

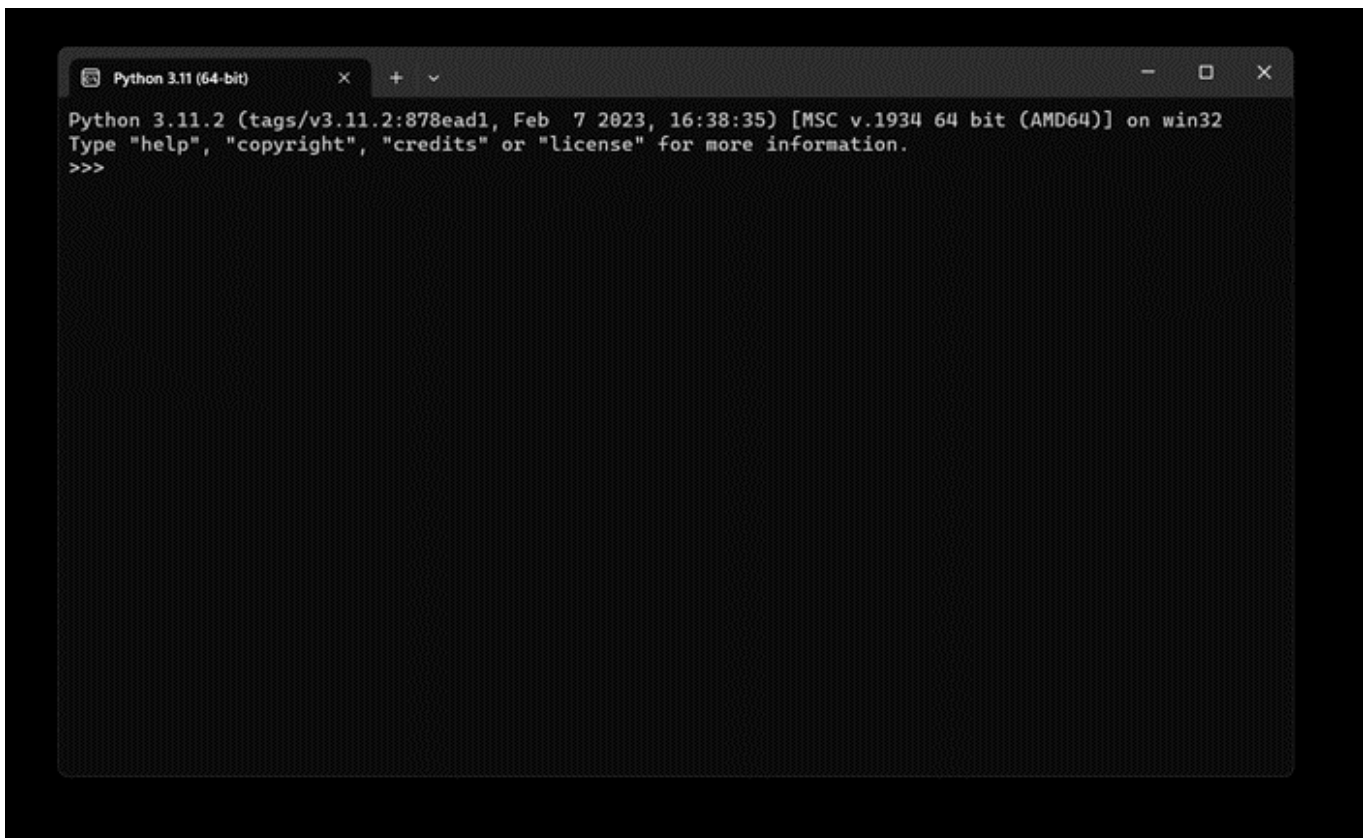
Python

Principios Básicos

Interprete

- Python interpreta el código por medio del interprete o consola, el cual es esencial para trabajar con Python

Interprete :



Datos Simples

Los datos Simples son aquellos datos básicos, en ellos tenemos:

1- Text String o cadena de textos, como su nombre lo indica es un texto común el cual debe estar siempre definido entre comillas (" "), se puede definir en Python con 3 tipos de comillas distintas, las cuales son:

comillas normales("""

comillas simples(")

comillas triples (''' ''')

Las diferencias de estas es que las comillas triples al hacer un salto de línea sigue figurando como texto al contrario que con las comillas normales y las comillas simples, Ejemplos de los tipos de textos:

```
# Tipos de textos Con comillas ' ' " " ' ' ' ' ' '
"Texto"
"Otro texto"
"""Otro Tipo de texto"""
```

2- Números, en Python los números se pueden clasificar en dos (2) grupos

`#Int` y `#Float`, los datos `#Int` son datos de números enteros en cambio los datos `#Float` son datos de números decimales. Ejemplos:

```
# tipos de datos Numericos INT=Entero FLOAT=Decimal
40 # INT=Entero
40.2 # FLOAT=Decimal
```

3- Datos Booleanos, los datos de tipo Booleano constan de dos (2) datos los cuales son `#true` y `#false`, cuando estamos escribiendo código con estos datos se deben escribir siempre con la primera letra en mayúscula. Ejemplos:

```
# Tipos de datos booleanos
True # Verdadero
False # Falso
```

Si los datos se escriben con la primera letra en minúscula aparecerá un error

Comando `#Print`

Uno de los comandos principales en Python es el comando "print ()" este comando se usa para mostrar algo en consola por ejemplo "print ("Hola Mundo")" esto hará que se muestre en consola el texto "Hola mundo", otro ejemplo seria "print (3.45)" esto mostrara en consola el numero 3,45 el comando "print ()" siempre debe ser escrito en minúscula, si lo que se va a mostrar en consola con `#Print` es un texto se debe poner con comillas de esta forma- print ("Texto a mostrar")

- Ejemplos con el comando `#Print` :
- `#Print` con texto

```
print("Hola Mundo")
```

- `#Print` con números
- `#Int` (Enteros)

```
print(3)
```

- `#Float` (Decimales)

```
print(3.45)
```

Comando Print en la consola de Python

```
Python 3.11 (64-bit)
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hola Mundo")
Hola Mundo
>>> print (3.45)
3.45
>>> print (3)
3
>>> |
```

Variables

Definición

Para definir una variable en Python se necesita la siguiente estructura

Nombre de la variable = Contenido de la variable (Si el contenido de la variable es un texto este se debe escribir dentro de comillas "")

Cuando el nombre de una variable es largo las variables se pueden definir de dos modos las cuales son:

- CamelCase
- Snake_Case

#CamelCase

- Se pone en mayúsculas las primeras letras de cada palabra, por ejemplo:

```
# Definición de una variable con CamelCase
nombreCompletoDelUsuario = "Felipe"
```

#Snake_Case

- Se pone un guion bajo entre cada palabra, por ejemplo:

```
# Definición de una variable con Snake_Case
nombre_completo_del_usuario = "Juan"
```

Concatenación de una variable

La concatenación es la unión de 2 o más variables, en Python las formas más usadas de concatenar son con el símbolo + o con la concatenación F-Strings.

- +

La concatenación con el símbolo + une a las variables de texto, para concatenar usando esta tipo de concatenación es de la siguiente manera:

```
# concatenacion con +
Bienvenida = "Hola " + nombre + " ¿Como estas?"
#NuevaVariable |         |         |
#              Texto   variable  texto
```

- F-Strings

La concatenación con F-Strings une a cualquier variable sin importar lo que contenga porque convierte cualquier cosa en texto , para concatenar usando este tipo de concatenación es de la siguiente manera

```
# concatenacion con F-Strings (Convertir cualquier cosa a texto)
Bienvenida = f"Hola {nombre2} ¿Como estas?"
#NuevaVariable |         |         |
#              Texto   variable  texto
```

Variables usadas en los ejemplos

```
nombre = "Felipe"  
nombre2 = 3
```

Datos Compuestos

Los datos compuestos se utilizan principalmente para agrupar distintos tipos de datos en ellos, los datos compuestos son:

- **La Lista:**

Estructura: Nombre de la lista = []

- Dentro de [] se ponen los elementos de la lista, cada elemento debe ser separado por comas (,) si un elemento es una cadena de texto de debe poner en comillas (" "). La característica principal de una lista es que es Modificable, Ejemplo de lista:

```
# Lista = Modificable  
  
lista = ["Juan", True, 160, "Python"]
```

- **La Tupla**

Estructura: Nombre de la tupla = (),

- dentro de () se ponen los elementos de una Tupla, cada elemento debe ser separado por comas (,), si un elemento es una cadena de texto de debe poner en comillas (" "). En comparación con las listas las Tuplas no tienen la virtud de ser Modificables, Ejemplo de Tupla:

```
# Tupla = No se puede modificar  
  
tupla = ("Juan", True, 160, "Python")
```

Para Mostrar el contenido de una lista o tupla con el comando `#Print` se debe hacer la siguiente forma:

- `print = (Nombre de la lista/tupla[Numero del objeto])`
- Ejemplos:

```
print(lista[1])
```

```
print(tupla[1])
```

En la lista o en la tupla el numero del objeto empieza desde cero, ejemplo:

```
lista = ["Juan", True, 160, "Python"]
#           0       1       2       3

tupla = ("Juan", True, 160, "Python")
#           0       1       2       3
```

- **Conjunto**

Estructura: `NombreDelConjunto = {}`

- Dentro de `{ }` se ponen los elementos de un Conjunto, cada elemento debe ser separado por comas (,), si un elemento es una cadena de texto de debe poner en comillas (" "). En comparación con las listas o las Tuplas en los conjuntos no se repiten los datos, es decir si tenemos dos (2) o más datos iguales dentro del conjunto este lo va a contar como uno, además en un conjunto no se pueden cambiar los datos pero si se puede reasignarlos, aparte de esto al conjunto no se puede entrar por medio de un índice es decir por medio del numero de sus datos, para acceder a el se tiene que mostrar con el comando `#Print` el conjunto completo . Ejemplo de Conjunto:

```
# conjunto (Set) (no se repiten valores no se accede por un indice)

conjunto = {"Juan", 1.60 , "Python"}
```

Como se debe mostrar un conjunto con el comando `#Print` :

```
print (conjunto)
```

- **Diccionario**

Estructura: Nombre_del_diccionario = {

```
#Key : #Value ,
}
```

- Funciona como un diccionario tradicional, en `#Key` colocamos el nombre de lo que vamos a poner dentro de esta y en `#Value` lo que significa la `#Key` cada vez que pongamos una `#Key` nueva con su respectivo `#Value` se debe poner una coma al final del `#Value` si se va a agregar abajo un nuevo dato, `#Key` y `#Value` ejemplo:

```
"altura": 160,
#   |      |
# Key Value
```

Ejemplo de Diccionario:

```
Diccionario = {
    "nombre": "Juan",
    "emocion": "Feliz",
    "edad": 20,
    "altura": 160,
}
```


La `#Key` y el `#Value` siempre que sea un texto se debe poner en comillas " "

- Cuando vamos a usar el comando `#Print` para mostrar un diccionario se puede acceder a un objeto en específico accediendo a él por medio de su `#Key` por ejemplo:

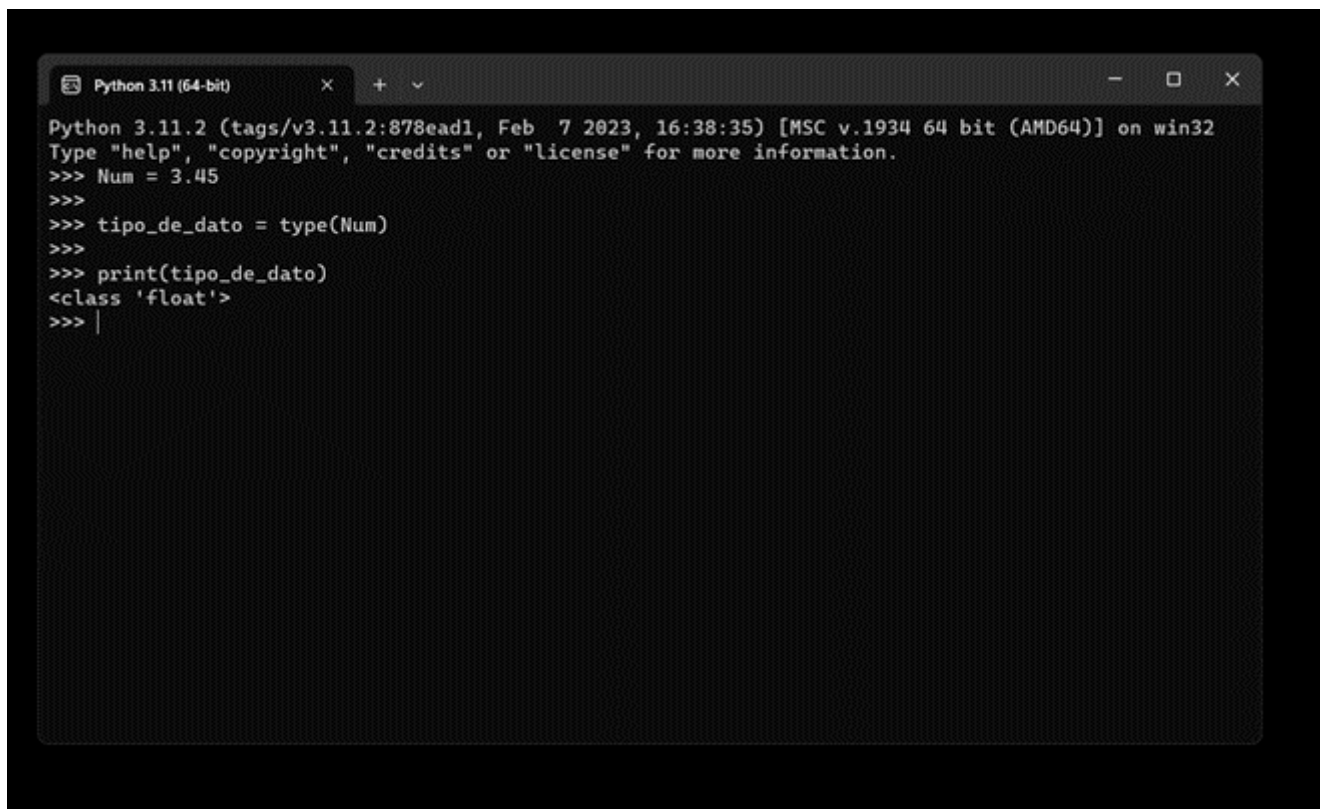
```
print(Diccionario["altura"])  
# |  
# Muestro solo este dato
```

Comando `#Type`

- Otro de los comandos más usados e importantes en Python es el comando `#Type` que nos muestra el tipo del dato que le demos, al igual que con `#Print` se debe escribir en minúscula.
- Estructura: `type()`
Dentro de `()` se pone el dato el cual queremos saber el tipo, además el comando se debe alojar en una variable para poder ser mostrado con el comando `#Print`. Ejemplo:

```
Num = 3.45  
# |  
# Variable  
  
tipo_de_dato = type(Num)  
  
print(tipo_de_dato)
```

Comando Type en la consola de Python:

A screenshot of a Python 3.11.2 terminal window. The window title is "Python 3.11 (64-bit)". The terminal shows the following code and output:

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> Num = 3.45
>>>
>>> tipo_de_dato = type(Num)
>>>
>>> print(tipo_de_dato)
<class 'float'>
>>> |
```

- En "Class" podemos ver como me dice el tipo de la variable "Num"

Condicionales

Las condicionales las usamos para que según la acción del código haga una cosa u otra, las condicionales las podemos dividir en tres (3)

- If
- Else
- Elif

If-Else

Si tenemos un código y por ejemplo tenemos que validar una contraseña usamos las condicionales para ejecutar una acción u otra según la contraseña que tengamos en el sistema y la que la persona escriba por ejemplo:

```
contraseña_base_de_datos = "Fernando123"
contraseña_escrita = "Fernando123"

if contraseña_base_de_datos == contraseña_escrita:
    print("Iniciando Sesion....")
else:
    print("Contraseña incorrecta, Intente de Nuevo")
```

Aquí lo que estamos haciendo es que si la variable "contraseña_base_de_datos" y la variable "contraseña_escrita" son iguales dice "Iniciando Sesión ..." si no es así nos dice "Contraseña incorrecta, Intente de Nuevo"

De esta forma podemos hacer comparaciones y que nuestro programa actúe depende de el resultado

Elif

Esta nos sirve para mostrar una segunda opción, si el primer (If) no se cumple comprueba con la segunda opción (Elif) y si ninguna se cumple se ejecuta la tercera opción (Else) por ejemplo:

```
ingresos = 5000
gastos_mensuales = 4000

if ingresos - gastos_mensuales < 0:
    print("Estas En deficit")
elif ingresos - gastos_mensuales > 3000:
    print("Tas Bien")
else:
    print("Estas Gastando Bastante")
```