rgrimson actualizo repo

1 contributor

Raw Blame

446 lines (319 sloc) 12.6 KB

Contenidos | Anterior (1 Python) | Próximo (3 Números)

1.2 Un primer programa

En esta sección vas a crear tu primer programa en Python, ejecutarlo y debuguearlo.

Ejecutando Python

Los programas en Python siempre son ejecutados en un intérprete de Python.

El intérprete es una aplicación que funciona en la consola y se ejecuta desde la terminal.

```
python3
Python 3.6.1 (v3.6.1:69c0db5050, Mar 21 2017, 01:21:04)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Les programadores no suelen tener problemas en usar el intérprete de esta forma, aunque no es la más cómoda para principiantes. Más adelante vamos a porponerles usar entornos de desarrollo más sofisticados, pero por el momento quedémosnos con la incomodidad que nos va a enseñar cosas útiles.

Modo interactivo

Cuando ejecutás Python, entrás al modo *interactivo* en el que podés experimentar.

Si escribís un comando, se va a ejecutar inmediatamente. No hay ningún ciclo de edición-compilación-ejecución-debug en esto, como en otros lenguajes.

```
>>> print('hello world')
hello world
>>> 37*42
1554
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
>>>
```

Esta forma de escribir código (en una consola del lenguaje) que se evalúa inmediatamente e imprime el resultado, se denomina *bucle de Lectura-Evaluación-Impresión* (REPL por las siglas en inglés de «Read-Eval-Print-Loop»). Asegurate de poder interactuar con el intérprete antes de seguir.

Veamos en mayor detalle cómo funciona este REPL:

- `>>>` es el símbolo del intérprete para comenzar un nuevo comando.
- `...` es el símbolo del intérprete para continuar con un comando comenzado antes. Dejá una línea en blanco para terminar lo que ya ingresaste.

El símbolo `...` puede mostrarse o no dependiendo de tu entorno. En este curso lo mostraremos como líneas en blanco para facilitar el copy-paste. de fragmentos de código.

Antes vimos que el guión bajo `_` guarda el último resultado.

```
>>> 37 * 42
1554
>>> _ * 2
3108
>>> _ + 50
```

```
3158
>>>
```

Esto solo es válido en el modo interactivo que estamos viendo. No uses el guión bajo en un programa.

Crear programas

Los programas se guardan en archivos `.py`.

```
# hello.py
print('hello world')
```

Podés crear estos archivos con tu editor de texto favorito. Más adelante vamos a proponerles usar el `spyder` que es un entorno de desarrollo integrado (IDE) que permite tener en la pantalla un editor y un intérprete al mismo tiempo, entre otras cosas. Pero por ahora usemos el block de notas, o el gedit para seguir con estos ejemplos.

Ejecutar programas

Para ejecutar un programa, correlo en la terminal con el comando `python` seguido del nombre del archivo a ejecutar. Por ejemplo, en una línea de comandos Unix (por ejemplo Ubuntu):

```
bash % python hello.py
hello world
bash %
```

O en una terminal de Windows:

```
C:\SomeFolder>hello.py
hello world

C:\SomeFolder>c:\python36\python hello.py
hello world
```

Obervación: En Windows puede ser necesario especificar el camino (path) completo al intérprete de Python como en `c:\python36\python`. Sin embargo, si Python está instalado del modo usual, podría alcanzar con que tipees el nombre del programa como en `hello.py`.

Un ejemplo de programa

Resolvamos el siguiente problema:

> Una mañana ponés un billete en la vereda al lado del obelisco porteño. A partir de ahí, cada día vas y duplicás la cantidad de billetes, apilándolos prolijamente. ¿Cuánto tiempo pasa antes de que la pila de billetes sea más alta que el obelisco?

Acá va una solución:

```
# obelisco.py
grosor_billete  = 0.11 * 0.001 # 0.11 mm escrito en metros
altura_obelisco = 67.5         # altura en metros
num_billetes    = 1
dia             = 1

while num_billetes * grosor_billete < altura_obelisco:
    print(dia, num_billetes, num_billetes * grosor_billete)
    dia = dia + 1
    num_billetes = num_billetes * 2

print('Cantidad de días', dia)
print('Cantidad de billetes', num_billetes)
print('Altura final', num_billetes * grosor_billete)
```

Cuando lo ejecutás, la salida será la siguiente:

```
bash % python3 obelisco.py
1 1 0.00011
2 2 0.00022
3 4 0.00044
4 8 0.00088
5 16 0.00176
6 32 0.00352
...
19 262144 28.83584
20 524288 57.67168
Cantidad de días 21
Cantidad de billetes 1048576
Altura final 115.34336
```

A continuación vamos a usar este primer programa como ejemplo para aprender algunas cosas fundamentales sobre Python.

Comandos

Un programa de Python es una secuencia de comandos:

```
a = 3 + 4
b = a * 2
print(b)
```

Cada comando se termina con una nueva línea. Los comandos son ejecutados uno luego del otro hasta que el intérprete llega al final del archivo.

Comentarios

Los comentarios son texto que no será ejecutado.

```
a = 3 + 4
# Esto es un comentario
b = a * 2
print(b)
```

Los comentarios comienzan con `#` y siguen hasta el final de la línea.

Variables

Una variable es un nombre para un valor. Estos nombres pueden estar formados por letras (minúsculas y mayúsculas) de la a a la z. También pueden incluir el guión bajo, y se pueden usar números, salvo como primer caracter.

```
altura = 442 # válido
_altura = 442 # válido
altura2 = 442 # válido
2altura = 442 # inválido
```

Tipos

El tipo de las variables no debe ser declarado como en otros lenguajes. El tipo es asociado con el valor del lado derecho.

```
height = 442           # Entero
height = 442.0         # Punto flotante
height = 'Really tall' # Cadena de caracteres
```

Decimos que Python tiene tipado dinámico. El tipo percibido por el intérprete puede cambiar a lo largo de la ejecución dependiendo el valor asignado a la variable.

Python distingue mayúsculas y minúsculas

Mayúsculas y minúsculas son diferentes para Python. Por ejemplo, todas las siguientes variables son diferentes.

```
name = 'David'
Name = 'Diego'
NAME = 'Rosita'
```

Los comandos de Python siempre se escriben con minúsculas.

```
while x < 0:   # OK
WHILE x < 0:   # ERROR
```

Ciclos

El comando `while` ejecuta un ciclo o *loop*.

```
while num_billetes * grosor_billete < altura_obelisco:
    print(dia, num_billetes, num_billetes * grosor_billete)
    dia = dia + 1
    num_billetes = num_billetes * 2

print('Cantidad de días', dia)
```

Los comandos indentados debajo del `while` se van a a ejecutar mientras que la expresión luego del `while` sea verdadera (`true`).

Indentación

La indentación se usa para marcar grupos de comandos que van juntos. Considerá el ejemplo anterior:

```
while num_billetes * grosor_billete < altura_obelisco:
    print(dia, num_billetes, num_billetes * grosor_billete)
    dia = dia + 1
    num_billetes = num_billetes * 2

print('Cantidad de días', dia)
```

La indentación agrupa los comandos siguientes como las operaciones a repetir:

```
    print(dia, num_billetes, num_billetes * grosor_billete)
    dia = dia + 1
    num_billetes = num_billetes * 2
```

Como el comando `print()` del final no está indentado, no pertenece al ciclo. La línea en blanco que dejamos entre ambos solo está para facilitar la lectura y no afecta la ejecución.

Indentando adecuadamente

Algunas recomendaciones sobre cómo indentar:

- Usá espacios y no el tabulador.
- Usá 4 espacios por cada nivel.
- Usá un editor de textos que entienda que estás escribiendo en Python.

El único requisito del intérprete de Python es que la indentación dentro de un mismo bloque sea consistente. Por ejemplo, esto es un error:

```
while num_billetes * grosor_billete < altura_obelisco:
    print(dia, num_billetes, num_billetes * grosor_billete)
        dia = dia + 1 # ERROR
    num_billetes = num_billetes * 2
```

Condicionales

El comando `if` es usado para ejecutar un condicional:

```
if a > b:
    print('Gana a')
else:
    print('Gana b')
```

Podés verificar múltiples condiciones agregando condiciones extras con `elif` .

```
if a > b:
    print('Gana a')
elif a == b:
    print('Empate!')
else:
    print('Gana b')
```

El comando `elif` viene de *else, if* y puede traducirse como "si no se da la condición del *if* anterior, verificá si se da la siguiente".

Imprimir en pantalla

La función `print` imprime una línea de texto con el valor pasado como parámetro.

```
print('Hello world!') # Imprime 'Hello world!'
```

Podés imprimir variables. El texto impreso en ese caso será el valor de la variable y no su nombre.

```
x = 100
print(x) # imprime el texto '100'
```

Si le pasás más de un valor al `print` los separa con espacios.

```
name = 'Juana'
print('Mi nombre es', name) # Imprime el texto 'Mi nombre es Juana'
```

`print()` siempre termina la línea impresa pasando a la sugiente.

```
print('Hola')
print('Mi nombre es', 'Juana')
```

This prints:

```
Hola
Mi nombre es Juana
```

El salto de línea entre ambos comandos puede ser suprimido:

```
print('Hola', end=' ')
print('Mi nombre es', 'Juana')
```

Este código va a imprimir:

```
Hola Mi nombre es Juana
```

Ingreso de valores por teclado

Para leer un valor ingresado por el usuario, usá la función `input()`:

```
name = input('Ingresá tu nombre:')
print('Tu nombre es', name)
```

`input` imprime el texto que le pases como parámetro y espera una respuesta. Es útil para programas pequeños, ejercitarse o para debuguear un código. Casi no se lo usa en programas reales.

El comando pass

A veces es conveniente especificar un bloque de código que no haga nada. El comando `pass` se usa para eso.

```
if a > b:
    pass
else:
    print('No ganó a')
```

Este comando no hace nada. Sirve para guardar el lugar para comando que querramos agregar luego.

Ejercicios

Ejercicio 1.4: Debuguear

El siguiente fragmento de código está relacionado con el problema del obelisco. Tiene un bug, es decir, un error.

```
# obelisco.py

grosor_billete  = 0.11 * 0.001 # 0.11 mm en metros
altura_obelisco = 67.5         # altura en metros
num_billetes    = 1
dia             = 1

while num_billetes * grosor_billete < altura_obelisco:
    print(dia, num_billetes, num_billetes * grosor_billete)
    dia = dias + 1
    num_billetes = num_billetes * 2

print('Cantidad de días', dia)
print('Cantidad de billetes', num_billetes)
print('Altura final', num_billetes * grosor_billete)
```

Copiá y pegá el código que aparece arriba en un nuevo archivo llamado `obelisco.py` . Cuando ejecutes el código vas a obtener el siguiente mensaje de error que hace que el programa se detenga:

```
Traceback (most recent call last):
  File "obelisco.py", line 10, in <module>
```

```
    dia = dias + 1
NameError: name 'dias' is not defined
```

Aprender a leer y entender los mensajes de error es una parte fundamental de programar en Python. Si tu programa *crashea* (se rompe, da error) la última línea del mensaje de error indica el motivo. Sobre eso, vas a ver un fragmento de código, un nombre de archivo y un número de línea que identifican el problema.

- ¿En qué linea está el error?
- ¿Cuál es el error?
- Repará el error.
- Ejecutá el programa exitosamente.

Ejercicio 1.5: La pelota que rebota

Este es el primer conjunto de ejercicios en el que vas a tener que crear un archivo de Python y correrlo. A partir de aca, vamos a asumir que estás trabajando en el subdirectorio `Ejercicios/`. Para ayudarte a ubicar el lugar correcto ya creamos un par de archivos en ese directorio. Por ejemplo, ahora buscá el archivo `Ejercicios/rebotes.py` que vamos a usar en este ejercicio.

Una pelota de goma es arrojada desde una altura de 100 metros y cada vez que toca el piso salta 3/5 de la altura desde la que cayó. Escribí un programa `rebotes.py` que imprima una tabla mostrando las alturas que alcanza luego de sus primeros diez rebotes.

Tu programa debería hacer una tabla que se parezca a esta:

```
1 60.0
2 36.0
3 21.599999999999998
4 12.959999999999999
5 7.775999999999999
6 4.6655999999999995
7 2.7993599999999996
8 1.6796159999999998
9 1.0077695999999998
10 0.6046617599999998
```

Nota: Podés limpiar un toque la salida si usás la función round() de la que miraste el help hace un rato. Tratá de usarla para redondear a cuatro dígitos.

```
1 60.0
2 36.0
3 21.6
4 12.96
```

```
5 7.776
6 4.6656
7 2.7994
8 1.6796
9 1.0078
10 0.6047
```

Ejercicio 1.6: Saludos

Escribí un programa llamado `saludo.py` que pregunte el nombre de le usuarie, imprima un saludo (por ejemplo, "Hola, Juana") y termine.

Contenidos | Anterior (1 Python) | Próximo (3 Números)