Python

By: Moises Ubaldo Cruz

¿Que es python?

Python es un leguaje de programación de alto nivel y de propósito general. Utiliza un enfoque multiparadigma, lo que significa que soporta programación orientada a objetos, procedural y en menor medida, programación funcional.

Fue creado por **Guido van Rossun** como sucesor a otro lenguaje (llamado ABC) entre 1985 y 1990, y es usado actualmente en una gran variedad de campos, como el desarrollo web, en la creación de aplicaciones actuales y para la construcción de archivos de procesamiento por lotes (Scripts)..

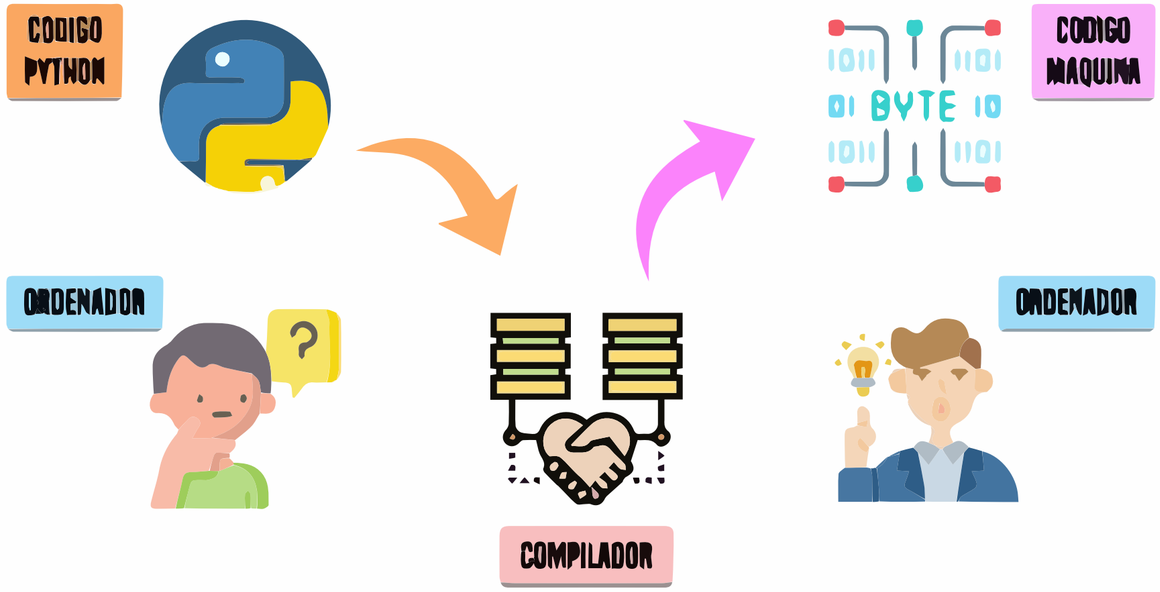
Guido Van Rossum

¿Que es un interprete?

El intérprete funciona de manera similar al shell de Unix: cuando se le llama con una entrada estándar conectada a un **terminal, lee y ejecuta comandos de manera interactiva;** cuando se le llama con un argumento de nombre de archivo o con un archivo como entrada estándar, lee y ejecuta un script desde ese archivo.

Tipado dinamico

Los lenguajes de tipado dinámico son aquellos (como JavaScript) donde el intérprete asigna a las variables un tipo durante el tiempo de ejecución basado en su valor en ese momento.



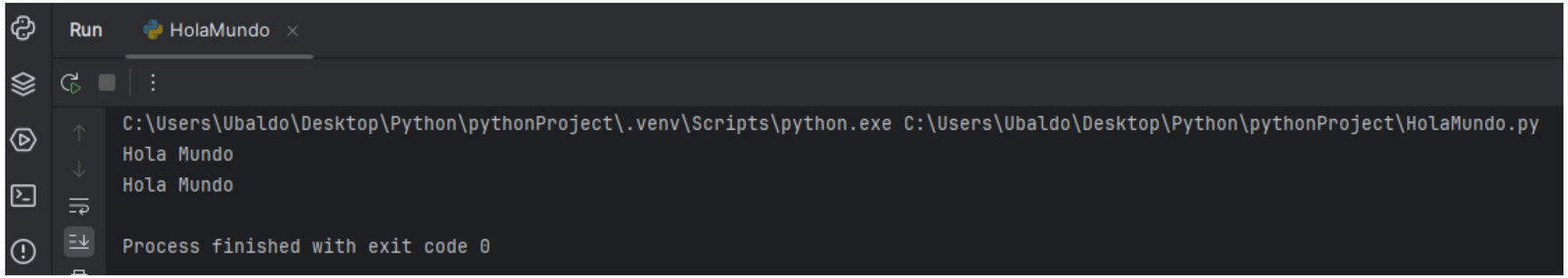
Mi primer programa

HolaMundo.py

```python
#imprimir en la consola

print("Hola Mundo") #Comillas dobles
print('Hola Mundo') #Comillas simples

```



Tipos de datos

Floatantes

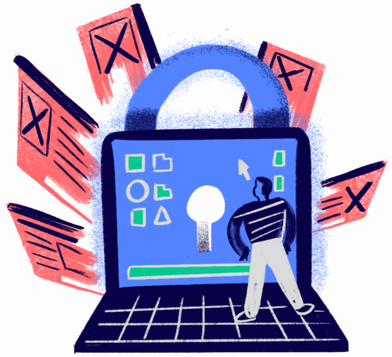
Flotante, o "número de punto flotante" es un número, positivo o negativo, que contiene uno o más decimales.

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

Enteros

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

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



Complejos

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

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

Cadena

La asignación de una cadena a una variable se realiza con el nombre de la variable seguido de un signo igual y la cadena:

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

Ejemplo

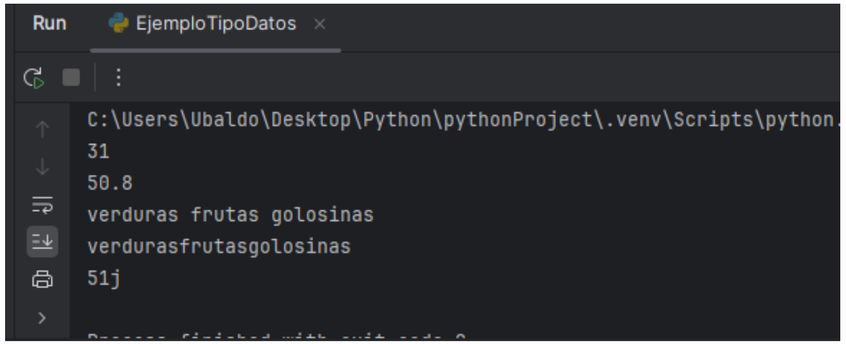
EjemploTipoDatos.py

```python
# Ejemplo con enteros
a = 18
b = 13
print(a+b)

# Ejemplo con float
c = 18.2
d = 32.6
print(c+d)

# Ejemplo con String
e = "verduras"
f = "frutas"
g = "golosinas"
print(e,f,g)
print(e+f+g)

#Ejemplo con complejos
h = 18j
i = 33j
print(h+i)
```



Entrada

Teclado.py

```python
texto = input("Ingresa tu nombre:")
num1 = int(input("Ingresa un numero:"))
num2 = float(input("Ingresa un numero decimal:"))

print("Tu nombre es:", texto)
print("Tu numero enteros es:", num1)
print("Tu numero decimal es:", num2)
```

```
C:\Users\Ubaldo\Desktop\Python\python
Ingresa tu nombre:juan
Ingresa un numero:12
Ingresa un numero decimal:19.5
Tu nombre es: juan
Tu numero enteros es: 12
Tu numero decimal es: 19.5

Process finished with exit code 0
```

Operadores aritméticos

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 comparación

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

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

if-else

```python
usuario = input("Ingresa tu usuario:")
contra = input("Ingresa tu contraseña:")

if (usuario =='root') and (contra == '1234'):
    print("Acceso permitido")
else:
    print("Acceso denegado")
```

Condicional

```
C:\Users\Ubaldo\Desktop\Python\pythonProject\.venv\Script
Ingresa tu usuario:root
Ingresa tu contraseña:1234
Acceso permitido

Process finished with exit code 0
```

elif

```
resul = 0
op = int(input('''Elige una de las siguientes operaciones aritmetica
1)Suma
2)Resta
'''))
num1 = float(input("Ingresa tu primer numero:"))
num2 = float(input("Ingresa tu segundo numero:"))

if op == 1:
    resul = num1+num2
    print("El resultado de tu suma es:",resul)

elif op == 2:
    resul = num1 - num2
    print("El resultado de tu resta es:", resul)

else:
    print("Opcion no valida")
```

```
Elige una de las siguientes operaciones
1)Suma
2)Resta
1
Ingresa tu primer numero:12
Ingresa tu segundo numero:2
El resultado de tu suma es: 14.0
```

Cadena

Función len()

La len() función devuelve el número de elementos de un objeto.
Cuando el objeto es una cadena, la len()función devuelve el número de caracteres de la cadena.

```
txt = "Juan"
texto = "luis"

print(txt.lower())
print(texto.upper())
print(len(txt))
```

Metodo upper()

El upper()método devuelve una cadena donde todos los caracteres están en mayúsculas.
Se ignoran los símbolos y números.

Metodo lower()

El lower()método devuelve una cadena donde todos los caracteres están en minúsculas.
Se ignoran los símbolos y números.

Lista

Las listas se utilizan para almacenar varios elementos en una sola variable.
Las listas son uno de los 4 tipos de datos integrados en Python que se utilizan para almacenar colecciones de datos

Los elementos de la lista están ordenados, modificables y permiten valores duplicados.

```
thislist = ["apple", "banana", "cherry"]
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Tupla

Elementos de tupla

Los elementos de tupla están ordenados, no se pueden modificar y permiten valores duplicados.

Los elementos de tupla están indexados, el primer elemento tiene índice [0], el segundo elemento tiene índice [1], etc.

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
```

```
tuple1 = ("abc", 34, True, 40, "male")
```

Conjuntos

Los conjuntos se utilizan para almacenar varios elementos en una sola variable.
Set es uno de los 4 tipos de datos integrados en Python que se utilizan para almacenar colecciones de datos; los otros 3 son List , Tuple y Dictionary , todos con diferentes calidades y usos.
Un conjunto es una colección desordenada , inmutable * y no indexada .

```
myset = {"apple", "banana", "cherry"}
```

Diccionario

Los diccionarios se utilizan para almacenar valores de datos en pares clave:valor.
Un diccionario es una colección ordenada*, modificable y que no permite duplicados.
Los diccionarios están escritos entre llaves y tienen claves y valores:

```
thisdict = {
  "brand": "Ford",
"model": "Mustang",
  "year": 1964
      }
print(thisdict)
```

While

Con el bucle while podemos ejecutar un conjunto de declaraciones siempre que una condición sea verdadera.

```
i = 1
while i < 6:
  print(i)
  i += 1
```

For

Python para bucles

Un bucle for se utiliza para iterar sobre una secuencia (es decir, una lista, una tupla, un diccionario, un conjunto o una cadena).

Esto se parece menos a la palabra clave for en otros lenguajes de programación y funciona más como un método iterador como el que se encuentra en otros lenguajes de programación orientados a objetos.

Con el bucle for podemos ejecutar un conjunto de declaraciones, una vez para cada elemento de una lista, tupla, conjunto, etc.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

Funcion

Una función es un bloque de código que sólo se ejecuta cuando se llama.
Puede pasar datos, conocidos como parámetros, a una función.
Una función puede devolver datos como resultado.

```
def my_function():
print("Hello from a function")
my_function()
```

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

Herencia

La herencia nos permite definir una clase que hereda todos los métodos y propiedades de otra clase.

La clase principal es la clase de la que se hereda, también llamada clase base.

La clase hija es la clase que hereda de otra clase, también llamada clase derivada.

```
class Person:
def __init__(self, fname, lname):
      self.firstname = fname
       self.lastname = lname

     def printname(self):
print(self.firstname, self.lastname)

   x = Person("John", "Doe")
        x.printname()
```

```
class Student(Person):
        pass


x = Student("Mike", "Olsen")
      x.printname()
```

Iterador

Iteradores de Python

Un iterador es un objeto que contiene un número contable de valores.

Un iterador es un objeto sobre el que se puede iterar, lo que significa que puede recorrer todos los valores.

Las listas, tuplas, diccionarios y conjuntos son todos objetos iterables. Son contenedores iterables de los que puede obtener un iterador.

```
mystr = "banana"
myit = iter(mystr)

print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
```

Modulos

¿Qué es un módulo?

Considere que un módulo es lo mismo que una biblioteca de códigos.
Un archivo que contiene un conjunto de funciones que desea incluir en su aplicación.

Crear un módulo

Guarde este código en un archivo llamadomymodule.py

```
def greeting(name):
print("Hello, " + name)
```

Utilice un módulo

```
import mymodule

mymodule.greeting("Jonathan")
```

Fecha

Una fecha en Python no es un tipo de datos en sí misma, pero podemos importar un módulo nombrado datetimepara trabajar con fechas como objetos de fecha.

```
import datetime

x = datetime.datetime.now()
print(x)
```

```
import datetime

x = datetime.datetime.now()

print(x.year)
print(x.strftime("%A"))
```

Math

Python tiene un conjunto de funciones matemáticas integradas, incluido un módulo matemático extenso, que le permite realizar tareas matemáticas con números.

```
x = abs(-7.25)

print(x)
```

El módulo de matemáticas
Python también tiene un módulo integrado llamado math, que amplía la lista de funciones matemáticas.
Para usarlo, debes importar el mathmódulo:

```
import math

x = math.sqrt(64)

print(x)
```

excepciones

Cuando ocurre un error, o una excepción como la llamamos, Python normalmente se detendrá y generará un mensaje de error.
Estas excepciones se pueden manejar usando la trydeclaración:

```
numero1 = 20
numero2 = 0
    try:
resultado = numero1 / numero2
except ZeroDivisionError:
print("No se puede dividir un numero entre 0")

    finally:
print("Aqui termina el programa")
```

Expresion Regular

Una expresión regular, o expresión regular, es una secuencia de caracteres que forma un patrón de búsqueda.
RegEx se puede utilizar para comprobar si una cadena contiene el patrón de búsqueda especificado.

```
import re

texto = "Encuentra la palabra free para un descuento"
x=re.search("free",texto)
print(x)
```

Formato

F-String se introdujo en Python 3.6 y ahora es la forma preferida de formatear cadenas.

Antes de Python 3.6 teníamos que usar el format()método.

```
price = 59
txt = f"The price is {price} dollars"
print(txt)
```

```
quantity = 3
itemno = 567
price = 49
myorder = "I want {} pieces of item number {} for {:.2f} dollars."
print(myorder.format(quantity, itemno, price))
```

JSON

JSON es una sintaxis para almacenar e intercambiar datos.
JSON es texto escrito con notación de objetos JavaScript.

```
import json

# some JSON:
x =  '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y["name"])
```