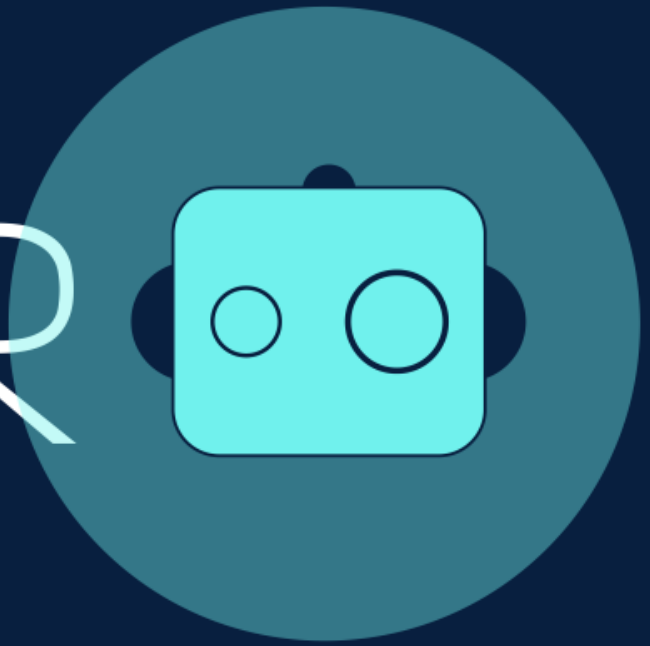
UMAKER



PYTHON

Christian Quispe Canchari

¿Por qué la gente usa Python?

- En los últimos años han surgido algunos temas comunes. Los principales factores citados por los usuarios de Python parecen ser estos :

1. Calidad del software
2. Productividad del desarrollador
3. Portabilidad del programa
4. Bibliotecas de soporte
5. Integración de componentes
6. Disfrute

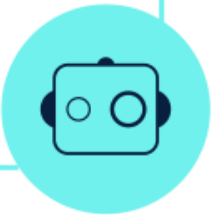
- ¿Es Python un “Lenguaje de script” ?

De echo la gente a menudo usa la palabra “script” en lugar de “programa” para describir un archivo de código Python.

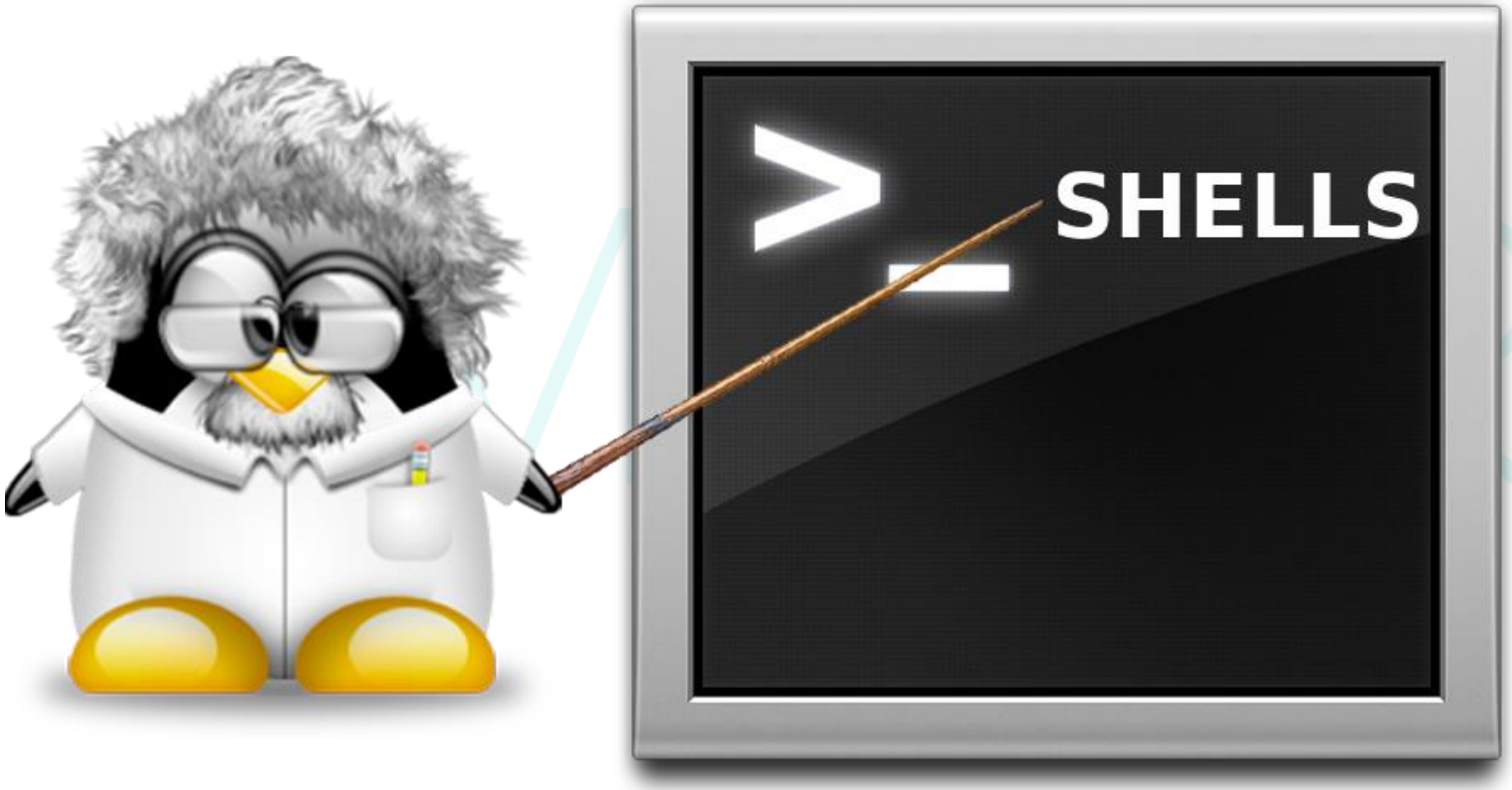
Los términos "script y "programa" se usan indistintamente, con una ligera preferencia por "script" para describir un archivo de nivel superior más simple y "programa" para referirse a una aplicación más sofisticada.

Porque el término "lenguaje de script" tiene muchos significados diferentes para diferentes observadores, algunos preferirían que no se aplique a Python en absoluto. De hecho, las personas tienden para hacer tres asociaciones muy diferentes, algunas de las cuales son más útiles que otras,

Cuando se escucha de Python se entiende como:

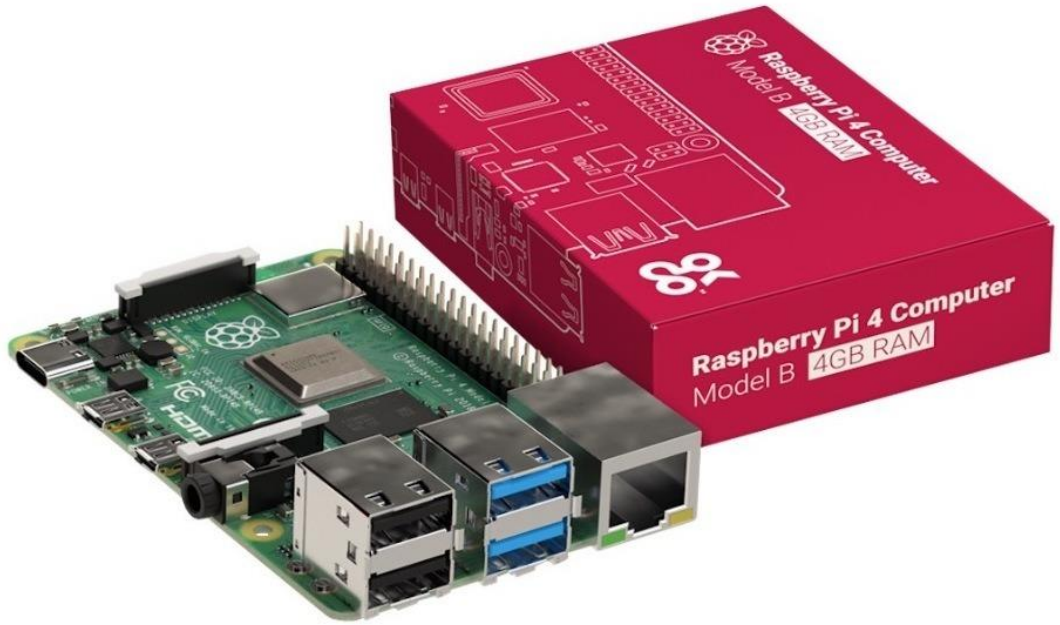


Herramientas de Shell



Lenguaje de control

Los programas de Python a menudo se implementan en el contexto de aplicaciones más grandes. Por ejemplo, para probar dispositivos de hardware, los programas Python pueden llamar a componentes que dan acceso de bajo nivel a un dispositivo.



Facilidad de uso

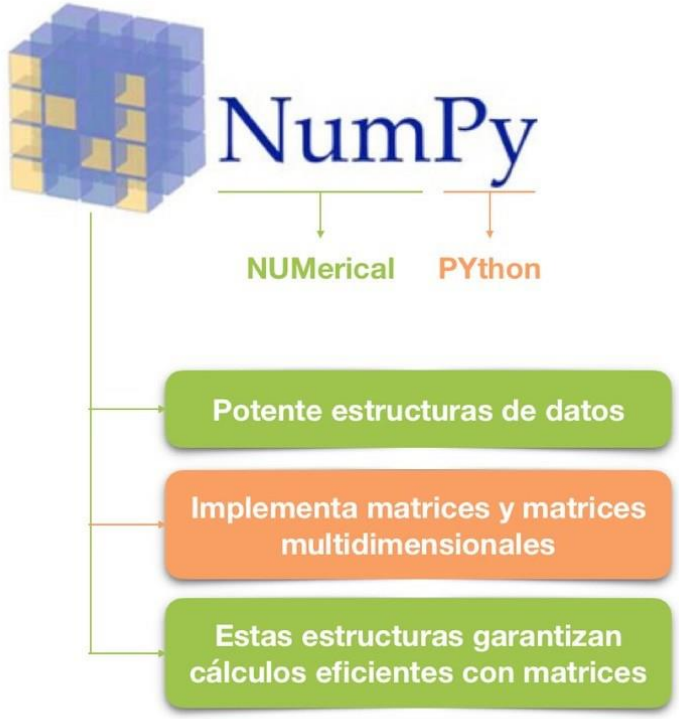
Probablemente la mejor manera de pensar en el término "lenguaje de secuencias de comandos" es que se refiere a un lenguaje simple utilizado para codificar rápidamente las tareas. Esto es especialmente cierto cuando el término se aplica a Python, lo que permite un desarrollo de programa mucho más rápido que lenguajes compilados como C ++.

```
yo = Persona()

while   yo.vivo

    yo.despertar()
    yo.comer()
    yo.programar()
    yo.dormir()
```

Entonces, ¿Python es un lenguaje de script o no?
Depende de a quién le preguntes. En general, el
El término "scripting" probablemente se usa mejor para describir el modo de desarrollo rápido y flexible que admite Python, en lugar de un dominio de aplicación particular

Si alguna vez le importará la diferencia de velocidad de ejecución depende de qué tipo de programas que escribes. Python ha sido optimizado en numerosas ocasiones, y el código Python funciona lo suficientemente rápido por sí mismo en la mayoría de los dominios de aplicación

No hablaremos mucho sobre extensiones en esta sesion, pero podemos nombrar un excelente ejemplo de este lenguaje es la extensión de programación numérica NumPy para Python; combinando bibliotecas de extensiones numéricas compiladas y optimizadas con el lenguaje Python,
NumPy convierte Python en una herramienta de programación numérica que es eficiente y fácil de usar.

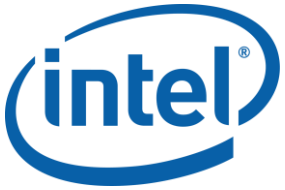


¿Quién usa Python hoy?

Al momento de escribir esto, la mejor estimación que cualquiera puede hacer del tamaño de Python

La base de usuarios es que actualmente hay aproximadamente 1 millón de usuarios de Python en todo el mundo (más o menos). Esta estimación se basa en varias estadísticas, como las tasas de descarga y Encuestas de desarrolladores. Debido a que Python es de código abiert

- Google hace un uso extensivo de Python en sus sistemas de búsqueda web y emplea.
- El servicio para compartir videos de YouTube está escrito principalmente en Python.
- El popular sistema de intercambio de archivos punto a punto BitTorrent es un programa Python.
- EVE Online, un juego en línea multijugador masivo (MMOG), hace un uso extensivo de pitón.
- Maya, un potente sistema integrado de modelado y animación en 3D, proporciona un API de secuencias de comandos de Python.
- Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm e IBM usan Python para las pruebas de hardware.




INDUSTRIAL
LIGHT & MAGIC
CREATING
THE IMPOSSIBLE


PIXAR

• JPMorgan Chase, UBS, Getco y Citadel aplican Python para el mercado financiero previsión.

- NASA, Los Alamos, Fermilab, JPL y otros usan Python para tareas de programación científica.
- iRobot usa Python para desarrollar dispositivos robóticos comerciales.
- ESRI usa Python como una herramienta de personalización del usuario final para su popular mapeo GIS productos
- La NSA utiliza Python para la criptografía y el análisis de inteligencia.
- El producto del servidor de correo electrónico IronPort usa más de 1 millón de líneas de código Python para hacer su trabajo
- El proyecto One Laptop Per Child (OLPC) construye su interfaz de usuario y actividad modelo en Python.

Y así. Probablemente el único hilo común entre las compañías que usan Pythonhoy es que Python se usa en todo el mapa, en términos de dominios de aplicación. Su naturaleza de uso general lo hace aplicable a casi todos los campos, no solo a uno. De hecho, es seguro decir que prácticamente todas las organizaciones importantes que escriben software utilizan Python,ya sea para tareas tácticas a corto plazo, como pruebas y administración, o para el desarrollo estratégico de productos a largo plazo. Python ha demostrado funcionar bien en ambos modos.


CLI
GUI

Cómo Python ejecuta programas

- Presentamos al intérprete de Python
- Hasta ahora, he estado hablando principalmente de Python como lenguaje de programación. Pero, como se implementa actualmente, también es un paquete de software llamado intérprete. Un intérprete es
- Un tipo de programa que ejecuta otros programas. Cuando escribes un programa Python,
- El intérprete de Python lee su programa y lleva a cabo las instrucciones que contiene.
- En efecto, el intérprete es una capa de lógica de software entre su código y la computadora
- hardware en su máquina.
- Cuando el paquete Python se instala en su máquina, genera una serie de componentes, como mínimo, un intérprete y una biblioteca de soporte. Dependiendo de cómo uses
- el intérprete de Python puede tomar la forma de un programa ejecutable o un conjunto de
- bibliotecas vinculadas a otro programa. Dependiendo de qué sabor de Python ejecutes,
- el propio intérprete puede implementarse como un programa en C, un conjunto de clases Java o
- algo más. Cualquiera sea la forma que tome, el código de Python que escriba siempre debe ejecutarse
- por este intérprete Y para habilitar eso, debe instalar un intérprete de Python en su
- computadora.

Ejecución del programa Lo que significa escribir y ejecutar un script de Python depende de si observa estos tareas como programador o como intérprete de Python. Ambas vistas ofrecen perspectivas importantes sobre la programación de Python.

La vista del programador En su forma más simple, un programa de Python es solo un archivo de texto que contiene declaraciones de Python. Por ejemplo, el siguiente archivo, llamado script0.py, es uno de los scripts Python más simples. Podría soñar, pero pasa por un programa Python completamente funcional:

```
print('hello world')
print(2 ** 100)
```

Este archivo contiene dos declaraciones de impresión de Python, que simplemente imprimen una cadena (el texto en comillas) y un resultado de expresión numérica (2 a la potencia 100) a la secuencia de salida.
No se preocupe aún por la sintaxis de este código; para este capítulo, solo nos interesa para que funciona

Compilación de código de bytes

Internamente, y casi completamente oculto para ti, cuando ejecutas un programa Python primero compila su código fuente (las declaraciones en su archivo) en un formato conocido
como código de byte. La compilación es simplemente un paso de traducción, y el código de bytes es un nivel inferior,
Python traduce cada una de sus declaraciones de origen en un grupo de instrucciones de código de bytes descomponiendo ellos en pasos individuales. Esta traducción de código de bytes se realiza para acelerar ejecución: el código de bytes se puede ejecutar mucho más rápido que el código fuente original.
á estos archivos aparecer en su computadora después de ejecutar algunos

Python almacenará el código de bytes de sus programas en archivos que terminan con una extensión .pyc (".pyc" significa ".py" compilado) Python guarda código de bytes como este como una optimización de velocidad de inicio. La próxima vez que corras su programa, Python cargará los archivos .pyc y omitirá el paso de compilación, siempre que no ha cambiado su código fuente desde la última vez que se guardó el código de bytes.

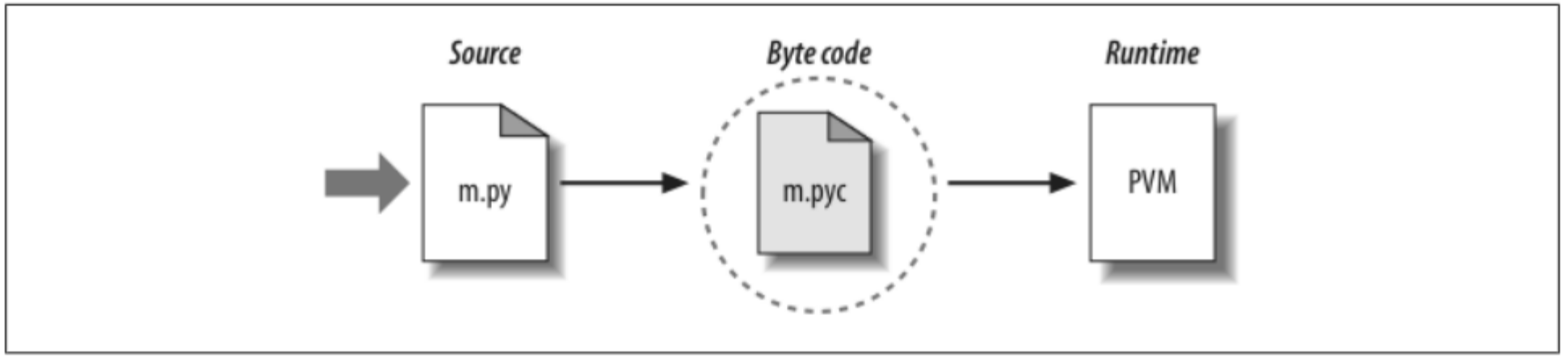


Figura 1. El modelo en tiempo de ejecución tradicional de Python: el código fuente que se traduce a código byte, que luego ejecuta Python Virtual Machine. Su código se compila automáticamente, pero luego Es interpretado.

Comenzando

Instalar Python

Muchas PC y Mac ya tendrán Python instalado. Para verificar si tiene Python instalado en una PC con Windows, busque Python en la barra de inicio o ejecute lo siguiente en la Línea de comandos (cmd.exe):

```
C:\Users\Christian Quispe>python --version
```

Para verificar si tiene Python instalado en Linux o Mac, luego en Linux abra la línea de comando o en Mac abra la Terminal y escriba:

```
kuse@DESKTOP-V1HDASV:~$ python --version
```

Si no tiene Python instalado en su computadora, puede descargarlo de forma gratuita desde el siguiente sitio web: https://www.python.org/

Python es un lenguaje de programación interpretado, esto significa que, como desarrollador, usted escribe archivos Python (.py) en un editor de texto y luego coloca esos archivos en el intérprete de Python para su ejecución.

La forma de ejecutar un archivo de Python es así en la línea de comando:

```
C:\Users\Christian Quispe>python helloworld.py
```

Donde "helloworld.py" es el nombre de su archivo de Python.

Escribamos nuestro primer archivo de Python, llamado helloworld.py, que se puede hacer en cualquier editor de texto.

helloworld.py

```
print("Hola mundo")
```

Simple como eso. Guarda tu archivo. Abra su línea de comando, navegue hasta el directorio donde guardó su archivo y ejecute:

```
C:\Users\Christian Quispe>python helloworld.py
```

La salida debería leer:

```
Hola mundo
```

Felicitaciones, ha escrito y ejecutado su primer programa Python.

La línea de comando de Python

Para probar una pequeña cantidad de código en Python, a veces es más rápido y fácil no escribir el código en un archivo. Esto es posible porque Python se puede ejecutar como una línea de comando en sí.

Escriba lo siguiente en la línea de comandos de Windows, Mac o Linux:

```
C:\Users\Christian Quispe>python
```

O, si el comando "python" no funcionó, puede probar "py":

```
C:\Users\Christian Quispe>py
```

Variables de Python

En Python, las variables se crean cuando le asigna un valor:

```
x = 5
y = "Hello, World!"
```

Python no tiene comando para declarar una variable.

```
x = 5
y = "John"
print(x)
print(y)
```

No es necesario declarar las variables con ningún tipo en particular e incluso puede cambiar el tipo después de que se hayan establecido.

```
x = 4 # x is of type int
x = "Sally" # x is now of type str
print(x)
```

Las variables de cadena se pueden declarar mediante comillas simples o dobles:

```
x = "John"
# is the same as
x = 'John'
```

Nombres Variables

Una variable puede tener un nombre corto (como x e y) o un nombre más descriptivo (edad, nombre del automóvil, volumen_total). Reglas para las variables de Python:

- Un nombre de variable debe comenzar con una letra o el carácter de subrayado
- Un nombre de variable no puede comenzar con un número
- Un nombre de variable solo puede contener caracteres alfanuméricos y guiones bajos (Az, 0-9 y _)
- Los nombres de las variables distinguen entre mayúsculas y minúsculas (edad, edad y EDAD son tres variables diferentes)

```
#Legal variable names:
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"

#Illegal variable names:
2myvar = "John"
my-var = "John"
my var = "John"
```

Recuerde que los nombres de las variables distinguen entre mayúsculas y minúsculas

Asignar valor a múltiples variables

Python le permite asignar valores a múltiples variables en una línea:

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

Y puede asignar el *mismo* valor a múltiples variables en una línea:

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

Variables de salida

La declaración “print” de Python se usa a menudo para generar variables.
Para combinar texto y una variable, Python usa el “+”carácter:

```
x = "awesome"
print("Python is " + x)
```

También puede usar el ”+”carácter para agregar una variable a otra variable:

```
x = "Python is "
y = "awesome"
z =  x + y
print(z)
```

Para los números, el + funciona como un operador matemático:

```
x = 5
y = 10
print(x + y)
```

Si intentas combinar una cadena y un número, Python te dará un error

Variables globales

Las variables que se crean fuera de una función (como en todos los ejemplos anteriores) se conocen como variables globales.
Las variables globales pueden ser utilizadas por todos, tanto dentro de las funciones como fuera.

Ejemplo
Cree una variable fuera de una función y úsela dentro de la función

```
x = "awesome"

def myfunc():
  print("Python is " + x)

myfunc()
```

Si crea una variable con el mismo nombre dentro de una función, esta variable será local y solo puede usarse dentro de la función. La variable global con el mismo nombre permanecerá como estaba, global y con el valor original.

Cree una variable dentro de una función, con el mismo nombre que la variable global

```
x = "awesome"

def myfunc():
  x = "fantastic"
  print("Python is " + x)

myfunc()

print("Python is " + x)
```

La palabra clave global

Normalmente, cuando crea una variable dentro de una función, esa variable es local y solo puede usarse dentro de esa función.

Para crear una variable global dentro de una función, puede usar la palabra clave *global*.

Ejemplo

Si usa la palabra clave *global*, la variable pertenece al alcance global:

```
def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

Además, use la palabra clave *global* si desea cambiar una variable global dentro de una función.

Ejemplo

Para cambiar el valor de una variable global dentro de una función, consulte la variable usando la palabra clave global:

```
x = "awesome"

def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

Tipos de datos de Python

Tipos de datos incorporados

En programación, el tipo de datos es un concepto importante.
Las variables pueden almacenar datos de diferentes tipos, y diferentes tipos pueden hacer cosas diferentes.
Python tiene los siguientes tipos de datos integrados de manera predeterminada, en estas categorías:

Tipo de texto:	`str`
Tipos numéricos:	`int` ` `float` ` `complex`
Tipos de secuencia:	`list` ` `tuple` ` `range`
Tipo de mapeo:	`dict`
Establecer tipos:	`set` , `frozenset`
Tipo booleano:	`bool`
Tipos binarios:	`bytes` ` `bytearray` ` `memoryview`

Obtener el tipo de datos

Puede obtener el tipo de datos de cualquier objeto utilizando la función *type()* :

Imprima el tipo de datos de la variable x:

```
x = 5
print(type(x))
```

Establecer el tipo de datos

En Python, el tipo de datos se establece cuando asigna un valor a una variable:

`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool
`x = b"Hello"`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview

Establecer el tipo de datos específico

Si desea especificar el tipo de datos, puede usar las siguientes funciones de constructor:

`x = str("Hello World")`	str
`x = int(20)`	int
`x = float(20.5)`	float
`x = complex(1j)`	complex
`x = list(("apple", "banana", "cherry"))`	list
`x = tuple(("apple", "banana", "cherry"))`	tuple
`x = range(6)`	range
`x = dict(name="John", age=36)`	dict
`x = set(("apple", "banana", "cherry"))`	set
`x = frozenset(("apple", "banana", "cherry"))`	frozenset
`x = bool(5)`	bool
`x = bytes(5)`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview

Números de Python

Hay tres tipos numéricos en Python:

- En t
- flotador
- complejo

Las variables de los tipos numéricos se crean cuando les asigna un valor:

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

Para verificar el tipo de cualquier objeto en Python, use la `type()`función:

```
print(type(x))
print(type(y))
print(type(z))
```

Entero

Int, o entero, es un número entero, positivo o negativo, sin decimales, de longitud

```
x = 1
y = 35656222554887711
z = -3255522

print(type(x))
print(type(y))
print(type(z))
```

Flotador

Flotante o "número de coma flotante" es un número, positivo o negativo, que contiene uno o más decimales.

```
x = 1.10
y = 1.0
z = -35.59

print(type(x))
print(type(y))
print(type(z))
```

El flotador también puede ser números científicos con una "e" para indicar el poder de 10.

```
x = 35e3
y = 12E4
z = -87.7e100

print(type(x))
print(type(y))
print(type(z))
```

Complejo

Los números complejos se escriben con una "j" como parte imaginaria:

```
x = 3+5j
y = 5j
z = -5j

print(type(x))
print(type(y))
print(type(z))
```

Conversión de tipo

Puede convertir de un tipo a otro con el int(), float()y complex()métodos:

```
x = 1 # int
y = 2.8 # float
z = 1j # complex

#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)

#convert from int to complex:
c = complex(x)

print(a)
print(b)
print(c)

print(type(a))
print(type(b))
print(type(c))
```

No puede convertir números complejos en otro tipo de número.

Número aleatorio
Python no tiene una función ramdon () para hacer un número aleatorio, pero Python tiene un módulo incorporado llamado randomque se puede usar para hacer números aleatorios:

```
import random

print(random.randrange(1,10))
```

Python Casting

Especificar un tipo de variable
Puede haber ocasiones en las que desee especificar un tipo en una variable. Esto se puede hacer con un casteo. Python es un lenguaje orientado a objetos y, como tal, utiliza clases para definir tipos de datos, incluidos sus tipos primitivos.

Por lo tanto, la conversión en python se realiza utilizando funciones de constructor:

`int ()` : construye un número entero a partir de un literal entero, un literal flotante (redondeando al número entero anterior) o un literal de cadena (siempre que la cadena represente un número entero)

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

`float ()` : construye un número flotante a partir de un literal entero, un literal flotante o un literal de cadena (siempre que la cadena represente un flotante o un entero)

```
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
```

str () : construye una cadena a partir de una amplia variedad de tipos de datos, incluidas cadenas, literales enteros y literales flotantes

```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

String en Python

Los literales de cadena en Python están rodeados por comillas simples o comillas dobles.
'hola' es lo mismo que "hola"

```
print("Hello")
print('Hello')
```

Asignar un String a una variable
La asignación de una cadena a una variable se realiza con el nombre de la variable

```
a = "Hello"
print(a)
```

Cuerdas Multilínea

Puede asignar una cadena multilínea a una variable utilizando tres comillas:

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

Las cadenas son matrices

Al igual que muchos otros lenguajes de programación populares, las cadenas en Python son matrices de bytes que representan caracteres unicode.

Sin embargo, Python no tiene un tipo de datos de caracteres, un solo carácter es simplemente una cadena con una longitud de 1.

Los corchetes se pueden usar para acceder a elementos de la cadena.

```
a = "Hello, World!"
print(a[1])
```

Rebanar

Puede devolver un rango de caracteres utilizando la sintaxis de corte.

Especifique el índice inicial y el índice final, separados por dos puntos, para devolver una parte de la cadena.

```
b = "Hello, World!"
print(b[2:5])
```

Indexación Negativa

Use índices negativos para comenzar el corte desde el final de la cadena:

Ejemplo

Obtenga los caracteres de la posición 5 a la posición 1, comenzando el conteo desde el final la cadena:

```
b = "Hello, World!"
print(b[-5:-2])
```

Longitud de un String

Para obtener la longitud de una cadena, use la len()función.

Ejemplo

La len()función devuelve la longitud de una cadena:

```
a = "Hello, World!"
print(len(a))
```

Métodos de un String

Python tiene un conjunto de métodos integrados que puede usar en cadenas.

Valores booleanos

En la programación, a menudo necesita saber si una expresión es True o False.

Puede evaluar cualquier expresión en Python y obtener una de dos respuestas, True o False.

Cuando compara dos valores, se evalúa la expresión y Python devuelve la respuesta booleana:

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

Cuando ejecuta una condición en una instrucción if, Python devuelve Trueo False:

```
a = 200
b = 33

if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Evaluar valores y variables

La bool()función le permite evaluar cualquier valor y darle Trueo, False a cambio,

```
print(bool("Hello"))
print(bool(15))
```

Ejemplo

Evaluar dos variables:

```
x = "Hello"
y = 15

print(bool(x))
print(bool(y))
```

La mayoría de los valores son verdaderos

Casi cualquier valor se evalúa Truesi tiene algún tipo de contenido.

Cualquier cadena es True, excepto las cadenas vacías.

Cualquier número es True, excepto 0.

Cualquier lista, tupla, conjunto y diccionario son True, excepto los vacíos.

Ejemplo

Lo siguiente devolverá True:

```
bool("abc")
bool(123)
bool(["apple", "cherry", "banana"])
```

Algunos valores son falsos

De hecho, no hay muchos valores que se evalúa False, a excepción de los valores vacíos, tales como (), [], {}, "", el número 0y el valor None. Y, por supuesto, el valor se Falseevalúa como False.

Ejemplo

Lo siguiente devolverá False:

```
bool(False)
bool(None)
bool(0)
bool("")
bool(())
bool([])
bool({})
```

Las funciones pueden devolver un valor booleano

Puede crear funciones que devuelvan un valor booleano:

Ejemplo

Imprime la respuesta de una función:

```
def myFunction() :
  return True

print(myFunction())
```

Puede ejecutar código basado en la respuesta booleana de una función:

Ejemplo

Imprimir "SÍ!" si la función devuelve True, de lo contrario imprima "NO!":

```
def myFunction() :
  return True

if myFunction():
  print("YES!")
else:
  print("NO!")
```

Python también tiene muchas funciones integradas que devuelven un valor booleano, como la isinstance() función, que se puede usar para determinar si un objeto es de cierto tipo de datos:

Ejemplo

Comprueba si un objeto es un entero o no:

```
x = 200
print(isinstance(x, int))
```

Operadores de Python

Los operadores se utilizan para realizar operaciones en variables y valores.
Python divide los operadores en los siguientes grupos:

- Operadores aritméticos
- Operadores de Asignación
- Operadores de comparación
- Operadores logicos
- Operadores de identidad
- Operadores de membresía
- Operadores bit a bit

Operadores aritméticos de Python

Los operadores aritméticos se usan con valores numéricos para realizar operaciones matemáticas comunes:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Operadores de asignación de Python

Los operadores de asignación se utilizan para asignar valores a las variables:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Operadores de comparación de Python

Los operadores de comparación se utilizan para comparar dos valores:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Operadores lógicos de Python

Los operadores lógicos se utilizan para combinar declaraciones condicionales:

Operator	Description
and	Returns True if both statements are true
or	Returns True if one of the statements is true
not	Reverse the result, returns False if the result is true

Example
x < 5 and x < 10
x < 5 or x < 4
not(x < 5 and x < 10)

Aprendiendo con ejemplos

```
print("Hello, World!")
```