



Universidad Autónoma del Estado de México

Facultad de Geografía



Programación con Python

Orientado a la automatización de procesos en QGIS

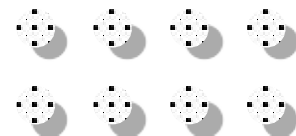
Ing. Adonai Emmanuel
Nicanor Bautista

¿Qué es Python?

- Es un lenguaje de programación potente y fácil de aprender.
- Es un lenguaje ideal de scripting, lo que permite un desarrollo rápido de aplicaciones en distintas áreas y son multiplataforma (para diferentes sistemas operativos).



Illustrations by Pixeltrue on [icons8](#)

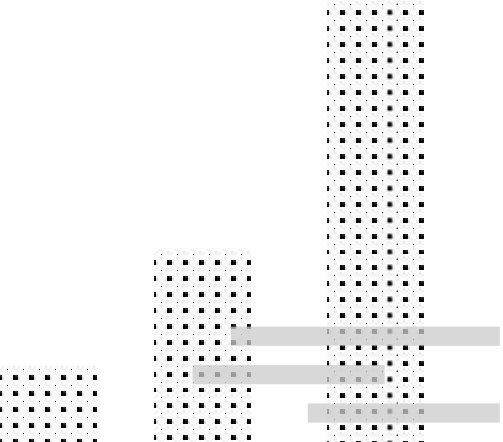




REPL

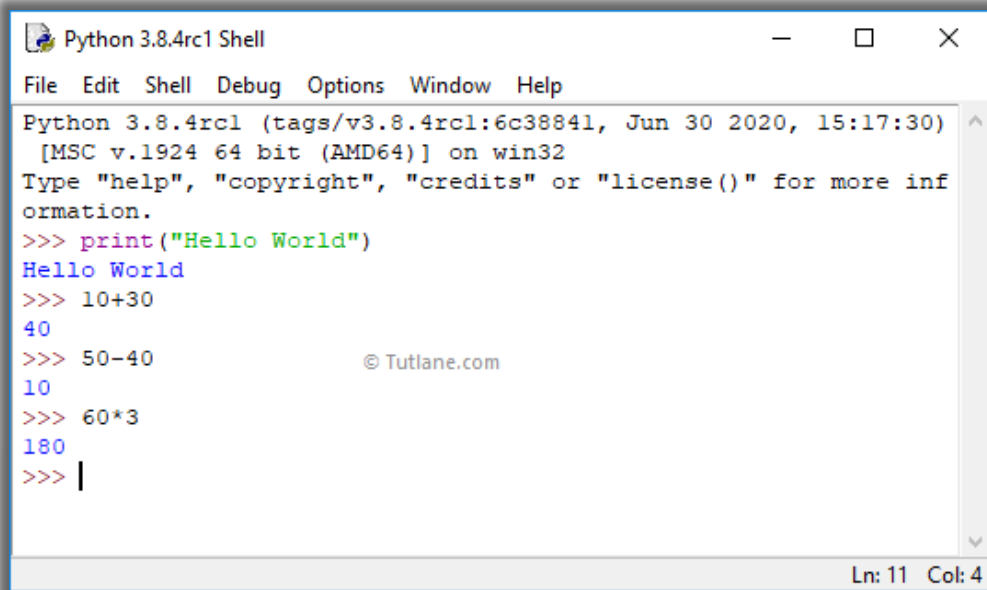
Read Evaluate Print Loop
(Leer Evaluar Imprimir Bucle)

- 1) **Lee** los comandos de Python.
- 2) **Evalúe** el código para averiguar qué significa.
- 3) **Imprime** los resultados para ver la respuesta.
- 4) **Regresar** al paso 1 para seguir comunicándose.

- Permite comunicarnos con el interprete de Python.
 - Cuando se ejecuta una instrucción en el REPL de forma automática se visualiza el resultado de esta instrucción.
 - Permite obtener ayuda de métodos y funciones.
- 

REPL

Read Evaluate Print Loop
(Leer Evaluar Imprimir Bucle)



```
Python 3.8.4rc1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.4rc1 (tags/v3.8.4rc1:6c38841, Jun 30 2020, 15:17:30)
[MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf
ormation.
>>> print("Hello World")
Hello World
>>> 10+30
40
>>> 50-40
10
>>> 60*3
180
>>> |
```

© Tutlane.com

Ln: 11 Col: 4

```
python3
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

Tipos de Operadores

ARITMÉTICOS

Operador	Descripción
+	Adición
-	Sustracción
*	Multiplicación
/	División
//	División (entero)
%	Módulo
**	Potencia

RELACIONALES

Operador	Descripción
>	Mayor que
<	Menor que
==	Igual a
>=	Mayor o igual que
<=	Menor o igual que
!=	Diferente de

LÓGICOS

Operador	Descripción
and	Éste y éste
or	Éste o éste
not	Negación de éste

Tipos de Operadores

PERTENENCIA

Operador	Descripción
in	Ésta en
not in	No esta en

IDENTIDAD

Operador	Descripción
is	Es
is not	No es

Más tipos de operadores en:
<https://cutt.ly/E2JTIIj>

Indentación/Indentación

Estilo de Sangría – Indentation Style

- Número de espacios en blanco después de cada línea de código
- En Python es obligatorio que sean 4 espacios (1 tabulador) en blanco, de lo contrario mostrará error.

```
1. 🐍 suma.py
1 def sumar(num1, num2):
2     suma = num1 + num2
3     print("Resultado:", suma)
4
5 if __name__ == "__main__":
6     num1 = 10
7     num2 = 3
8     sumar(num1, num2)
9     ----|
```

```
1. 🐍 suma.py (6) ?
1 def sumar(num1, num2):
2     suma = num1 + num2
3     print("Resultado:", suma)
4
5 if __name__ == "__main__":
6     num1 = 10
7     | num2 = 3
8     sumar(num1, num2)|
```

Diagnostics:
1. Unindent not expected

No se esperaba una sangría

Indentación/Identación

Estilo de Sangría – Indentation Style

- No es necesario teclear los 4 espacios en blanco, con la tecla TAB (Tabulador) nos brinda la facilidad de colocar estos 4 espacios.
- Es importante mencionar que **NO SE DEBEN COMBINAR** espacios en blanco y tabuladores, ya que muchas de las veces lo detecta como errores.



Comentarios en Código

- Son la forma más sencilla de documentar el código que se escribe, permite identificar las variables que se están utilizando y toda aquella información relevante que se requiere **recordar/saber** sobre el código.

- Comentarios en una sola línea
- Comentarios múltiples líneas

```
3 # Esto es un comentario
4
5 """
6
7 Esto es un comentario
8 multiples lineas
9
10 """
```

Variables

- Es la representación de algo que esta sujeto a un cambio.
- Se encarga de almacenar y recuperar datos dentro de un fragmento de código.
- La creación de variables de forma general en la programación esta sujeto a un par de reglas.
 - 1) No se deben utilizar espacios entre palabras (es mejor usar guion bajo [_] para unir dos palabras o usar primera minúscula y después mayúscula para la segunda palabra)
 - 2) No deben comenzar con un número
 - 3) No deben utilizarse palabras reservadas (propias del lenguaje)

La estructura es: **nomVariable = algo**

Concatenación


- Es el proceso de unir dos o más cadenas de texto, esta unión se realiza al anexar una cadena al final de otra cadena.
- En Python, la concatenación se realiza con el signo [+]

```
2 cadena1 = "Hola "  
3 cadena2 = "Mundo"  
4  
5 resultado = cadena1 + cadena2
```



Tipos de datos



- Son el conjunto de valores que tienen una serie de propiedades y características determinadas.
 - **Números**
 - Enteros (int/integer)
 - Reales (float)
 - Complejos (imaginarios)
 - **Strings**
 - Cadenas de texto
 - Carácter
 - **Booleanos (Verdadero/Falso)**
 - **Listas**
 - **Tuplas**
 - **Conjuntos**
 - **Diccionarios**
- 

Strings

- **Métodos de Búsqueda (Retornan la posición del substring)**

- `Str.find(substr)` - primer identificación
- `Str.rfind(substr)` - para la ultima identificación

- **Métodos de Unión (Unir cadenas a través de un iterable)**

- `Str.join(iterable)`

- **Métodos de Reemplazo (Cambiar un string por otro)**

- `Str.replace(old_str,new_str,n_veces)`

- **Métodos para eliminar (Stripping) espacios en blanco (cualquier otro carácter)**

- `Str.lstrip()` - Eliminación a la izquierda
- `Str.rstrip()` - Eliminación a la derecha
- `Str.strip()` - Eliminar en toda la cadena

- **Métodos para división**

- `Str.split(algo,n_veces)`

Listas (list)

- Es una estructura de datos ordenada, modificable, dinámica y que permite almacenar elementos repetidos y de diferente tipo.
- Se caracteriza por la estructura []
- Los métodos más utilizados son:
 - Append()
 - Copy()
 - Count()
 - Insert()
 - Reverse()
 - Remove()
 - Sort()
 - Pop()
 - Extend()
 - Index()
 - Clear()

```
2  
3 lista = [1, 2, 3, 4, 5, 6, 7, 8, 8, 8, 8]  
4
```

Tuplas (tuples)

- Es una estructura de datos ordenada, inmutable (no se puede modificar), que puede almacenar diferentes tipos de datos.
- Se caracteriza por la estructura ()
- Sus métodos son:
 - Count()
 - Index()

```
2 tupla = (1, 2, 3, 4, 5, 6, "abc", 1)
```

Conjuntos (set)

- Es una estructura de datos **NO** ordenada que puede incluir ciertos tipos de datos (booleanos, flotantes, Strings, tuplas). Sus usos se extienden principalmente a la lógica y las matemáticas.
- Se caracteriza por la estructura { }
- Los métodos más utilizados son:
 - add()
 - clear()
 - pop()
 - union()
 - issuperset()
 - issubset()
 - intersection()
 - difference()
 - isdisjoint()
 - setdiscard()
 - copy()

```
2 conjunto = {1, True, None, (5, 6), "abc"}
```


Diccionarios (dict)

- Permiten almacenar cualquier información, son mutables (se pueden modificar), e incluso pueden combinar distintos tipos de datos como tuplas, listas, conjuntos u otros diccionarios.
- Su estructura es la siguiente: { **"clave única"** : **tipo de dato** }
- Los métodos más usados son los siguientes:
 - copy()
 - clear()
 - fromkeys()
 - items()
 - get()
 - keys()
 - pop()
 - values()
 - update()
 - setdefault()
 - popitem()

```
2 dicc = { "nombres" : ["Pedro", "Juan", "Carlos"],  
3         "apellidos": ["Gonzales", "Perez", "Garcia"] }
```

Lectura de datos (input)

- Permite introducir datos capturados por el teclado a través de la terminal o shell.
- Sin importar lo que se introduzca, estos datos serán leídos como **STRING**.
- Para dar valor a un string número a número, es necesario realizar un **CASTEO (cast/casting)**.
- La sintaxis es la siguiente: **nomVariable = input()**

```
4 nombre = input()  
5 print("Mi nombre es:", nombre)
```

Funciones

- Son un fragmento de código que contiene rutinas específicas para una acción, las funciones pueden ser llamadas tantas veces sean requeridas, pueden recibir o no argumentos (datos) y generar procesos de salida.
- Su estructura es la siguiente:

```
2 def suma_2num(numero1, numero2):  
3     sumar = numero1 + numero2  
4     return sumar
```

Def – Sintaxis para inicializar una función

suma_2num – Nombre de la función

(numero1, numero2) – Argumentos de la función

Sumar – rutina específica

Return sumar – generación de procesos de salida

Módulos

- Son bibliotecas que permiten tener acceso a otros códigos de programación, ya sean propios, del mismo Python o elaborados por terceros. Estos módulos contienen funciones adicionales que facilitan el procesamiento de datos e información.
- Los módulos de terceros pueden ser instalados con el gestor de librerías llamado **pip** a través de la consola.
- La palabra reservada para invocar a estos módulos es a través de **import**.

```
1 import math
2 pi = math.pi
```