ls- comando para listar directorios y archivos

pwd – para ver el directorio actual

git init- para iniciar un nuevo proyecto

git status – para ver el estado de nuestros archivos

git add – para agregar un archivo al staging área

git commit – para crear un primer punto de control e nuestro código

git config – para configurar ciertos campos

git log – para ver todos los commits que has hecho

git checkout – para revertir los cambios de los archivos

git diff – para ver las diferencias hechas en los archivos

git Branch – saber las ramas

git remote – decir donde se almacena el código

git clone – para poder copiar de github a consola de git

git push—remplaza master por la rama a la que quieres enviar tus cambios

git pull -- para actualizar tu repositorio local al commit mas nuevo se ejecuta en el directorio de trabajo para bajar y fusionar los cambios remotos

Como abrir extensions: (CTRL+MAYUSC+X)

Como abrir explorer: (CTRL+MAYUSC+E)

Folder-Workspace-File

Intellisense (Ayuda a terminar o dictar la función que quieres utilizar) (CTRL+BARRA ESPACIADORA)

Code Snippets: Facilita para repetir patrones de codigos como loops o estados de condición.

Mandar a correr un código: (CTRL+ALT+N)

Como guardar un archivo: (CTRL+S)

Guardar como: (CTRL+MAYUSC+S)

PYTHON GENERALIDADES:

* **Propósito Generales:** Permite desarrollar aplicaciones de cualquier tipo de rama del conocimiento y cualquier tipo de aplicación.
* **Tercera Generación:** Es llamado así, ya que esta basado en un conjunto de instrucciones que permiten detallar de forma procedural y secuencial la forma en como han de realizarse las tareas.
* **Orientado a objetos:** Permite la creación e instanciación de clases, permite implementar las tres características POO:

1. Encapsulamiento: Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad. Al mismo nivel de abstracción.
2. Herencia: Las clases no están aisladas, sino que se relacionan entre si, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. Organiza y facilita el polimorfismo y encapsulamiento.
3. Polimorfismo: Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, pero al llamarlos se utilizará el comportamiento correspondiente al objeto que se esté usando.

* **Interpretado:** Cada vez que se manda a ejecutar un programa, se realiza la verificación sintáctica (parsing) y la generación de código binario ejecutable, para la plataforma en que es ejecutado (compilación).
* **Sensible a mayúsculas y minúsculas (case sensitive):** Si importa el uso de mayúsculas y minúsculas, para palabras reservadas, como para el nombre de elementos creados por el desarrollador.
* **Posicional (Indented):** La posición en la que inicia una línea de código influye en la manera en la que el interprete de Python, produce resultados.
* **No requiere la especificación detallada de tipos (Non Strong Type):** Las variables no requieren ser declaradas antes de utilizarse, ni deben utilizarse si son declaradas.
* **Sin terminador de línea:** No requiere de “FinMientras” etc.

**PYTHON TUTORIAL**

* Cuál es el terminador de línea en Python (**Enter**).
* Cómo defino un bloque de código en Python (*Identacion*).

If 5>2:

print (“El 5 es mayor a 2”)

* Cómo se colocan comentarios en Python (**#**).

#Este programa sirve de tal y tal forma.

* Cómo genero una salida de consola en Python (**print**).
* Cómo se definen las variables en Python.

X = “Hola” o Y = ‘hola’

* Funciones para el establecimiento explícito de *data type*.

|  |  |  |
| --- | --- | --- |
| **Example** | **Data Type** | **Try it** |
| x = str("Hello World") | str | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_str2) |
| x = int(20) | int | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_int2) |
| x = float(20.5) | float | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_float2) |
| x = complex(1j) | complex | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_complex2) |
| x = list(("apple", "banana", "cherry")) | list | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_list2) |
| x = tuple(("apple", "banana", "cherry")) | tuple | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_tuple2) |
| x = range(6) | range | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_range2) |
| x = dict(name="John", age=36) | dict | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_dict2) |
| x = set(("apple", "banana", "cherry")) | set | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_set2) |
| x = frozenset(("apple", "banana", "cherry")) | frozenset | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_frozenset2) |
| x = bool(5) | bool | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_bool2) |
| x = bytes(5) | bytes | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_bytes2) |
| x = bytearray(5) | bytearray | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_type_bytearray2) |
| x = memoryview(bytes(5)) | memoryview |  |

* Cuáles son las reglas de nombrado de variable en Python.

1. El nombre de una variable debe iniciar con una letra o guion bajo.
2. No puede iniciar con un número.
3. Solo puede tener caracteres alfanuméricos y guion bajos.
4. Las variables son sensibles, “edad”, “Edad” y “EDAD” son diferentes.

* Qué tipos de notación son los más utilizados en la actualidad para el nombrado de elementos en un programa:

**NombreUsuario** - Pascal Casing (Todos los inicios de palabra en mayúsculas)

**nombreUsuario** - Camel Casing (Primera minúscula, y luego mayúsculas los inicios de palabra.

**objLista** – Hungarian notation (Incluye referencia al tipo, como prefijo o como sufijo)

* Qué notación se sugiere para el nombrado de variables en Python (*camelCase*).
* Qué notación se sugiere para el nombrado de clases, métodos y funciones (*PascalCase*).
* Cómo se declaran funciones simples, que no reciben parámetros, y no retornan valores (**def**).

def my\_function():  
  print("Hello from a function")

* Qué son las variables globales. Son las que se crean fuera de una función y pueden ser utilizadas tanto por fuera de la función como adentro
* Que son las variables locales. Es cuando se crea una variale dentro de una función y slo puede usarse dentro de esta
* Para que sirve **global.** Para crear una variable global dentro de una función
* Cuáles son los *data types* admitidos por Python

|  |  |
| --- | --- |
| * Text Type: | str |
| * Numeric Types: | int, float, complex |
| * Sequence Types: | list, tuple, range |
| * Mapping Type: | dict |
| * Set Types: | set, frozenset |
| * Boolean Type: | bool |
| * Binary Types: | bytes, bytearray, memoryview |

* Función para determinar el tipo de dato de una variable (**type()**)

Print(Type(x))

* Funciones de conversión de tipos de datos.

a = float(x)  
  
#convert from float to int:  
b = int(y)  
  
#convert from int to complex:  
c = complex(x)

* Uso de **str.isdigit()**  
  da verdadero si todos los caracteres de la cadena son dígitos
* Cómo se importan librerías a un programa Python (**import**)
* Función de generación de números aleatorios (**random.randrange()**). Requiere importar la librería (**random**).
* Función para preguntar información desde la consola (**input()**)
* Trabajo con strings.
  + Uso de literal.

1. are surrounded by either single quotation marks, or double quotation marks.
   * Uso de multilínea.
2. You can assign a multiline string to a variable by using three quotes:

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

* + Uso de **str.len**.

Para poder saber la longitud de la cadena

a = "Hello, World!"  
print(len(a))

* + Métodos:

## str.strip

quita los espacios en blanco desde el principio hasta el final

a = " Hello, World! "  
print(a.strip()) # returns "Hello, World!"

* + - **str.lower:**

hace que la cadena este en minusculas

a = "Hello, World!"  
print(a.lower())

* + - **str.upper**

hace que la cadena este en mayúsculas

a = "Hello, World!"  
print(a.upper())

* + - **str.replace**

remplaza una cadena por otra

a = "Hello, World!"  
print(a.replace("H", "J"))

* + - **str.split**

divide la cadena en subcadenas si hay un separador

a = "Hello, World!"  
print(a.split(",")) # returns ['Hello', ' World!']

* + - **str.partition**

|  |  |
| --- | --- |
| [partition()](https://www.w3schools.com/python/ref_string_partition.asp) | Returns a tuple where the string is parted into three parts |

* + **not in**

para verificar si esta cierta frase o carácter en a cadena

txt = "The rain in Spain stays mainly in the plain"  
x = "ain" not in txt  
print(x)

* + Concatenación

a = "Hello"  
b = "World"  
c = a + b  
print(c)

* Dando formato a strings usando placeholders (**str.format**)

The format() method formats the specified value(s) and insert them inside the string's placeholder.

string.format(value1, value2...)

* Manejo de datos booleanos.

You can evaluate any expression in Python, and get one of two answers, True or False.

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

* Manejo de operadores (aritméticos, asignación, lógicos, comparación)
  + Sólo los más usuales.
  + Especial atención en el uso de **is** e **is not**, con **type()**.
* Manejo de **if:**evalua condiciones a cumplir
* Manejo de **if :**/ **else:** es la ultima opción si no se cumplen las condiciones verdaderas
* Manejo de **while:** se ejecuta un bloque de código siempre y cuando tenga una condición verdadera
* Manejo de **while:** / **else:**
* Uso de **break** (para ciclos infinitos)

With the break statement we can stop the loop even if the while condition is true:

i = 1  
while i < 6:  
  print(i)  
  if i == 3:  
    break  
  i += 1

* Manejo de **for**

Se usa para poder iterar cierto bloque de código

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

* Uso de **range()** con **for.**

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

for x in range(6):  
  print(x)

for x in range(2, 30, 3):  
  print(x)

for x in range(6):  
  print(x)  
else:  
  print("Finally finished!")