Ing. Ventre, Luis O.

CAPITULO 3:

PROGRAMAS

- Hasta el momento hemos utilizado el INTERPRETE de Pythong.
- Utilizar el sistema de esta manera es limitante.
- Es de interés, **generar un archivo** con un grupo de instrucciones y solicitarle al interprete que las ejecute a todas JUNTAS.
- Denominaremos **PROGRAMA**, al contenido de este archivo.

El entorno PYTHONG

- El entorno PythonG, es un programa escrito en Python útil para el desarrollo de programas.

- Escribamos nuestro primer programa:

- Click en FICHERO

- Click en NUEVO

- Puedes regresar al **INTERPRETE INTERACTIVO** con un click en la pestaña <PYTHON>

- Copia el siguiente código

miprograma_4.py

miprograma.py

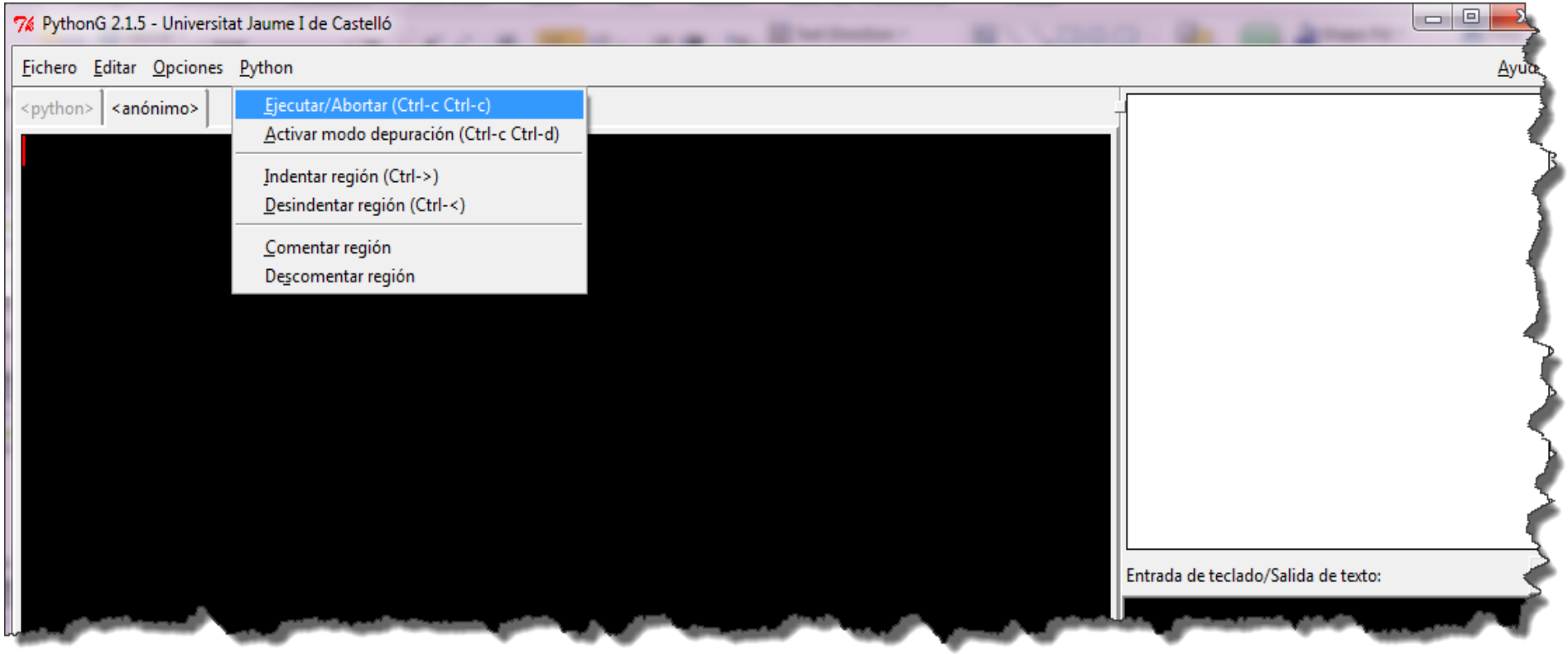
```
1 from math import pi
2
3 radio = 1
4 perimetro = 2 * pi * radio
5
6 perimetro
```

- Click en fichero

- Click en GUARDAR COMO, y escoge un NOMBRE.PY

El entorno PYTHONG

- La opción EJECUTAR/ABORTAR del menú Python, sirve para ejecutar todo tu programa, selecciónala, que pasa?



El entorno PYTHONG

miprograma_4.py

miprograma.py

```
1 from math import pi
2
3 radio = 1
4 perimetro = 2 * pi * radio
5
6 perimetro
```

Línea 1: importar una definición de una constante.

Línea 2: en blanco = LEGIBILIDAD.

Línea 3: asignación e inicialización de variable radio.

Línea 4: asignación e inicialización de variable perímetro

Línea 5: en blanco = LEGIBILIDAD.

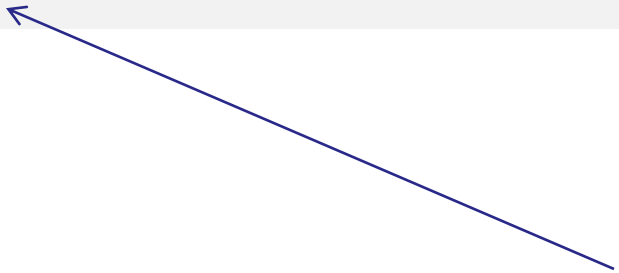
Línea 6: expresión, muy simple.

El entorno PYTHONG

miprograma_4.py

miprograma.py

```
1 from math import pi
2
3 radio = 1
4 perimetro = 2 * pi * radio
5
6 perimetro
```



- **A diferencia del interprete, un PROGRAMA no muestra por pantalla el resultado de una expresión!!! OJO**

El entorno PYTHONG

- **Como IMPRIMIR un resultado por pantalla…**
- **Se utilizara una nueva sentencia: PRINT**
- En principio se utilizaría así:

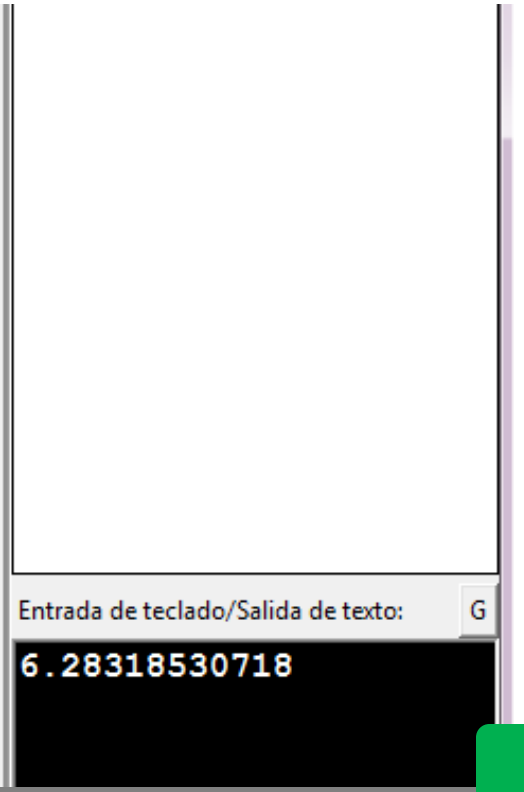
miprograma.py

```
1 from math import pi
2
3 radio = 1
4 perimetro = 2 * pi * radio
5
6 print perimetro
```

El entorno PYTHONG

- Al ejecutarlo, hay alguna diferencia?.....
- En la ventana inferior derecha, debería aparecer la **SALIDA POR PANTALLA** generada por la sentencia **PRINT**.

```
from math import pi

radio = 1
perimetro = 2 * pi * radio

print perimetro
```



Entrada/Salida

- Supongamos que escribimos este programa.
- Pero ahora el **radio de nuestra esfera es de 2.5**…
- Que deberíamos hacer para conocer el volumen nuevo?

volumen_esfera_8.py

volumen_esfera.py

```
1 from math import pi
2
3 radio = 1
4 volumen = 4.0 / 3.0 * pi * radio ** 3
5
6 print volumen
```

Entrada/Salida

- Y si ahora el **radio de nuestra esfera es de 3.5**…

- Que deberíamos hacer para conocer el volumen nuevo?

volumen_esfera_8.py

volumen_esfera.py

```
1 from math import pi
2
3 radio = 1
4 volumen = 4.0 / 3.0 * pi * radio ** 3
5
6 print volumen
```

Lectura de datos de teclado

- Esto es el colmo!...
- Vamos a modificar nuestro programa para que nos **SOLICITE** el **ingreso** del RADIO por **teclado** antes de hacer el calculo del volumen.
- Para esto se usara una función predefinida llamada:

raw_input (arg.)

- Esta función, **detiene la ejecución** del programa y devuelve una **‘cadena’** con el texto que tecleo el usuario.!

- Pero necesitamos que el **RADIO** sea un valor **FLOTANTE**.

- Debemos usar una **función ya vista**, para **convertir** esta cadena en flotante.

float (arg.)

- Esta función, recibirá como argumento la cadena ingresada por teclado y proporcionara un numero con coma flotante.

- La función raw_input, por ser una función debe llevar si o si los paréntesis de apertura y cierre.

Lectura de datos de teclado

- Las modificaciones serán:

volumen_esfera_10.py — volumen_esfera.py

```
1 from math import pi
2
3 texto_leido = raw_input()
4 radio = float(texto_leido)
5 volumen = 4.0 / 3.0 * pi * radio ** 3
6
7 print volumen
```

- El nuevo programa quedará!...

volumen_esfera_10.py — volumen_esfera.py

```
1 from math import pi
2
3 texto_leido = raw_input()
4 radio = float(texto_leido)
5 volumen = 4.0 / 3.0 * pi * radio ** 3
6
7 print volumen
```

Lectura de datos de teclado

- Una versión mas breve:

volumen_esfera_11.py

volumen_esfera.py

```
1 from math import pi
2
3 radio = float(raw_input())
4 volumen = 4.0 / 3.0 * pi * radio ** 3
5
6 print volumen
```

- Este programa nos permite modificar el valor del radio, pero es poco elegante, **NO NOS LO INDICA!**...un usuario distraído difícilmente lo entenderá.

Lectura de datos de teclado

- La función:

raw_input (arg.)

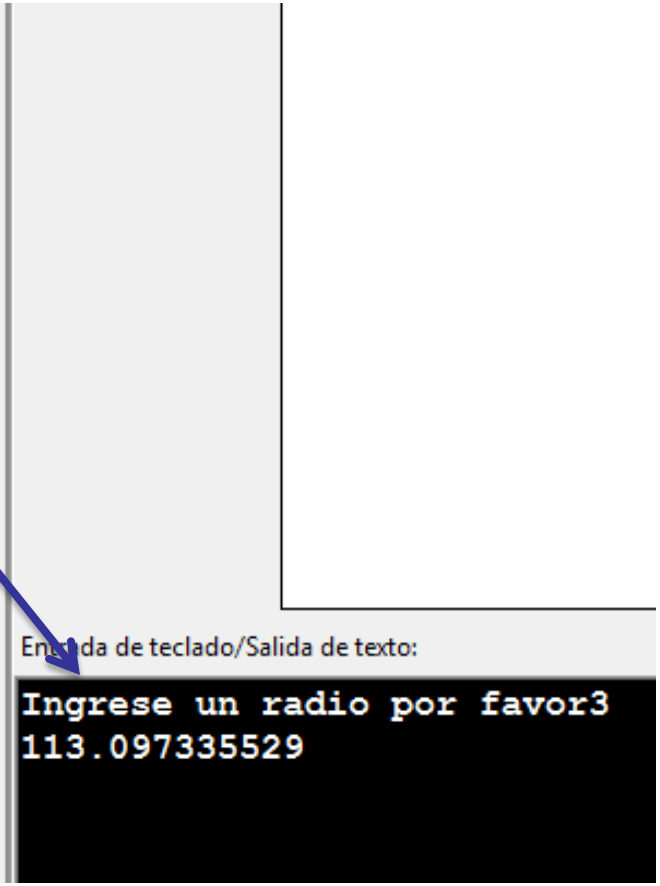
- Acepta un argumento, el cual es el mensaje a mostrar en solicitud del ingreso.

raw_input (‘Ingrese un radio:’)

Lectura de datos de teclado

- El programa quedara:

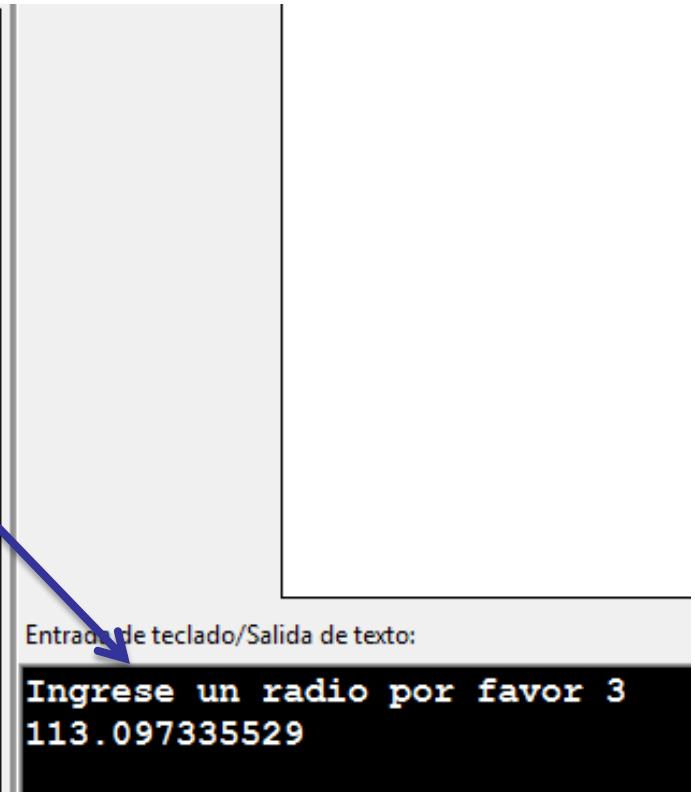
```
from math import pi

textoleido=raw_input('Ingrese un radio por favor')
radio=float(textoleido)
volumen=4.0/3.0 * pi * radio**3

print volumen
```



Lectura de datos de teclado

- Y su versión acotada:

```
from math import pi

radio=float(raw_input('Ingrese un radio por favor'))
volumen=4.0/3.0 * pi * radio**3

print volumen
```



Mas sobre PRINT

- Con la función **PRINT**, podemos hacer aún más presentable nuestro programa.

- Agreguemos estas dos líneas donde consideres!

```
print 'Programa para el cálculo  del volumen:'
```

```
print 'Muchas gracias por usar este programa!'
```

Mas sobre PRINT

```
from math import pi

print 'Programa para el calculo del volumen'
radio=float(raw_input('Ingrese un radio por favor'))
volumen=4.0/3.0 * pi * radio**3


print volumen

print 'Muchas gracias por usar este programa'
```

Entrada de teclado/Salida de texto:

```
Programa para el calculo del volumen
Ingrese un radio por favor 3
113.097335529
Muchas gracias por usar este programa
```

Mas sobre PRINT

- La sentencia PRINT, puede imprimir mas de un valor en la misma línea.

- Si separamos los mismos con una ‘,’ aparecerá un espacio en la impresión.

```
print ‘Volumen’,valor, ‘metros cubicos’
```

- De aquí podemos deducir que todo print hace que se comience en una **nueva** línea!, a menos que **finalice** con «,»

Mas sobre PRINT

volumen_esfera.py

```
1 from math import pi
2
3 print 'Programa para el cálculo del volumen de una esfera.'
4
5 radio = float(raw_input('Dame el radio (en metros): '))
6 volumen =  4.0/3.0 * pi * radio ** 3
7
8 print 'Volumen de la esfera:',
9 print volumen, 'metros cúbicos'
```

```
Programa para el cálculo del volumen de una esfera.
Dame el radio (en metros): 2
El volumen de la esfera es de 33.5103216383 metros cúbicos
```

Sera?….

Salida con Formato

- Que pasa si queremos controlar aun mas la salida de los datos. Por ej, sin espacios en blanco cuando colocamos una «,».

- Si queremos que todos los números ocupen el mismo numero de casillas?.

- Python, nos brinda el operador de formato **%**.

Vimos su funcionamiento con float y enteros. Pero aquí operara con cadenas a la izquierda y valores a su derecha.

Salida con Formato

- Observa detenidamente el programa.

potencias_1.py

potencias.py

```
1 numero = int(raw_input('Dame un número: '))
2
3 print '%d elevado a %d es %d' % (numero, 2, numero ** 2)
4 print '%d elevado a %d es %d' % (numero, 3, numero ** 3)
5 print '%d elevado a %d es %d' % (numero, 4, numero ** 4)
6 print '%d elevado a %d es %d' % (numero, 5, numero ** 5)
```

- Las ultimas 4 líneas tienen la forma:

print ‘cadena’ % (valor, valor, valor)

Salida con Formato

- La ejecución de este programa es:.

```
Dame un número: 3
3 elevado a 2 es 9
3 elevado a 3 es 27
3 elevado a 4 es 81
3 elevado a 5 es 243
```

- Cada marca %, en la cadena:

```
'%d␣elevado␣a␣%d␣es␣%d'
```

ha sido reemplazada por un valor entero.

- **El fragmento «%d» significa aquí va un numero entero.**

- Ok, «%d» significa acá va un numero entero. Cual?

- El que resulte de evaluar las 3 expresiones que están a la derecha del operador %.

```
(numero, 2, numero ** 2)
```

- Un ejemplo:

```
'%d elevado a %d es %d' % (numero, 2, numero ** 2)
```

- Veamos el siguiente programa:

potencias.py

```
1 numero = int(raw_input('Dame un número: '))
2
3 print '%d elevado a %d es %4d' % (numero, 2, numero ** 2)
4 print '%d elevado a %d es %4d' % (numero, 3, numero ** 3)
5 print '%d elevado a %d es %4d' % (numero, 4, numero ** 4)
6 print '%d elevado a %d es %4d' % (numero, 5, numero ** 5)
```

- El tercer %d, de cada línea fue reemplazado por un %**4**d

- Y la ejecución?

Salida con Formato

- El valor al lado del %, implica que los valores aparecerán alineados a la derecha en la cantidad de casilleros indicados.

```
from math import pi

radio=float(raw_input('Ingrese un valor por favor'))

print '%d elevado a %d es %4d' % (radio, 2, radio**2)
print '%d elevado a %d es %4d' % (radio, 3, radio**3)
print '%d elevado a %d es %4d' % (radio, 4, radio**4)
print '%d elevado a %d es %4d' % (radio, 5, radio**5)
```

Entrada de teclado/Salida de texto:

```
Ingrese un valor por favor 5
5 elevado a 2 es   25
5 elevado a 3 es  125
5 elevado a 4 es  625
5 elevado a 5 es 3125
```

Salida con Formato

- Si la cantidad de casilleros selectos no alcanzan el valor se mostrara correctamente, aunque desalineado…

```
from math import pi

radio=float(raw_input('Ingrese un valor por favor'))

print '%d elevado a %d es %3d' % (radio, 2, radio**2)
print '%d elevado a %d es %3d' % (radio, 3, radio**3)
print '%d elevado a %d es %3d' % (radio, 4, radio**4)
print '%d elevado a %d es %3d' % (radio, 5, radio**5)
```

Entrada de teclado/Salida de texto:

```
Ingrese un valor por favor 5
5 elevado a 2 es  25
5 elevado a 3 es 125
5 elevado a 4 es 625
5 elevado a 5 es 3125
```

Salida con Formato

- El operador **%** acompañado de cadenas con formato se llama **OPERADOR DE FORMATO**.

- Veamos otro ejemplo:

area_con_formato.py

```
1 from math import pi
2
3 radio = float(raw_input('Dame el radio: '))
4 area = pi*radio**2
5
6 print 'El área de un círculo de radio %f es %f' % (radio, area)
7 print 'El área de un círculo de radio %6.3f es %6.3f' % (radio, area)
```

Salida con Formato

- La %f, implica que allí se reemplazara por un numero FLOTANTE.

area_con_formato.py

```
area_con_formato.py
1 from math import pi
2
3 radio = float(raw_input('Dame el radio: '))
4 area = pi*radio**2
5
6 print 'El área de un círculo de radio %f es %f' % (radio, area)
7 print 'El área de un círculo de radio %6.3f es %6.3f' % (radio, area)
```

- La utilización de un **numero con decimales** entre el operador % y la f. Implica cantidad de casilleros a ocupar, y luego del punto cuantos de ellos se destinaran a decimales.

Salida con Formato

- La %f, implica que allí se reemplazara por un numero FLOTANTE.

area_con_formato.py

```
area_con_formato.py
1 from math import pi
2
3 radio = float(raw_input('Dame el radio: '))
4 area = pi*radio**2
5
6 print 'El área de un círculo de radio %f es %f' % (radio, area)
7 print 'El área de un círculo de radio %6.3f es %6.3f' % (radio, area)
```

Salida con Formato

area_con_formato.py

```
area_con_formato.py
1 from math import pi
2
3 radio = float(raw_input('Dame el radio: '))
4 area = pi*radio**2
5
6 print 'El área de un círculo de radio %f es %f' % (radio, area)
7 print 'El área de un círculo de radio %6.3f es %6.3f' % (radio, area)
```

- La utilización de un **numero con decimales** entre el operador % y la f implica
- Su parte entera cantidad de casilleros a ocupar por todo el n*
- Su parte decimal casilleros que se destinaran a decimales.

```
Dame el radio: 2
El área de un círculo de radio 2.000000 es 12.566371
El área de un círculo de radio  2.000 es 12.566
```

- Hemos visto la utilización del operador de formato con enteros y flotantes, y strings?

saludo_3.py

```
1 nombre = raw_input('Tu␣nombre:␣')
2 print 'Hola,␣%s.' % (nombre)
```

saluda.py

```
Tu nombre: Juan
Hola, Juan.
```

Legibilidad

- Nuestros programas hasta aquí, solicitan el ingreso de datos, hacen cálculos e imprimen en pantalla. Puedes explicar este programa?

ilegible.py

```
1 h = float(raw_input('Dame h: '))
2 v = float(raw_input('y v: '))
3 z = h * v
4 print 'Resultado 1 %6.2f' % z
5 v = 2 * h + v + v
6 print 'Resultado 2 %6.2f' % v
```

- No esta muy claro?.....

Legibilidad

- Y ahora?

legible.py

```
1  print 'Programa para el cálculo del perímetro y el área de un rectángulo.'
2
3  altura = float(raw_input('Dame la altura (en metros): '))
4  anchura = float(raw_input('Dame la anchura (en metros): '))
5
6  area = altura * anchura
7  perimetro = 2 * altura + 2 * anchura
8
9  print 'El perímetro es de %6.2f metros.' % perimetro
10 print 'El área es de %6.2f metros cuadrados.' % area
```

- Es lo mismo….**pero distinto** no?
- Separar visualmente 4 zonas. Mensaje de bienvenida. Ingreso de datos. Cálculos. Impresión resultados.

Legibilidad

- Porque fue más fácil de leer el segundo?
- Utilización de identificadores representativos y tan largos como sea necesario.
- Estructura clara. No mezclar cálculos con impresión de resultados.
- Utilización de formulas frecuentes.
- Calidad en los mensajes para solicitar los valores.

Legibilidad

- Porque es bueno que sean legibles los programas:

- **Capacidad de adaptación a futuro**
  - Si hay que actualizar el programa luego de un tiempo.

- **Detección de errores de programación**
  - Al comprender fácilmente el funcionamiento del mismo.

- Si hay un nuevo programador y debe entender que hace el programa que esta en uso por una empresa.

Comentarios

- A medida que los programas ganan complejidad es mas difícil comprenderlos.
- Podemos ayudar con **COMENTARIOS**.
- Como su objetivo es solo ayudar al lector, PYTHON LOS OMITE.
- Necesitamos una **MARCA** para indicarle al interprete que esa línea completa es un comentario y no una instrucción.
- Para esto se utiliza el **símbolo «#»**

Comentarios

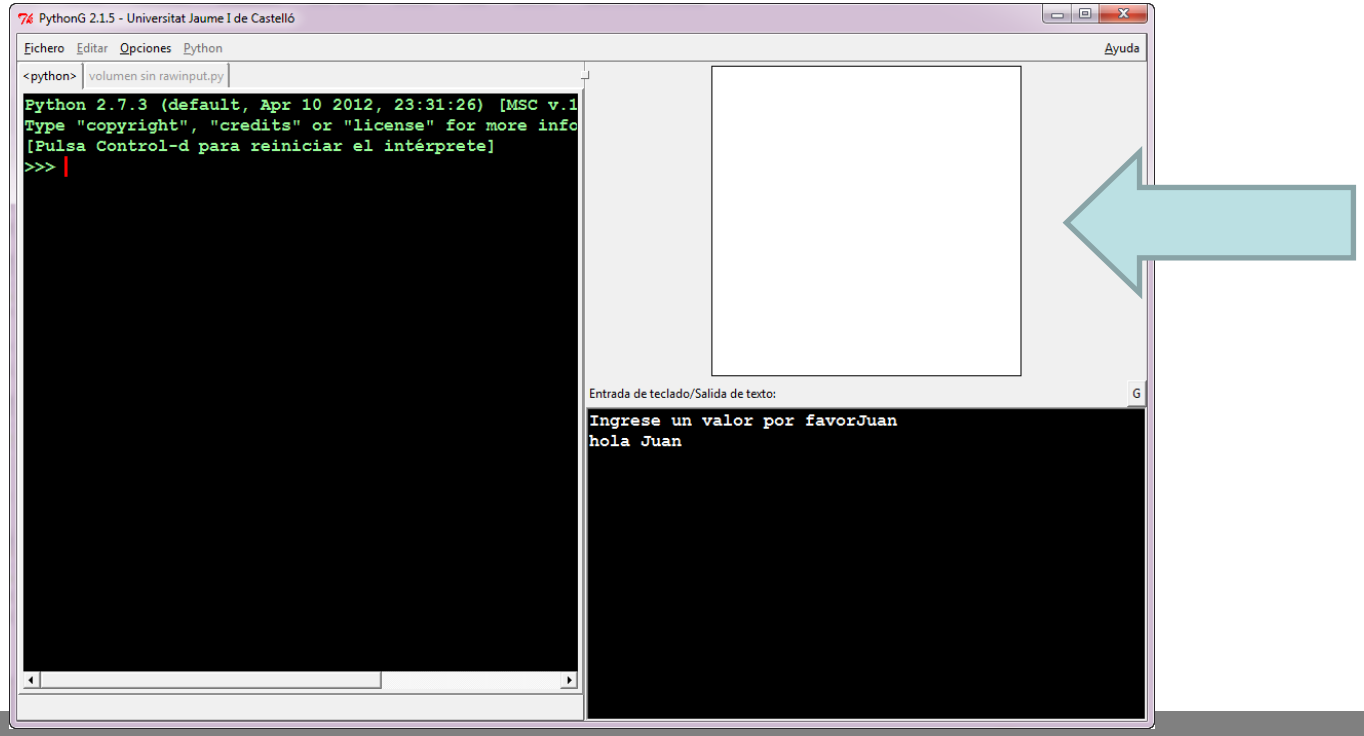
- No hay reglas fijas que indiquen cuando y donde comentar. Poco es malo, mucho también!....
- **Un ej.**

rectangulo.py

```
# Programa: rectangulo.py
# Propósito: Calcula el perímetro y el área de un rectángulo a partir
#            de su altura y anchura.
# Autor:     John Cleese
# Fecha:     1/1/2001

# Petición de los datos (en metros)
altura = float(raw_input('Dame la altura (en metros): '))
anchura = float(raw_input('Dame la anchura (en metros): '))

# Cálculo del área y del perímetro
area = altura * anchura
perimetro = 2 * altura + 2 * anchura

# Impresión de resultados por pantalla
print 'El perímetro es de %6.2f metros' % perimetro  # sólo dos decimales.
print 'El área es de %6.2f metros cuadrados' % area
```

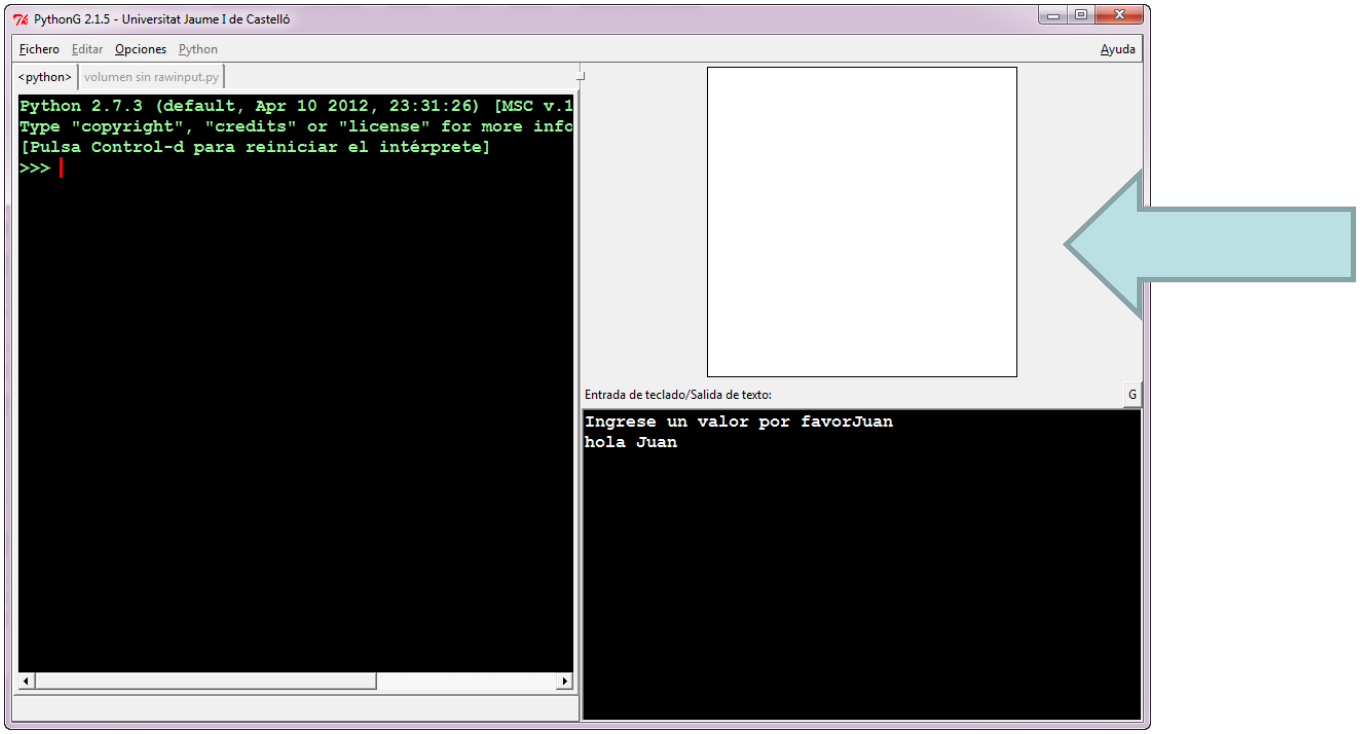
- Todos los programas vistos, fueron desarrollados con teclado y con la pantalla en modo texto para interactuar.

- **PythonG tiene una interfaz GRAFICA, que puedes observar en el recuadro superior derecho del entorno.**



Gráficos

- Este recuadro tiene coordenadas, la esquina inferior izquierda tiene coordenadas (0 , 0) y la superior derecha tiene coordenadas (1000, 1000)

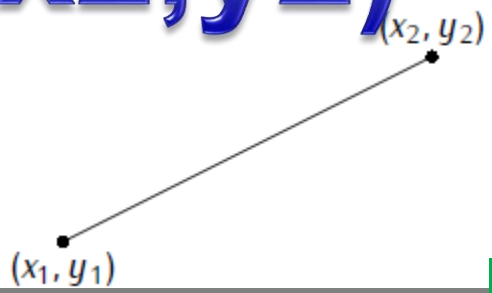


- Los recursos para graficar son:

create_point(x,y)

- **Nos permite dibujar un PUNTO en la coordenada x , y.**

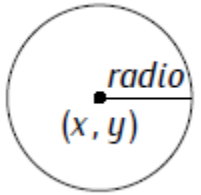
create_line(x1,y1,x2,y2)



- **Nos permite dibujar una LINEA entre los puntos (x1,y1) y (x2,y2).**

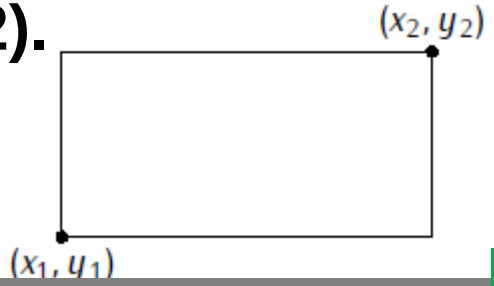
create_circle(x,y,radio)

- **Nos permite dibujar un CIRCULO en la coordenada x , y de radio «radio».**



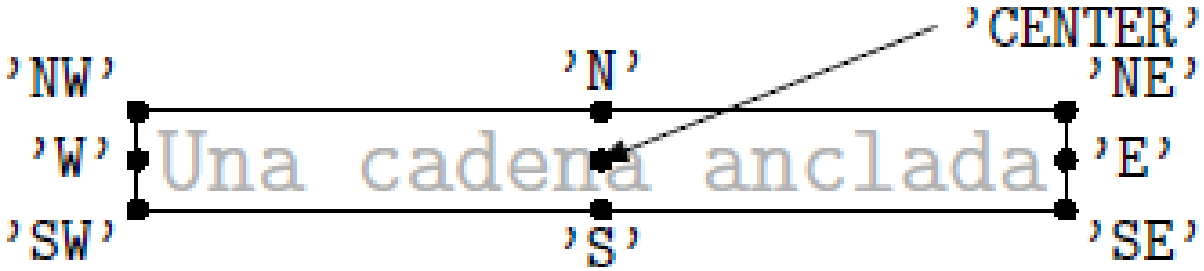
create_rectangle(x1,y1,x2,y2)

- **Nos permite dibujar un RECTANGULO con esquinas opuestas en los puntos (x1,y1) y (x2,y2).**



create_text(x,y,cadena,tamaño,anclaje)

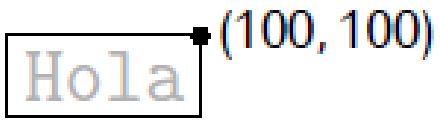
- **Nos permite dibujar el texto ‘cadena’ en el punto (x,y). El parámetro tamaño puede ser lógico entre 10 y 12 puntos. El ultimo parámetro anclaje indica si el texto se ‘ancla’a los distintos puntos del cuadro de texto.**

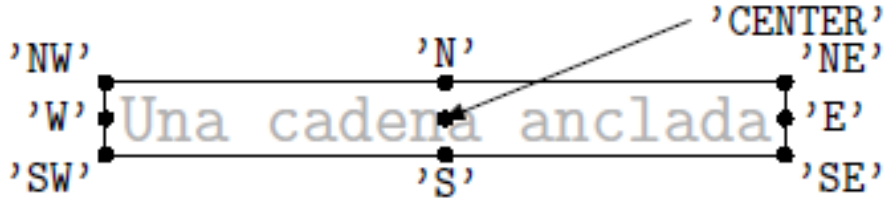


create_text(x,y,cadena,tamaño,anclaje)

- **Un ejemplo:**

create_text(100, 100, ‘hola’, 10, ‘NE’)





create_filled_circle(x,y,radio)

- **Nos permite dibujar un CIRCULO RELLENO (por defecto de color NEGRO) en la coordenada x , y de radio «radio».**

create_filled_rectangle(x1,y1,x2,y2)

- **Nos permite dibujar un RECTANGULO RELLENO (por defecto de color NEGRO) con esquinas opuestas en los puntos (x1,y1) y (x2,y2).**

erase()

- **Elimina todos los dibujos del lienzo**

Todas las funciones vistas aceptan un

PARAMETRO OPCIONAL MAS

Es el color. Por defecto es negro.

- **Un ejemplo:**

create_filled_rectangle(100, 100, 800, 800, ‘red’)

Repasando!...

- **Cambio de entorno interactivo a desarrollo de MI PROGRAMA.**
- **Nuestro primer programa….sin salida en pantalla?**
- **Sentencia PRINT.**
- **Entrada de informacion: raw_input()**
- **Mejoras con sentencias PRINT inicio y fin.**

Lo visto!

- **Salidas con FORMATO.**
- **Operador de formato: ‘cadena’ % (valor, valor, valor)**
- **Legibilidad de los programas.**
  - **Secciones/estructura.**
  - **Variables con nombre representativo**
  - **Etc.**
- **Comentarios, «#»**

- **GRAFICOS**

- **create_point()**
- **create_line()**
- **create_text()**
- **creat_circle()**
- **create_rectangle()**
- **create_filled_circle()**
- **create_filled_rectangle()**
- **`**
- **Mas opciones ver APENDICE B, libro MARZAL.**