LENGUAJE DE PROGRAMACIÓN INTERPRETADO PYTHON (BÁSICO)

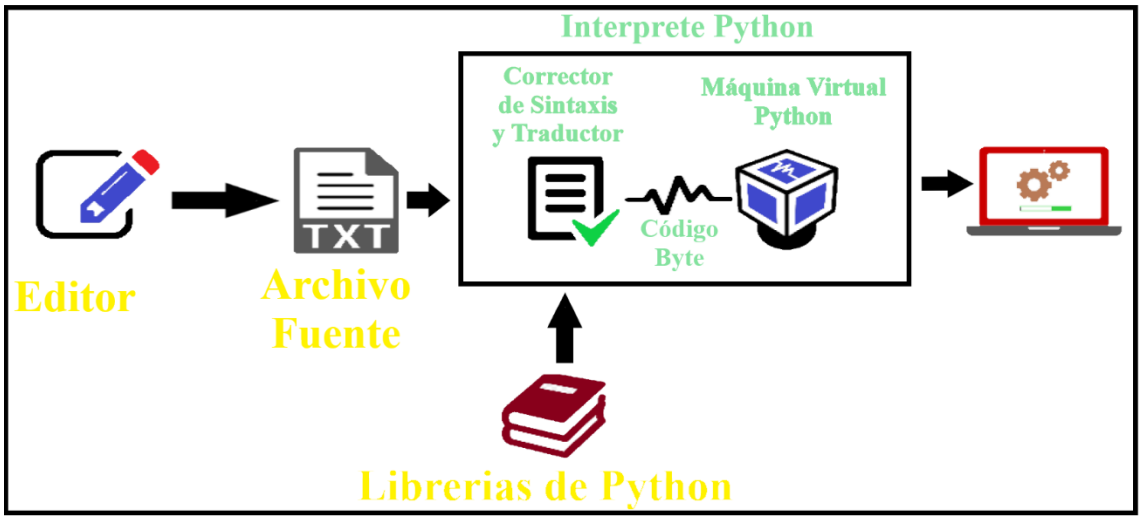
Python es el principal lenguaje de programación que usamos en nuestra escuela para enseñar a los estudiantes conceptos y algoritmos básicos de programación. A nuestros estudiantes les encanta Python: con Python pueden crear sus propias aplicaciones, sitios web, cuestionarios, resolver diferentes problemas y mucho más. Los estudiantes usan diferentes módulos como Pygame, Matplotlib, Numpy, Tkinter, etc. para convertir sus ideas en realidad y desarrollar habilidades como el pensamiento computacional, la creatividad y la resolución de problemas[1].

En este articulo veremos conceptos y aplicaciones de Python entre otros, de esta manera impulsamos la programación y el conocimiento de dicho lenguaje.

¿QUÉ ES PYTHON?

Python es un lenguaje de programación interpretado, cuando nos referimos a interpretado quiere decir que la computadora por sí sola no conoce el lenguaje, de modo que requiere de un intérprete.

Funcionamiento del Interprete Python



Creación Propia

APLICACIONES PARA PHYTON[2].

Desarrollo Web e Internet

Python ofrece muchas opciones para el desarrollo web:

- Frameworks como Django y Pyramid.
- Micro-frameworks como Flask y Bottle.
- Sistemas avanzados de gestión de contenidos como Plone y django CMS.

La biblioteca estándar de Python admite muchos protocolos de Internet:

- HTML y XML
- JSON
- Procesamiento de correo electrónico.
- Compatibilidad con FTP, IMAP y otros protocolos de Internet.
- Interfaz de enchufe fácil de usar.

Y el índice de paquetes tiene aún más bibliotecas:

- Solicitudes, una poderosa biblioteca de cliente HTTP.
- Beautiful Soup, un analizador de HTML que puede manejar todo tipo de HTML extraño.
- Feedparser para analizar fuentes RSS/Atom.
- Paramiko, implementando el protocolo SSH2.
- Twisted Python, un marco para la programación de redes asíncronas.

Científico y Numérico

Python es ampliamente utilizado en computación científica y numérica:

- SciPy es una colección de paquetes para matemáticas, ciencias e ingeniería.
- Pandas es una biblioteca de modelado y análisis de datos.
- Ipython es un poderoso illi interactivo que presenta una fácil edición y grabación de una sesión de trabajo y admite visualizaciones y computación paralela.
- El curso de carpintería de software enseña habilidades básicas para la computación científica, la ejecución de bootcamps y el suministro de materiales didácticos de acceso abierto.

Educación

Python es un lenguaje excelente para enseñar programación, tanto a nivel introductorio como en cursos más avanzados.

- Libros como Cómo pensar como un científico informático, Programación en Python: una introducción a la informática y Programación práctica.
- El Grupo de Interés Especial en Educación es un buen lugar para discutir temas de enseñanza.

GUI de escritorio

La biblioteca Tk GUI se incluye con la mayoría de las distribuciones binarias de Python.

Algunos kits de herramientas que se pueden usar en varias plataformas están disponibles por separado:

- wxWidgets
- Kivy, para escribir aplicaciones multitáctiles.
- Qt vía pyqt o pyside

Los kits de herramientas específicos de la plataforma también están disponibles:

- GTK+
- Microsoft Foundation Classes a través de las extensiones win32

Desarrollo de software

Python se usa a menudo como lenguaje de soporte para desarrolladores de software, para control y administración de compilaciones, pruebas y de muchas otras maneras.

- Scons para el control de compilación.
- Buildbot y Apache Gump para compilación y pruebas automatizadas.
- Roundup o Trac para el seguimiento de errores y la gestión de proyectos.

Aplicaciones de negocios

Python también se usa para construir sistemas ERP y de comercio electrónico:

- Odoo es un software de administración todo en uno que ofrece una gama de aplicaciones comerciales que forman un conjunto completo de aplicaciones de administración empresarial.
- Tryton es una plataforma de aplicaciones de propósito general de alto nivel de tres niveles.

SINTAXIS DE PYTHON

Python fue diseñado para facilitar la lectura y tiene algunas similitudes con el idioma inglés con influencia de las matemáticas.

Python usa nuevas líneas para completar un comando, a diferencia de otros lenguajes de programación que a menudo usan punto y coma o paréntesis.

Python se basa en la sangría, usando espacios en blanco, para definir el alcance; como el alcance de los bucles, funciones y clases. Otros lenguajes de programación a menudo usan corchetes para este propósito.

Ejemplo: print(“Hola, Mundo!”)

Python dará un error si omite la sangría

El número de espacios depende de ti como programador, pero tiene que ser al menos uno.

VARIABLES DE PYTHON[3].

Python no tiene ningún comando para declarar una variable.
Una variable se crea en el momento en que le asigna un valor por primera vez.
Ejemplo:
x = 8
y = "¡Hola, Programador!"
No es necesario declarar las variables con ningún tipo en particular, e incluso pueden cambiar de tipo después de que se hayan establecido.
X = 4 # es un tipo int(número entero).
X = "William" # es un tipo str(cadena).

Casting
Si desea especificar el tipo de datos de una variable, puede hacerlo con la conversión, a este proceso se le llama casting.
Ejemplo:
x = str(20) # x será '20'
y = int(20) # y será 20
z = float(20) # z será 20.0

Nombres de variables
Una variable puede tener un nombre corto (como x num y) o un nombre más descriptivo (fecha, nombre del perro, promedio_final).
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)

Ejemplo
Nombres de variables legales: variable1 = "William"
Variable_1 = "William"
_variable_1 = "William"
VARIABLE1 = William"
VARIABLE_2 = "William"
miVariable = "William"

Nombres de variables de varias palabras
Los nombres de variables con más de una palabra pueden ser difíciles de leer.
Hay varias técnicas que puede utilizar para hacerlos más legibles:
El caso de Carmel: Cada palabra, excepto la primera, comienza con una letra mayúscula:
miVariableNumero = 42
Caso Pascual: Cada palabra comienza con una letra mayúscula:
MiVariableNumero = 40
Caso de serpiente: Cada palabra está separada por un carácter de subrayado:
mi_variable_numero = 46

Asignar valores múltiples

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

```
x, y, z = “azul”, “naranja”, “rosa”
```

asegúrese de que la cantidad de variables coincida con la cantidad de valores, de lo contrario obtendrá un error.

Un valor para múltiples variables: Puede asignar el mismo valor a múltiples variables en una línea: Ejemplo: `x = y = z = “Amarillo”`

Desempaquetar una colección: Si tiene una colección de valores en una lista, tupla, etc. Python le permite extraer los valores en variables. Esto se llama desempacar. Ejemplo

```
colores = [“verde”, “rojo”, “negro”]
x, y, z = colores
```

Variables de salida

La declaración de Python print se usa a menudo para: generar variables.

Para combinar texto y una variable, Python usa el + carácter: Ejemplo

```
x = “William”
print(“Mi nombre es “ + x)
```

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

```
x = “Mi nombre es “
y = “William”
z = x + y
print(z)
```

Para números, el +carácter funciona como un operador matemático: Ejemplo

```
x = 2
y = 35
print(x + y)
```

Si intenta combinar una cadena y un número, Python le dará un error: Ejemplo

```
x = 20
y = “William”
print(x + y)
```

Variables globales

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

Crear una variable fuera de una función y usarla dentro de la función

```
x= “William”
def mifunc():
print(“Mi nombre es ”+ x)
mifunc()
```

Si crea una variable con el mismo nombre dentro de una función, esta variable será local y solo se puede usar dentro de la función. La variable global con el mismo nombre quedará como estaba, global y con el valor original. Ejemplo:
Crear una variable dentro de una función, con el mismo nombre que la variable global.

```
x = "Romeo"

def mifunc():
  x = "William"
  print("Mi nombre es " + x)

mifunc()

print("Su nombre es " + x)
```

resultado

```
Mi nombre es William
Su nombre es Romeo
```

La palabra clave global
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 ámbito global:

```
def mifunc():
global x
x=”William”
mifunc()
print(“Mi nombre es “ + 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 = “William”
def mifunc():
global x
x = “Romeo”
mifunc()
print(“Mi nombre es “ + x)
```

TIPOS DE DATOS DE PYTHON[4].

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 forma predeterminada, en estas categorías:

Tipo de texto	**str**
Tipo numérico	int, float, complex
Tipo de secuencia	list, tuple, range
Tipo de mapeo	dict
Establecer tipos	set, frozenset
Tipo booleano	bool
Tipo binario	bytes, bytearray, memoryview

Obtener el tipo de datos
Puede obtener el tipo de datos de cualquier objeto utilizando la type() función: Ejemplo
Imprime el tipo de datos de la variable y:

```
y = 10
print(type(y))
```

Resultado

```
<class 'int'>
```

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

Ejemplo	Tipo de Dato
x=" Hola Mundo"	str
y=10	int
z=0.6	float
x=2ª	complex
y= ["pera", "guineo", "fresa"]	list
z = ("uva", "mango", "marañón")	tuple
x = range(8)	range
y = {"nombre": "Axel", "Edad": 35}	dict
z = {"manzana", "uva", "mango"}	set
x = frozenset ({"uva", "mango", "manzana"})	frozenset
y = True	bool
z = b"Hola"	bytes
x = bytearray(4)	bytearray
x = memoryview(bytes(6))	memoryview

Configuración del tipo de datos específico

Si desea especificar el tipo de datos, puede utilizar las siguientes funciones de construcción:

Ejemplo	Tipo de Dato
x=str(" Hola Mundo")	str
y=int(10)	int
z=float(0.6)	float
x=complex(2ª)	complex
y= list(("pera", "guineo", "fresa"))	list
z =tuple(("uva", "mango", "marañón"))	tuple
x = range(8)	range
y = dict("nombre": "Axel", "Edad": 35)	dict
z = set (("manzana", "uva", "mango"))	set
x = frozenset (("uva", "mango", "manzana"))	frozenset
y = bool(6)	bool
z = bytes(3)	bytes
x = bytearray(4)	bytearray
x = memoryview(bytes(6))	memoryview

ELEMENTOS DE ACCESO[5].

No puede acceder a los elementos de un conjunto haciendo referencia a un índice o una clave, pero puede recorrer los elementos del conjunto usando un bucle for, o preguntar si un valor específico está presente en un conjunto, usando la palabra clave in. Ejemplo: Recorra el conjunto e imprima los valores:

```
thisset = {"rojo", "azul", "verde"}

for y in thisset:
  print(y)
```

Resultado

```
azul
verde
rojo
```

Ejemplo: Compruebe si "Kia" está presente en el conjunto:

```
thisset = {"toyota", "Kia", "Hyundai"}

print("Kia" in thisset)
```

```
True
```

```
thisset = {"toyota", "Volvo", "Hyundai"}

print("Kia" in thisset)
```

```
False
```

Agregar elementos del conjunto

Para agregar un elemento a un conjunto, utilice el add() método. Ejemplo:
Agregue un artículo a un conjunto, usando el add() método:

```
thisset = {"manzana", "pera", "uva"}

print(thisset)
```

```
{'uva', 'pera', 'manzana'}
```

```
thisset = {"manzana", "pera", "uva"}

thisset.add("fresa")

print(thisset)
```

```
{'pera', 'manzana', 'uva', 'fresa'}
```

Agregar cualquier iterable

El objeto en el update() método no tiene que ser un conjunto, puede ser cualquier objeto iterable (tuplas, listas, diccionarios, etc.). Ejemplo:
Agregue elementos de una lista a un conjunto:

```
thisset = {"caballo", "perro", "gato"}
mylist = ["loro", "oso"]
thisset.update(mylist)
print(thisset)
```

```
{'loro', 'gato', 'perro', 'oso', 'caballo'}
```

QUITAR ELEMENTOS DEL CONJUNTO[6].

Remover el artículo

Para eliminar un elemento de un conjunto, utilice el método remove() o discard().
Ejemplo: Elimina “gallo” usando el método remove().

```
thisset = {"perro", "gallo", "gato"}

print(thisset)
```

```
{'gallo', 'gato', 'perro'}
```

Aplicando remove()

```
thisset = {"perro", "gallo", "gato"}
thisset.remove("gallo")
print(thisset)
```

```
{'perro', 'gato'}
```

Ejemplo: Elimina “perro” usando el discard() método:

```
thisset = {"ballena", "perro", "pulpo"}
print(thisset)
```

```
{'perro', 'ballena', 'pulpo'}
```

Aplicando discard()

```
thisset = {"ballena", "perro", "pulpo"}
thisset.discard("perro")
print(thisset)
```

```
{'pulpo', 'ballena'}
```

También puede usar el pop() método para eliminar un elemento, pero este método eliminará el último elemento. Recuerde que los conjuntos están desordenados, por lo que no sabrá qué elemento se elimina, el valor de retorno del pop() método es el elemento eliminado.

Ejemplo: Elimine el último elemento utilizando el pop() método:

```
thisset = {"casa", "carro", "moto"}
x = thisset.pop()
print(x) #item removido
print(thisset) #despues de remover el item
```

```
casa
{'carro', 'moto'}
```

```
thisset = {"casa", "carro", "moto"}
x = thisset.pop()
print(x) #item removido
print(thisset) #despues de remover el item
```

```
moto
{'casa', 'carro'}
```

Ejemplo: El clear() método vacía el conjunto:

```
thisset = {"uva", "pera", "fresa"}
thisset.clear()
print(thisset)
```

```
set()
```

CONDICIONES DE PYTHON Y SENTENCIAS IF[7].

Python admite las condiciones lógicas habituales de las matemáticas:

Es igual a:	**a == b**
No es igual a:	a != b
Menos a:	a < b
Menor o igual que:	a <= b
Mayor que:	a > b
Mayor o igual que:	a >= b

if

Estas condiciones se pueden usar de varias maneras, más comúnmente en "sentencias if" y bucles. Una "sentencia if" se escribe utilizando la palabra clave if.

Ejemplo: if declaración:

```
x = 33
y = 200
if y > x:
 print("y mayor que x")
```

```
y mayor que x
```

elif

La palabra clave elif es la forma de Python de decir "si las condiciones anteriores no fueron ciertas, intente esta condición". Ejemplo:

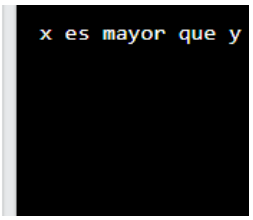
```
x = 10
y = 10
if y > x:
  print("y es mayor que x")
elif x == y:
  print("x y y son iguales")
```

```
x y y son iguales
```

else

La palabra clave else captura cualquier cosa que no esté capturada por las condiciones anteriores. Ejemplo:

```
x = 110
y = 39
if y > x:
  print("y es mayor que x")
elif x == y:
  print("x y y son iguales")
else:
  print("x es mayor que y")
```

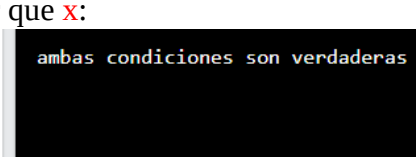


En este ejemplo, x es mayor que y, por lo que la primera condición no es verdadera, tampoco la condición elif es verdadera, así que vamos a la condición else e imprimimos en la pantalla que "x es mayor que y".

and

La palabra clave and es un operador lógico y se usa para combinar sentencias condicionales: Ejemplo: Prueba si x es mayor que y, Y si z es mayor que x:

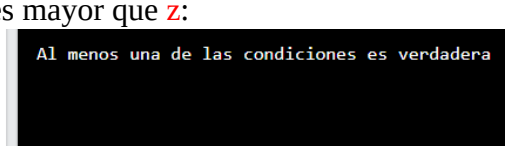
```
x = 200
y = 33
z = 500
if x > y and z > x:
  print("ambas condiciones son verdaderas")
```



or

La or palabra clave es un operador lógico y se usa para combinar declaraciones condicionales: Ejemplo: Prueba si x es mayor que y, O si x es mayor que z:

```
x = 350
y = 21
z = 724
if x > y or x > z:
  print("Al menos una de las condiciones es verdadera")
```



INFOGRAFÍA

[1] «Welcome to Python.org», *Python.org*. https://www.python.org/success-stories/elementary-school-education-is-it-love-or-just-python/ (accedido 9 de marzo de 2022).

[2] «Applications for Python», *Python.org*. https://www.python.org/about/apps/ (accedido 9 de marzo de 2022).

[3] «Python - Nombres de variables». https://www.w3schools.com/python/python_variables_names.asp (accedido 9 de marzo de 2022).

[4] «Tipos de datos de Python». https://www.w3schools.com/python/python_datatypes.asp (accedido 10 de marzo de 2022).

[5] «Python - Elementos del conjunto de acceso». https://www.w3schools.com/python/python_sets_access.asp (accedido 10 de marzo de 2022).

[6] «Python - Quitar elementos del conjunto». https://www.w3schools.com/python/python_sets_remove.asp (accedido 10 de marzo de 2022).

[7] «Condiciones de Python». https://www.w3schools.com/python/python_conditions.asp (accedido 10 de marzo de 2022).

Elaborado por: William Abrego