Python 3

> Hello, World!

Acerca de Python

Python es un lenguaje de programación de alto nivel. Fue desarrollado durante los finales de los 80s por Guido van Rossum.

La primer versión de Python fue publicada en 1991 y su versión 1.0 fue lanzada en 1994.

Implementaciones de Python

Existen distintas implementaciones del intérprete de Python. Cuando la gente habla de Python en general se refieren a CPython (la implementación de referencia de Python). Esta implementación está hecha en C.

Hay otras implementaciones como PyPy, Jython, IronPython, PythonNet, etc. Cada implementación tiene su propio versionado y el código fuente de un script de Python puede no ser intercompatible entre distintas implementaciones del intérprete.

Estado actual

Python 2.0 fue lanzado en el 2000. La última versión de Python 2, la versión 2.7, fue lanzada en el 2010.

Python 2 va a alcanzar EOL el 1/1/2020.

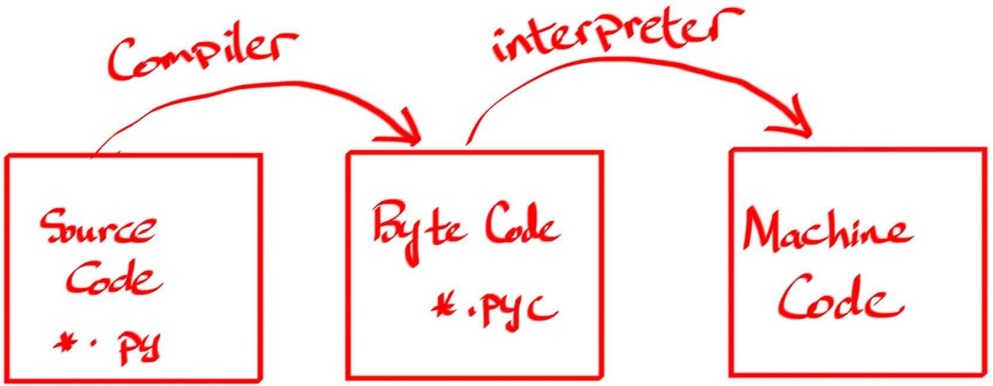
Python 3.0 fue lanzado a finales del 2008.

El último lanzamiento oficial de Python se realizó el 14/10/2019 y corresponde a la versión 3.8.

El interprete

Python es un lenguaje de programación interpretado. Esto quiere decir que su código no es convertido a código máquina.

En CPython el source code es convertido en byte code. Este bytecode e



Dos formas de correr al intérprete

Immediate mode

Es un REPL (read-print-eval-loop).

```
>>> 2 + 2
4
>>> print('Hello, World!')
Hello, World!
>>> 'Hello, World!'
'Hello, World!'
>>> type('Hello, World!')
<class 'str'>
```

Script mode

El programa es escrito en un archivo (script) y este es ejecutado por el intérprete.

```
noone@ms01:~$ python3.7 hello_world.py
Hello, World!
```

Tipado

Fuerte

Python es un lenguaje de tipado fuerte. Las operaciones entre valores de distinto tipo necesitan de conversiones explícitas.

```
>>> 'a' + 'b'
'ab'
>>> 'a' + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>> 'a' + str(1)
'a1'
>>>
```

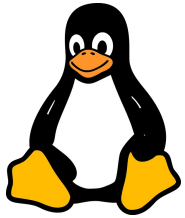
Dinámico

En Python los valores tienen un tipo, pero las variables no. En los lenguajes de tipado estático las variables son de un tipo definido.

```
>>> a = 1
>>> a
1
>>> a = 'one'
>>> a
'one'
```

Instalación

Linux

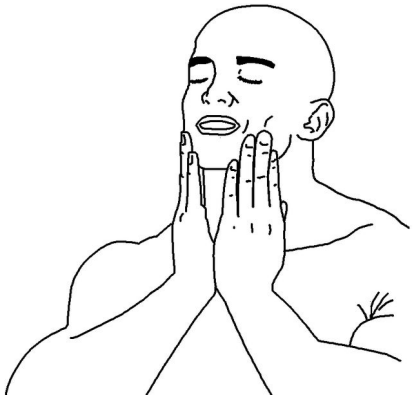


Python 3 ya viene preinstalado en muchas distros de Linux.

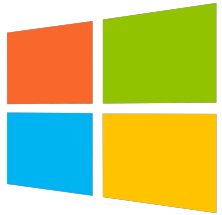
Si Python no se encuentra instalado, la forma más fácil de conseguirlo es a través de un gestor de paquetes.

Por ejemplo en Ubuntu Python 3 se puede instalar así:

```
$ sudo apt-get install python3
```



FuckOS



1. Descargue el instalador de Python 3 para Windows.

2. Ejecute el instalador.

3. Comience a rezar. La aberración que tiene instalada como sistema operativo necesita de toda la ayuda posible.

Python 101

Variables

Las variables son creadas utilizando la sintaxis

<nombre> = <valor>

Donde <nombre> puede contener letras, dígitos y guiones bajos, pero no puede comenzar con un dígito.

Las palabras reservadas del lenguaje no pueden ser utilizadas como nombres de variables.

Operadores

El operador `+` puede ser usado para concatenar secuencias (como strings o listas) o sumar valores numericos.

El operador `*` puede multiplicar valores numéricos o repetir secuencias.

Atajos como `a += 5` están disponibles en el lenguaje.

Los operadores de post y pre decremento no están disponibles.

Operator	Description
`:=`	Assignment expression
`lambda`	Lambda expression
`if – else`	Conditional expression
`or`	Boolean OR
`and`	Boolean AND
`not x`	Boolean NOT
`in`, `not in`, `is`, `is not`, `<`, `<=`, `>`, `>=`, `!=`, `==`	Comparisons, including membership tests and identity tests
`|`	Bitwise OR
`^`	Bitwise XOR
`&`	Bitwise AND
`<<`, `>>`	Shifts
`+`, `-`	Addition and subtraction
`*`, `@`, `/`, `//`, `%`	Multiplication, matrix multiplication, division, floor division, remainder [5]
`+x`, `-x`, `~x`	Positive, negative, bitwise NOT
`**`	Exponentiation [6]
`await x`	Await expression
`x[index]`, `x[index:index]`, `x(arguments...)`, `x.attribute`	Subscription, slicing, call, attribute reference
`(expressions...)`, `[expressions...]`, `{key: value...}`, `{expressions...}`	Binding or parenthesized expression, list display, dictionary display, set display

Precedencia de operadores

Strings

En Python un valor de tipo string puede declararse con comillas dobles o simples.

La única diferencia técnica es que dentro de un string con comillas dobles, el carácter de la comilla simple es válido, y viceversa.

El carácter de escape en Python es \.

“Hello\n” “Mi nombre es ‘Pedro’” ‘Mi nombre es “Pedro”’

Sentencias e indentación

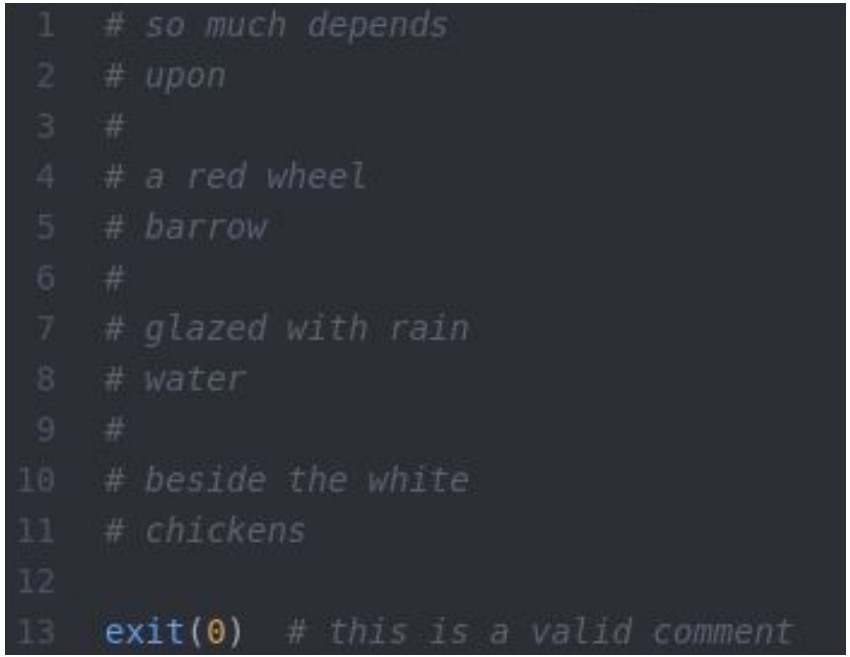
Cada fin de línea de código es interpretado como el final de una sentencia.

Los bloques de código (cuerpos de funciones, loop bodies, etc) son determinados por la indentación de sus líneas. Es decir la cantidad de tabs o espacios en blanco hasta su primer carácter.

Comentarios

El carácter # se utiliza para realizar comentarios dentro del código fuente.

En Python no existen los comentarios de múltiples líneas, a diferencia de lenguajes como C++ o PHP.



STD IO

La función `print()` puede ser utilizada para escribir strings en la consola.

La función `input()` espera a que el usuario ingrese una línea de texto en la consola y retorna esa misma línea como un string.

```
>>> print(':D')
:D
>>> a = input('--> ')
--> (¬_¬)
>>> print(a)
(¬_¬)
```

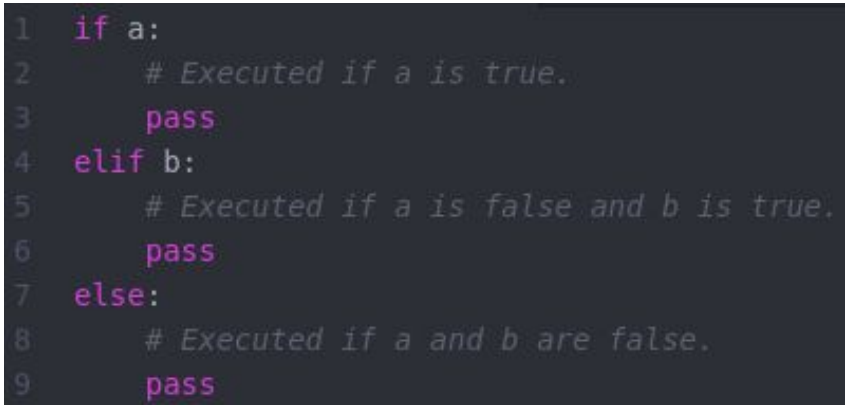
Algunas constantes importantes

`True` y `False` son constantes que representan los únicos valores posibles del tipo booleano.

`None` es una constante que generalmente se utiliza para representar la ausencia de un valor determinado.

```
>>> a = False
>>> b = True
>>> a and b
False
>>> a or b
True
>>> c = None
>>> bool(None)
False
```

if-elif-else



```
if a:
    # Executed if a is true.
    pass
elif b:
    # Executed if a is false and b is true.
    pass
else:
    # Executed if a and b are false.
    pass
```

Si `a` no fuese de tipo bool, la condición del `if` será evaluada como `bool(a)` (cast a tipo booleano).

Las cláusulas `elif` y `else` son opcionales.

En Python las funciones, sentencias if y loops tienen que tener un cuerpo de forma obligatoria. Si no se desea ejecutar ningún cómputo en algún cuerpo se puede utilizar la sentencia `pass`.

Equivalentes booleanos

Estos son los casos en los que el método `bool()` devuelve `False`. En cualquier otro caso devuelve `True`.

- Si se le pasa `False`.
- Se se le pasa `None`.
- Si se le pasa una secuencia vacía (`''`, `[]`, `()`).
- Si se le pasa un valor numérico equivalente a `0`.
- Si se le pasa un diccionario vacío (`{}`).
- Si se le pasa un objeto con un método `__bool__()` que devuelve `False` o un método `__len__()` que devuelve `0`.

Ranges

El tipo range representa una secuencia inmutable de números que es comúnmente usada para construir loops del tipo for.

Este tipo tiene los siguientes constructores:

range(stop)

range(start, stop[, step])

```
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> list(range(2, 5))
[2, 3, 4]
>>> list(range(2, 15, 3))
[2, 5, 8, 11, 14]
>>> list(range(0, -5))
[]
>>> list(range(0, -5, -1))
[0, -1, -2, -3, -4]
```

Loops (I)

---

for

```python
for i in range(10):
    if i % 2:
        print(str(i) + ' is an odd number.')
    else:
        print(str(i) + ' is an even number.')
```

El loop for itera sobre una lista de valores.

while

```python
i = 0
while i < 10:
    if i % 2:
        print(str(i) + ' is an odd number.')
    else:
        print(str(i) + ' is an even number.')
    i += 1
```

En Python no existe el loop do-while.

Las sentencias break y continue son reconocidas y operan igual que en lenguajes como C.

Loops (II)

Los loops `for` pueden iterar sobre cualquier iterable, por ejemplo strings, listas, diccionarios, etc.

```
for char in 'this is a string':
    print(char)


for word in ['this', 'is', 'a', 'list']:
    print(word)
```

A los loops `while` y `for` se les puede agregar una cláusula `else` que será ejecutada al finalizar exitosamente el loop (si se sale del loop con `break` la cláusula `else` no se ejecuta)

```
i = 0
while i < 10:
    print(i)
    i += 1
else:
    print('done')
```

code()

Ejercicios 1

- Instalar y correr Python 3.6/3.7/3.8.
- Escribir un programa que imprima:

  ```
  One of the months of the year is January
  One of the months of the year is February
  ...
  ```

- Escribir un programa que imprima los primeros n números triangulares (googlear que es un número triangular).
- Escribir un programa que cuente los dígitos pares de un número.

Bibliografía

- Python docs
- The Hitchhiker’s Guide to Python!
- How to Think Like a Computer Scientist: Learning with Python