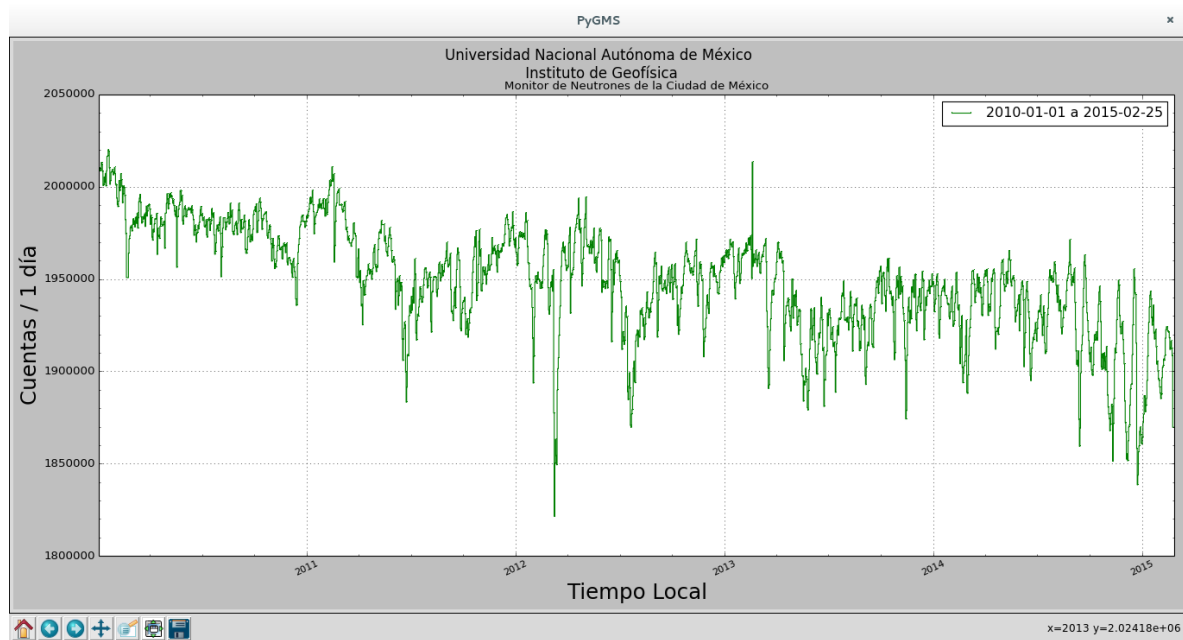
**Fuente:** Elaboración propia. 2015.

En la figura 5.5 podemos observar la gráfica para el mes de marzo de 2012 con rate de 1 hora. La gráfica muestra un comportamiento “diferente” a partir del 8 de marzo y se empieza a “normalizar” hasta el 21 de marzo. La figura 5.6 muestra el cálculo de la estadística del intervalo del 1 al 6 de marzo (1) comparado con el intervalo del 10 al 14 de marzo (2), donde podemos observar que el segundo intervalo se encuentra 4.27 % por debajo de la media del primer intervalo. A este efecto ocurrido en marzo del 2012 se le conoce como Decrecimiento Forbush<sup>27</sup> el cual es uno de los efectos más importantes que se puede visualizar con la detección de neutrones con el 6-NM64.

<sup>27</sup> Es la disminución temporal del número de rayos cósmicos galácticos que llegan a la Tierra; también conocida como el efecto de Forbush. Ocurre cuando una perturbación interplanetaria, como una CME (eyección de masa coronal) y precedida por una onda de choque, viaja hacia el exterior. Como resultado, existe un aumento en la intensidad del campo magnético interplanetario y en la densidad del viento solar, que difunde los rayos cósmicos incidentes lejos de la Tierra. El efecto se llama así en honor del geofísico norteamericano Scott Ellsworth Forbush.

## 5.4 GRÁFICAS DE 1 DÍA

Otro aspecto importante es el análisis para los ciclos solares los cuales duran aproximadamente 11 años. Para ejemplificar esto, en la figura 5.7 muestra la gráfica con rate de 1 día para el periodo de 2010-2015.



**Figura 5.7.** Gráfica que muestra la transición del monitoreo de neutrones del periodo comprendido entre el 1 de enero de 2010 hasta el 25 de febrero de 2015, con rate de 1 día.

**Fuente:** Elaboración propia. 2015.

# **CONCLUSIONES**



Sin duda hay avances tecnológicos en la historia de la humanidad que resultan más evidentes que otros y que tienen su sitio en los grandes libros de historia. Sin embargo, hay otros que son más parecidos a pequeños engranajes que impulsan dichos avances tecnológicos. PySMG es uno de ellos.

Todo lo expuesto anteriormente permite confirmar la hipótesis la cual se refiere a la posibilidad de desarrollar un software que permita la interpretación gráfica de los datos del Monitor de Neutrones de la Ciudad de México a partir de herramientas de propósito general, obteniendo de esta manera un software de propósito específico.

PySMG es un proyecto multidisciplinario el cual me permitió adquirir conocimientos en diferentes áreas de investigación. Me permitió abundar en áreas como la Física y la Astronomía, así como en áreas propias de la computación como el desarrollo de software, programación concurrente y la medición e instrumentación.

Para contextualizar el proyecto realice una pequeña investigación de la estructura básica del Sol. Un tema completamente nuevo para mí. Hice una pequeña introducción sobre su estructura interna y de su atmósfera. También describo brevemente la actividad solar, ya que es una parte muy importante sobre este trabajo.

Aprendí la forma en la que trabaja el Monitor de Neutrones de Ciudad Universitaria, el cual forma parte de una red mundial de monitores de neutrones. De los resultados arrojados por este monitor de neutrones depende mi proyecto, ya que estos datos son los que PySMG interpreta y grafica.

Para poder dar solución a la necesidad de interpretar estos datos, hice uso de tecnologías y metodologías adquiridas durante la ingeniería. Apliqué técnicas avanzadas de programación como lo es la Programación Orientada a Objetos, que si bien es cierto que es uno de los paradigmas más popular en los últimos años, no es fácil aplicarlo a todas las soluciones. También la parte visual es importante ya que es la forma en la que el usuario interactúa con mi aplicación, para esto hice

uso de las GUI. Otro aspecto fundamental para el correcto funcionamiento de PySMG radica en la programación concurrente. La solución a este problema para cualquier persona que tenga conocimiento de programación serían los Threads, sin embargo, Python nos brinda de herramientas muy potentes que a la hora de la implementación puede resultar un poco diferente a lo habitual.

No puedo dejar a un lado el lenguaje de programación. Hasta antes de desarrollar este proyecto no tenía conocimientos de Python. Python es un lenguaje de alto nivel con una sintaxis muy clara y sobre todo es un lenguaje potente. El aprendizaje de un nuevo lenguaje de programación representa para mí un crecimiento profesional importante.

En resumen, con este proyecto fui capaz de desarrollar e implementar nuevas tecnologías, colaborando con el trabajo realizado por los investigadores del Instituto de Geofísica de la UNAM.

Considero que la programación de este sistema me representó un reto profesional de alto nivel, ya que me enfrenté a problemas atípicos de los que comúnmente nos encontramos en los “proyectos” en la universidad, como lo es la implementación de Procesos en vez de utilizar Threads, debido a las inconveniencias que presenta la librería gráfica de Python que utilicé en el software y que, a pesar de que la Ingeniería de Software es parte de nuestra formación académica, no es sencillo aplicarla a problemas de la vida real.

Finalmente me dirijo a mis compañeros de Ingeniería en Computación, haciendo una invitación para participar en este tipo de proyectos que benefician la investigación no solo en la UNAM si no en todo el país, tratando de ésta manera de retribuir a nuestra máxima casa de estudios la oportunidad de contar con una educación pública, laica y gratuita.

## **TRABAJO A FUTURO**

Como ya se mencionó, el Monitor de Neutrones de Ciudad Universitaria no permite conocer la energía primaria de las partículas, ni la dirección en la que llegan, por lo que es necesario el uso de nuevos instrumentos para la observación de neutrones solares para conocer el espectro y dirección de arribo.

Para esto se cuenta con el Telescopio de Neutrones Solares (TNS) que tiene la capacidad de medir la energía primaria y la dirección de arribo de las partículas incidentes. Instalado en el volcán de Sierra Negra, Puebla a más de 4500 m.s.n.m.

Se tiene pensado extender las funcionalidades de PySMG para interpretar estos datos. El TNS arroja datos con resolución de 1 minuto con un formato de compresión. (\*.sn) y cuenta con más de un canal de graficación. Para más información acerca del funcionamiento del TNS consultar: (González, 2008).

# **APÉNDICES**

# APENDICE A.

## GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

## TERMS AND CONDITIONS

### 0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

#### 1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

#### 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided

the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

### 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

### 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

### 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions

of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

#### 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).



The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

## 7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

## 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any

patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

#### 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

#### 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

#### 11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent

infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

#### 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

#### 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

#### 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does

not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

#### 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

#### 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

## APENDICE B.

### MANUAL DE USUARIO

#### PySMG



#### PySMG 2.0

#### General

#### Requerimientos del sistema

#### Instalación

#### Uso de PySMG

#### Graficar datos del Monitor de Neutrones 6-NM64

#### Herramientas de las gráficas

#### General

PySMG es un programa de software libre. Es un programa de distribución gratuita bajo la Licencia **GNU/GPL v3** para analizar y graficar datos de monitores de neutrones solares.

PySMG está diseñado y desarrollado en Python 2.7 en plataformas GNU Linux. Sin embargo debido a que Python es multiplataforma puede ser ejecutado en MS Windows y Mac OS X.

## Requerimientos del sistema

Sistemas basados en MS Windows:

Hardware:

**Procesador:** 1 GHZ de 32 o 64 bits.

**Memoria:** 2 GB de RAM

**Almacenamiento:** 100 MB disponibles

Software:

**Python:** versión 2.7, disponible en: <http://www.python.org>

**Matplotlib:** versión más actual en: <http://www.matplotlib.org>

**Pycairo:** versión más actual en: <http://www.cairographics.org>

**Numpy:** versión más actual en: <http://www.numpy.org>

**PyGTK:** versión más actual en: <http://www.pygtk.org>

**PyGObject:** versión más actual en: <http://www.wiki.gnome.org>

**PyParsing:** versión más actual en: <http://pyparsing.wikispaces.com>

**Python-dateutil:** versión más actual en: <https://labix.org>

Sistemas basados en GNU Linux:

Hardware:

**Procesador:** 1 GHZ de 32 o 64 bits.

**Memoria:** 2 GB de RAM

**Almacenamiento:** 100 MB disponibles

Software:

**Python:** versión 2.7

**Matplotlib:** instalar mediante:

Debian/Ubuntu:

**sudo apt-get install python-matplotlib**

Fedora/RedHat:

**sudo yum install python-matplotlib**

NOTA: La mayoría de los sistemas basados en GNU Linux tienen pre-instalado Python, de no ser así instalar mediante:

Debian/Ubuntu:

```
sudo apt-get install python2.7
```

Fedora/RedHat:

```
sudo yum install Python
```

## **Instalación**

La versión más reciente de PySMG se encuentra disponible en:

<https://github.com/ingOsvaldo/PySMG>

Fedora/Ubuntu:

Para instalar PySMG copia el contenido de las carpetas en un directorio, ej:

```
/home/USUARIO/Pysmg-2.0
```

También se puede clonar directamente en el directorio de su elección mediante:

```
$ git clone https://github.com/ingOsvaldo/PySMG.git
```

Una vez descargada la versión más reciente de PySMG, cambie las rutas en el archivo PySMG.desktop y coloque las rutas donde copió el contenido del software, después copie y guarde el archivo en la siguiente ruta:

```
/usr/share/applications/Pysmg.desktop
```

MS Widows:

Ejecutar directamente el archivo **/Source/Pysmg.py**

## Uso de PySMG

Una vez que se hayan cumplido todas las dependencias y que Python 2.7 este instalado en nuestra computadora, basta con:

GNU/Linux:

Se puede buscar en el menú de lanzadores y ejecutar como cualquier otro programa.

También se puede ejecutar el archivo principal desde la consola:

**\$ python Pysmg.py**

MS Windows:

En MS Windows los archivos **\*.py** son ejecutables, por lo que solo hay que dar doble clic al archivo **Pysmg.py**.

The screenshot shows the PySMG application window. The title bar reads "PySMG - Instituto de Geofísica". Below the title bar is a menu bar with "Archivo", "Herramientas", and "Ayuda". The main content area is organized into several sections:

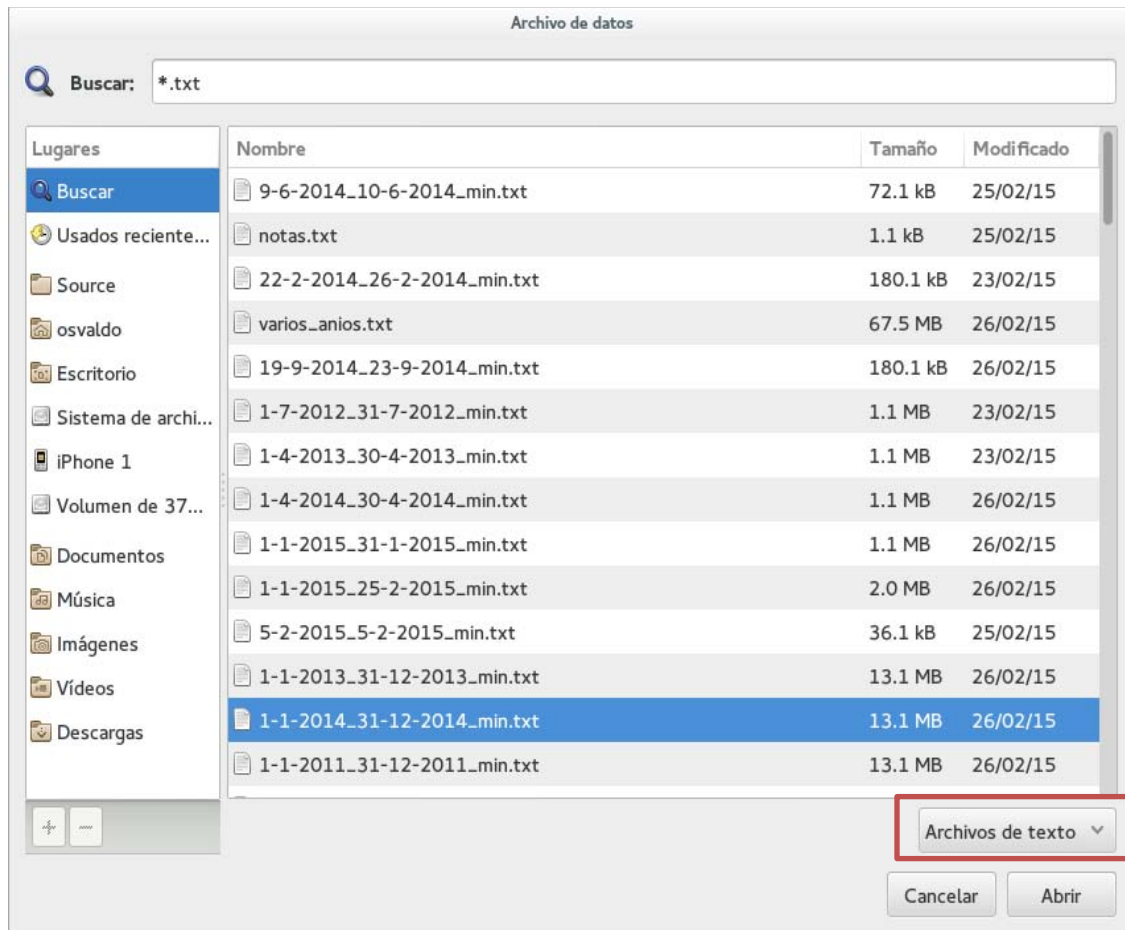
- Archivo de Datos:** Contains an "Examinar:" text input field, a "Cargar" button, and an "Examinar..." button.
- Fechas:** Features two radio buttons: "Graficar todas las fechas" (selected) and "Seleccionar fechas:". Below these are date pickers for "De:" and "Hasta:".
- Resolución:** Includes a dropdown menu for selecting resolution.
- Estadística:** Contains a "Rango:" section with date pickers for "De:" and "Hasta:", a "Generar gráfica" checkbox, and a "Minima (1 minuto)" dropdown. Below this is a "comparativo:" section with similar date pickers and a "Calcular" button.
- Bottom Bar:** Contains "Graficar" and "Salir" buttons.

Esta es la pantalla principal de Pysmg la cual nos permite tener acceso a todas sus funciones y que a continuación se describen:

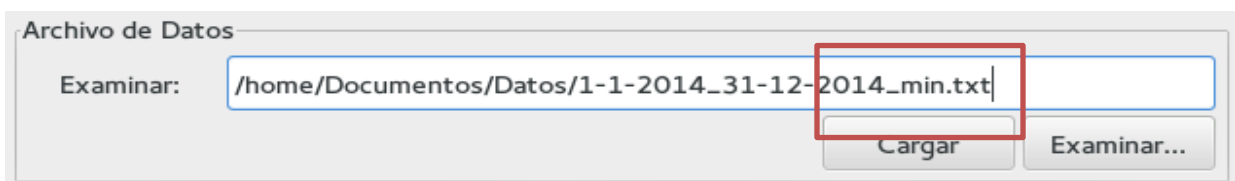


## Graficar datos del Monitor de Neutrones 6-NM64

La secuencia para graficar este tipo de datos (\*.txt) es básicamente la siguiente, primero se debe seleccionar el tipo de archivo a graficar:



El área seleccionada muestra el tipo de archivo de datos a buscar. Se selecciona el archivo (en este caso los datos del año 2014) y se cargan los datos:



PySMG identifica el tipo de archivo y de datos que el usuario carga y con base en ellos llena el panel de fechas:

**Fechas**

Graficar todas las fechas

Seleccionar fechas:

De:

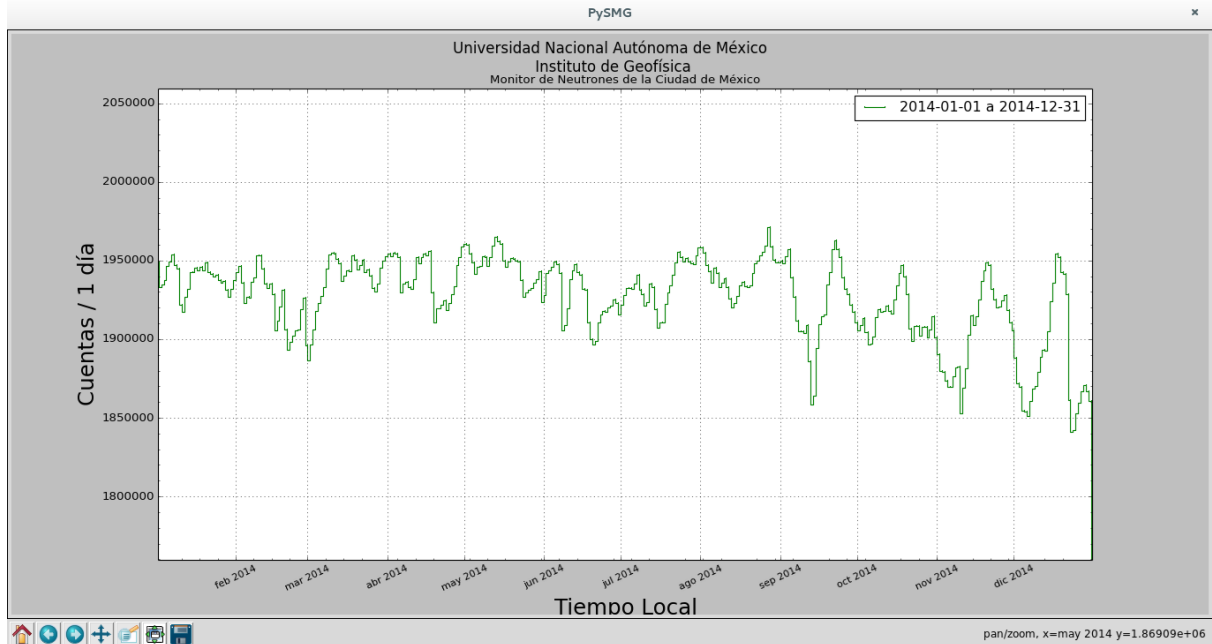
Hasta:

PySMG nos da opciones: (1) graficar todas las fechas contenidas en el archivo de datos o (2) graficar un intervalo de fechas/tiempo contenido en el archivo. Para archivos \*.txt está disponible una resolución de 1 (mínima), 5 y 30 minutos, 1 hora y 1 día:

**Resolución**

Resolución:

El resultado de esta gráfica es:



También se tiene la posibilidad de calcular estadística básica mediante el siguiente panel:

Estadística

Rango:

De: 2012-03-09 00:00 30 minutos

Hasta: 2012-03-13 23:59

comparativo:

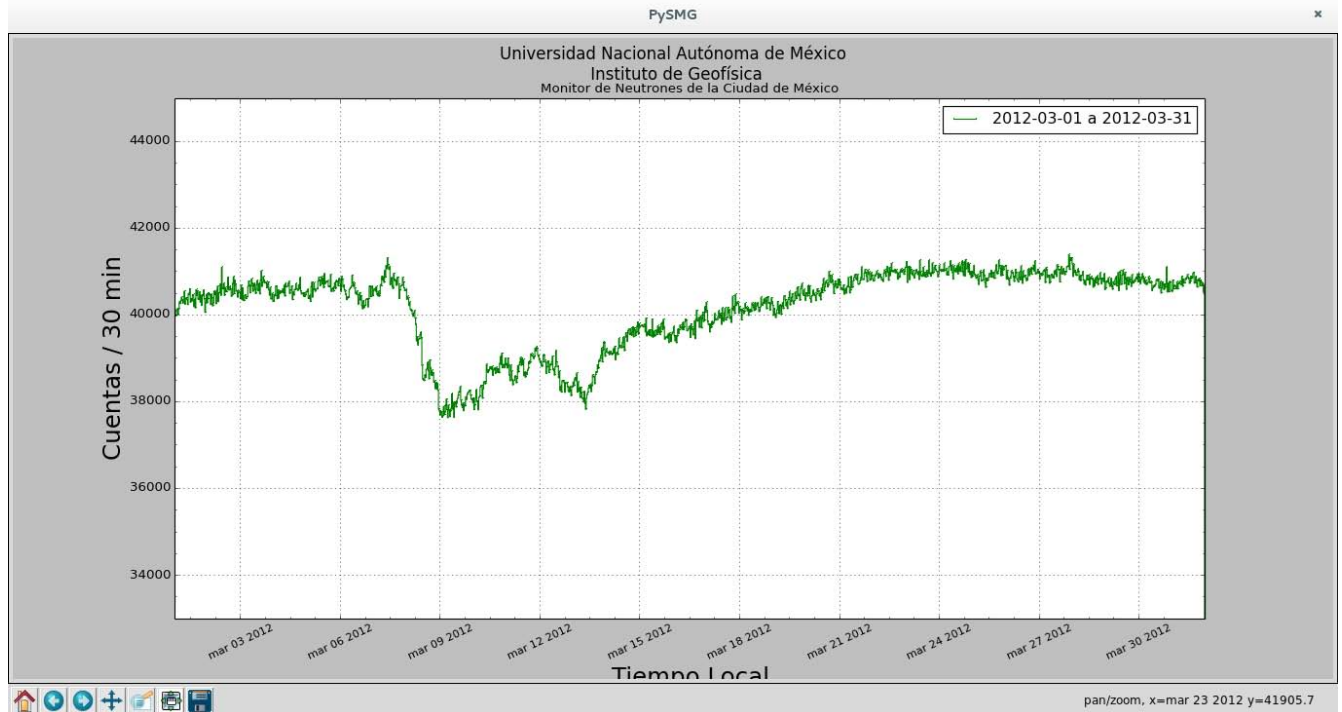
De: 2012-03-20 00:00

Hasta: 2012-03-24 23:59

Generar gráfica

Calcular

El panel de estadística nos da la opción de seleccionar dos (2) rangos de fechas a comparar, la posibilidad de generar la gráfica del rango elegido y la resolución de la gráfica. Este es el resultado:



En esta gráfica se muestra un decrecimiento en las cuentas a partir del 09 de marzo al 15 de marzo, por lo que se compara con un conteo normal del mismo mes:

Resultados				
	Rangos	Media	Min/Max	Media (%)
1	2012-03-09 a 2012-03-13 00:00 a 23:59	38485.08	37590/39356	100%
2	2012-03-20 a 2012-03-24 00:00 a 23:59	40858.63	39571/41259	6.17%

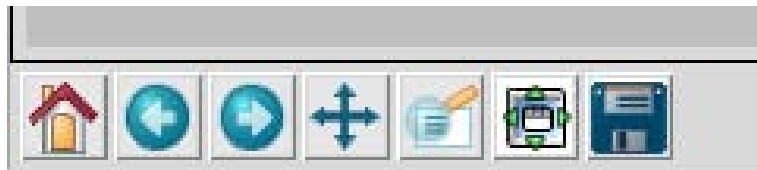
En la imagen anterior se muestra el cálculo de la estadística, la cual nos indica la Media de ambos rangos, los mínimos y máximos de cada rango y el comparativo en porcentaje. La última columna se obtiene al comparar el segundo rango de fechas con el primer rango de fechas.

**NOTA:** En caso de que el porcentaje salga negativo (ej. -6.17%) esto significará que el segundo rango de fechas está por debajo de la media del primer rango.

En este caso indica que el segundo rango está 6.17% por encima del primer rango.


## Herramientas de las gráficas

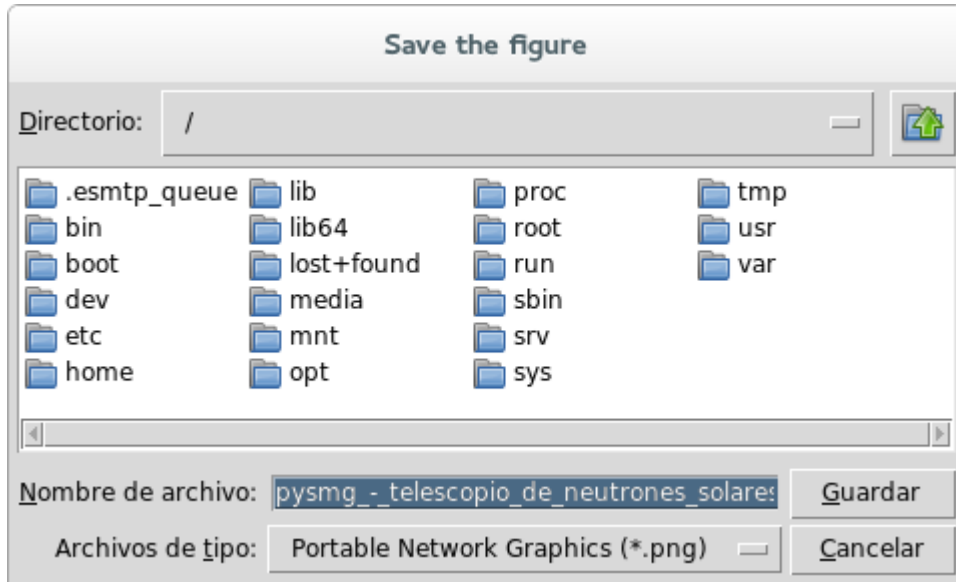
PySMG cuenta con herramientas para visualizar mejor las gráficas generadas, a continuación se muestra cómo usarlas:



Cada gráfica tiene una barra inferior izquierda con la cual podemos manipular las configuraciones de las mismas.


### Guardar

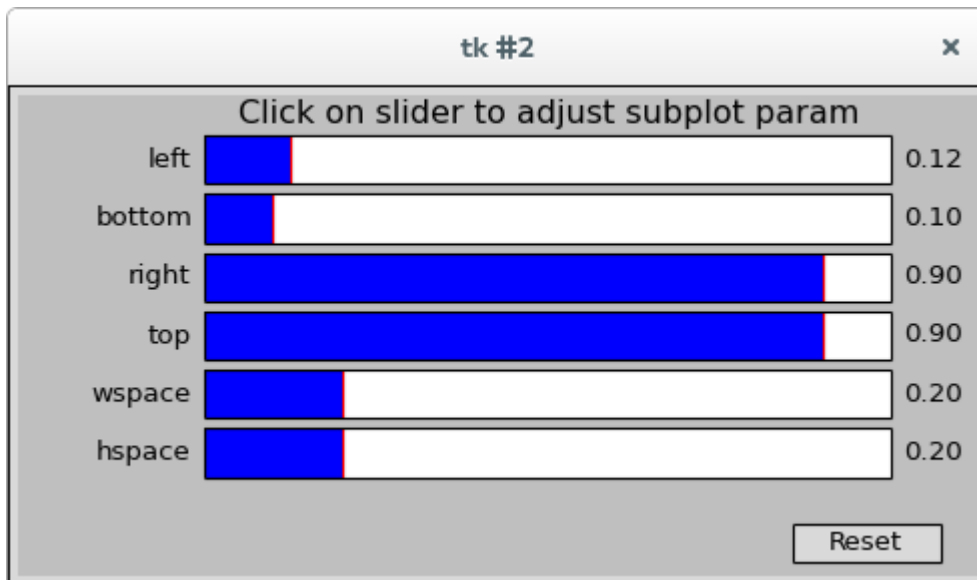
Para guardar una imagen de la gráfica que acabamos de realizar con PySMG debemos dar clic en el botón  el cual nos abrirá un explorador de archivos para guardar la imagen en la ubicación de nuestra preferencia:



Tenemos la facilidad de guardar la gráfica en formato imagen (\*.png o \*.jpg) o directamente en formato PDF (\*.pdf).

### Posición de las sub-gráficas

Podemos cambiar la posición de las gráficas, así como el espaciado y el tamaño que ocupan en la ventana mediante  :



El atributo **left** nos permite decidir a qué distancia del borde izquierdo queremos que se sitúen las sub-gráficas.

El atributo **bottom** nos permite decidir a qué distancia del borde inferior queremos que se sitúen las sub-gráficas.

El atributo **right** nos permite decidir a qué distancia del borde derecho queremos que se sitúen las sub-gráficas.


El atributo **top** nos permite decidir a qué distancia del borde superior queremos que se sitúen las sub-gráficas.

El atributo **wspace** nos permite decidir la distancia horizontal entre las sub-gráficas.

El atributo **hspace** nos permite decidir la distancia vertical entre las sub-gráficas.

Por último el botón **reset** regresa los valores por default.


## **Zoom**

En ocasiones, las gráficas con menor resolución y con un rango de fechas muy amplio, los datos se muestran muy “amontonados”, por lo que es necesario tener una mejor visión de las gráficas. Esto es posible con el botón .

Si se selecciona un espacio de la gráfica con el botón izquierdo del mouse, ese espacio seleccionado se ampliará. Si se selecciona un espacio de la gráfica con el botón derecho del mouse, ese espacio se alejará.

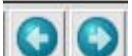
**NOTA:** Es importante mencionar que el Zoom es una herramienta con la que no se puede modificar las escalas (vertical y horizontal) por lo que si se necesita mejorar la gráfica con base en un rango de datos específicos, es mejor usar el panel de fechas de PySMG.

## Reajuste de las escalas


En ocasiones la media de las cuentas en resoluciones diferentes a la mínima, escapan ya sea por encima o por debajo de la media de la gráfica, provocando un mínimo o un máximo dependiendo del caso y la escala generada por PySMG no es la adecuada para la correcta visualización de la gráfica, para modificar el eje Y tenemos: 

Para mover la gráfica en cualquier dirección se debe arrastrar la gráfica mediante el clic izquierdo. Para ampliar la escala de igual manera se debe arrastrar la gráfica mediante el clic derecho. Ambas opciones funcionan de arriba-abajo y derecha-izquierda.

## Navegación entre vistas

Una vez que ya usamos el zoom en varias sub-gráficas, podemos regresar a la vista anterior o navegar entre ellas mediante la flechas de navegación: 

## Home

Finalmente para regresar los parámetros que hemos cambiado a los de inicio, lo podemos realizar mediante el botón  .



# **GLOSARIO**

---

## **Å**

### ångström

Unidad de longitud empleada principalmente para expresar longitudes de onda, distancias moleculares y atómicas. · 27

---

## **A**

### API

Interfaz de Programación de Aplicaciones. · 74, 76, 83, 88, 116

### ASCII

Código Estándar Estadounidense para el Intercambio de Información. · 44, 99

---

## **B**

### BF<sub>3</sub>

Trifloruro de Boro · 40, 41

### Bremsstrahlung · 27, 28, 29, 32, 35

### Bytecode

Código intermedio más abstracto que el código máquina. · 74

---

## **C**

### Convección

Es una de las tres formas de transferencia de calor y se caracteriza porque se produce por medio de un fluido que transporta el calor entre zonas con diferentes temperaturas. · 14, 16

### CPython

Es la aplicación canónica del lenguaje Python. · 74

---

## **D**

### Decrecimiento Forbush

Es la disminución temporal del número de rayos cósmicos galácticos que llegan a la Tierra. · 131

### Densidad

Es una magnitud escalar referida a la cantidad de masa en un determinado volumen de una sustancia. · 13, 15, 16, 17, 20, 21, 22, 27, 30, 33, 34, 42, 131

### Deuterio

Isótopo estable del hidrógeno que se encuentra en la naturaleza con una abundancia del 0,015 % átomos de hidrógeno · 29

---

## **E**

### Exotérmico

Cualquier reacción química que desprenda energía, ya sea como luz o calor. · 15

### Eyección

Onda hecha de radiación y viento solar que se desprende del Sol en el periodo llamado Actividad Máxima Solar. · 25, 26, 34, 131

---

## **F**

### Fulguraciones

Intenso estallido de radiación procedente de la liberación de energía magnética asociada a manchas solares. · 9, 12, 24, 25, 26, 27, 33, 34, 35

---

## **G**

### Giro-sincrotrón

Electrones moviéndose a velocidades cercanas a la luz emiten un haz de radiación al girar alrededor de una línea de campo magnético. · 32

### Granulación

Está formada por manchas más claras y manchas más oscuras que varían continuamente de forma y dimensiones. · 16

### GTK

Biblioteca grafica para el desarrollo de GNOME · 88

---

## **H**

### H $\alpha$

H-alfa es una de las líneas de emisión del espectro del Hidrógeno. · 18, 25, 26, 31, 33

---

## **I**

### IDE

Entorno de Desarrollo Integrado. · 79

### Interplanetario

Espacio existente entre dos o más planetas. · 7

---

## L

### Licencia

Es un contrato entre el autor de un software y el usuario que contiene una serie de términos y condiciones que se deben cumplir para su uso. · 93, 116

### Línea de Balmer

Conjunto de rayas que resultan de la emisión del átomo de hidrógeno cuando un electrón transita desde un nivel mayor a uno menor. · 31

### Linux

Sistema operativo de software libre y código abierto. · 9, 47, 82, 83, 85, 88, 90, 91, 117

### Lisp

Lenguaje de programación multiparadigma utilizado principalmente en inteligencia artificial. · 55

---

## M

### Magnetósfera

Es una región alrededor de un planeta en la que el campo magnético de éste desvía la mayor parte del viento solar formando un escudo protector contra las partículas cargadas de alta energía procedentes del Sol. · 7

### Matplotlib

Es una biblioteca para la generación de gráficos a partir de datos contenidos en listas. · 112, 117, 147

### MeV

El Electronvoltio (eV) es la unidad de energía que representa la variación de energía potencial que experimenta un electrón al moverse desde un punto de potencial  $V_a$  hasta un punto  $V_b$ . · 15, 29, 42

---

## N

### Neutrones

partícula subatómica, sin carga neta · 8, 10, 36, 38, 39, 40, 91, 127, 134, 135

### Numpy

Es el paquete fundamental para el cómputo científico del lenguaje Python. · 112, 117, 147

---

## P

### Paradigma

Es un estilo de programación empleado. · 9, 63, 71, 72, 79, 116

### PEP

Python Enhancement Proposal · 48, 50, 57

### Plasma

Gas totalmente ionizado, que consta de igual número de iones y electrones, por lo que es casi neutro en estado estacionario y cuya dinámica presenta efectos colectivos dominados por las interacciones electromagnéticas. · 13, 16, 20, 32, 33, 34

### Proceso

Una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados. · 14, 15, 16, 21, 24, 27, 73, 76, 77, 108, 111

### Protón

Es una partícula subatómica con una carga eléctrica elemental positiva 1, igual en valor absoluto y de signo contrario a la del electrón. · 15, 27

### Python

Lenguaje de programación. · 9, 10, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 61, 63, 64, 65, 66, 68, 69, 70, 71, 72, 73, 74, 75, 76, 79, 80, 82, 83, 86, 88, 89, 91, 116, 117, 118, 119, 135

---

## R

### Resolución

Exactitud o claridad en la reproducción de una imagen. · 25, 44, 99, 122, 123, 124, 127, 128, 135

---

## S

### Software

Equipamiento lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware. · 7, 8, 9, 10, 12, 44, 79, 83, 91, 93, 94, 95, 96, 97, 98, 119, 120, 134, 138, 141, 144

### Supergranulación

contiene cientos de gránulos individuales y sobrevive entre 12 a 20 horas. · 16

---

## **T**

### **Tcl/Tk**

Lenguaje de script utilizado para el desarrollo de interfaces gráficas. · 80

### **Thread**

Es un mecanismo que permite a una aplicación realizar varias tareas a la vez de manera concurrente. · 74, 111

### **Threads**

Es un mecanismo que permite a una aplicación realizar varias tareas a la vez de manera concurrente. · 73, 74, 76

---

## **U**

### **UML**

Lenguaje Unificado de Modelado. · 104

---

## **V**

### **Variación Diurna**

Se define como el cambio de temperatura entre el día y la noche, producido por la rotación de la Tierra. · 129

---

## **W**

### **Windows**

Sistema operativo de Microsoft. · 9, 47, 82, 83, 85, 88, 89, 90, 117

### **wxWidgets**

Biblioteca multiplataforma para el desarrollo de interfaces gráficas en C++. · 82

---

## **Z**

### **Zona Convectiva**

Es donde se lleva a cabo el proceso de convección que es el principal mecanismo de transporte de energía. · 14, 15, 16

### **Zona Radiativa**

Es donde se transporta la energía desde el núcleo por la radiación. · 14, 15

# **BIBLIOGRAFÍA**

- Ambler, S. W. (2003). *UML 2 Package Diagrams: An Agile Introduction*. Recuperado el 04 de Mayo de 2015, de <http://www.agilemodeling.com/artifacts/packageDiagram.htm>
- Bahit, E. (2012). *Curso Python para principiantes*. Recuperado en Enero 2015, de <http://www.cursosdeprogramacionadistancia.com/static/pdf/material-sin-personalizar-python.pdf>
- Cliver, E. W. (1996). *High Energy Solar Physics*. New York: AIP.
- Comunidad Python Argentina. (2012). *Interfaces Gráficas (GUI)*. Recuperado en Enero 2015, de <http://python.org.ar/InterfacesGraficas>
- Deitel, P. J. & Deitel, H. M. (2008). *Cómo programar en Java*. (7ma ed.). México: Pearson Educación.
- Downey, A., Elkner, J., y Meyers, C. (2002). *Aprende a pensar como un programador con Python*. Massachusetts: Green Tea Prees.
- Foukal, P., Gerald, N. y Tom, W. (2004). *A Stellar View on Solar Variations and Climate*. U.S.A: Science.
- Free Software Foundation. (2015). *Definición de Software Libre*. Recuperado en Febrero 2015, de <https://www.gnu.org/philosophy/free-sw.es.html>
- García, A. L. (2011). *Introducción a Python*. España.
- García, R. (2014). *Estimación del Espectro de Alturas de Pulsos con corrección del efecto apilamiento para el Monitor de Neutrones 6NM- 64*. Tesis de maestría. México: UNAM.
- González, L. X. (2008). *El telescopio de neutrones solares en Sierra Negra y aceleración de iones en la atmósfera solar*. Tesis doctoral. México: UNAM, Instituto de Geofísica.
- González, R. (2010). *Python para todos*. Recuperado en Febrero 2015, de: <http://mundogeek.net/tutorial-python/>

Gutiérrez, D. (2009). *UML Diagramas de paquetes*. Venezuela: Universidad de los Andes. Recuperado el 04 de mayo de 2015, de

[http://www.codecompiling.net/files/slides/UML\\_clase\\_05\\_UML\\_paquetes.pdf](http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf)

----- (2011a). *Diagramas de Casos de Uso*. Venezuela: Universidad de los Andes. Recuperado el 04 de mayo de 2015, de

[http://www.codecompiling.net/files/slides/UML\\_clase\\_02\\_UML\\_casos\\_de\\_uso.pdf](http://www.codecompiling.net/files/slides/UML_clase_02_UML_casos_de_uso.pdf)

----- (2011b). *UML Diagrama de Secuencia*. Venezuela: Universidad de los Andes. Recuperado el 04 de mayo de 2015, de

[http://www.codecompiling.net/files/slides/UML\\_clase\\_06\\_UML\\_secuencia.pdf](http://www.codecompiling.net/files/slides/UML_clase_06_UML_secuencia.pdf)

Hipertextual. (2012). *Historia del Software: GUI (Graphical User Interface)*.

Recuperado en Enero 2015, de <http://hipertextual.com/archivo/2012/02/historia-del-software-gui-graphical-user-interface/>

Instituto de Geofísica. *Departamento de Ciencias Espaciales*. Recuperado en Noviembre 2014, de <http://www.geofisica.unam.mx>

Lang, R. K. (2008). *The Sun from space*. (2a ed.). Alemania: Astronomy and Astrophysics library.

Longair, M. S. (1992). *High Energy Astrophysics, volume 1*. U.K.: Cambridge University Press.

Mundo Geek. (2008). *Threads en Python*. Recuperado en Enero 2015, de

<http://mundogeek.net/archivos/2008/04/18/threads-en-python/>

Murphy, R. J. (1991). *Astrophys*. U.S.A: AIP.

Open Source Initiative. *Open Source Definition*. Recuperado en Marzo 2015, de

<http://opensource.org/osd>

Peters, T. (2004). *PEP 20 – The Zen of Python*. Recuperado en Enero 2015, de

<https://www.python.org/dev/peps/pep-0020/>

Python Central. (2012). *Overview of Python GUI development (Hello World)*. Recuperado en Enero 2015, de <http://www.pythoncentral.io/introduction-python-gui-development/>

Python Software Foundation. (1994 – 2014). *Python 2.7.8 documentation*. Recuperado en Enero 2015, de <https://docs.python.org/2.7/>

----- (2013). *GUI Programming in Python*. Recuperado en Enero 2015, de <https://wiki.python.org/moin/GuiProgramming>

Reames, D. V. (1999). *Particle acceleration at the sun and in the Heliosphere*. U.S.A: NASA.

Recursos Python. (2013). *Multiprocesing – Tareas concurrentes con procesos*. Recuperado en Enero 2015, de <http://recursospython.com/guias-y-manuales/multiprocessing-tareas-concurrentes-con-procesos/>

Schmelz, J., y Brown, J. (1992). *The Sun: A Laboratory for Astrophysics*. U.S.A: Kluwer Academic Plubishers.

The Matplotlib Development Team. (2002). *Matplotlib*. Recuperado el 06 de mayo de 2015, de <http://matplotlib.org/>

Universidad de Chile. (1999). *Casos de uso*. Chile: Universidad de Chile. Recuperado el 06 de mayo de 2015, de <http://users.dcc.uchile.cl/~psalinas/uml/casosuso.html>

Van Rossum, G. (2001). *PEP 8 – Style Guide for Python Code*. Recuperado en Enero 2015, de <https://www.python.org/dev/peps/pep-0008/>

Vega, M. (2010). *Casos de Uso, UML*. España: Universidad de Granada. Recuperado el 06 de mayo de 2015, de <http://lsi.ugr.es/~mvega/docis/casos%20de%20uso.pdf>

Zirin H., (1972), *Astrophysics of the Sun*. U.K.: Cambridge University Press.