Clase 1-La sintaxis de Python

August 9, 2023

1 Iniciando en Python

1.1 Hola Mundo en Python

En cualquier introducción a un nuevo lenguaje de programación, no puede faltar el famoso Hola Mundo. Se trata del primer programa por el que se empieza, que consiste en programar una aplicación que muestra por pantalla ese texto. Si ejecutas el siguiente código, habrás cumplido el primer hito de la programación en Python.

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

Por lo tanto ya te puedes imaginar que la función print() sirve para imprimir valores por pantalla. Imprimirá todo lo que haya dentro de los paréntesis. Fácil ¿verdad? A diferencia de otros lenguajes de programación, en Python se puede hacer en 1 línea.

[ ]:
```
print("Hola mundo desde Python")
```

```
Hola mundo desde Python
```

1.2 Comentarios

- Los comentarios pueden ser de una sola línea o de varias líneas.

[ ]:
```
# Esto es un comentario en Python
print("Curso de Python desde Cero")
print("Hola desde python")
print(4+5)

"""
Esto es un comentario de varias
lineas en Python
print("Esto no se muestra")
"""
```

```
Curso de Python desde Cero
Hola desde python
9
```

[ ]:
```
'\nEsto es un comentario de varias \nlineas en Python\nprint("Esto no se
muestra")\n'
```

1.3 Definiendo variables en Python

Vamos a complicar un poco más las cosas. Creemos una variable que almacene un número. A diferencia de otros lenguajes de programación, no es necesario decirle a Python el tipo de dato que queremos almacenar en x. En otro lenguajes es necesario especificar que x almacenará un valor entero, pero no es el caso. Python es muy listo y al ver el número 5, sabrá de que tipo tiene que ser la x.

```
x = 5
```

Ahora podemos juntar el print() que hemos visto con la x que hemos definido, para en vez de imprimir el Hola Mundo, imprimir el valor de la x.

```
print(x) # Salida: 5
```

En el anterior fragmento habrás visto el uso #. Se trata de la forma que tiene Python de crear los denominados comentarios. Un comentario es un texto que acompaña al código, pero que no es código propiamente dicho. Se usa para realizar anotaciones sobre el código, que puedan resultar útiles a otras personas. En nuestro caso, simplemente lo hemos usado para decirte que la salida de ese comando será 5, ya que x valía 5.

1.4 Sumando Variables en Python

Vamos a sumar dos variables e imprimir su valor. Lo primero vamos a declararlas, con nombres a y b. Declarar una variable significa "crearla".

```
# Declaramos las variables a, b
# y asignamos dos valores
a = 3
b = 7
```

Ahora Python ya conoce a y b y sus respectivos valores. Podemos hacer uso de + para sumarlos, y una vez más de print() para mostrar su valor por pantalla.

`print(a+b)` Es importante que sólo usemos variables que hayan sido definidas, porque de lo contrario tendremos un error. Si hacemos:

```
print(z)
# Error! La variable no existe
```

Tendremos un error porque Python no sabe que es z, ya que no ha sido declarada con anterioridad.

[ ]:
```
# Variables y tipos:

texto = "Programación Competitiva"
print("Esto es un texto:")
print(texto)

entero = 10

print("Esto es un entero:")
print(entero)
```

```python
decimal = 10.0

print("Esto es un decimal:")

print(decimal)

logico = True

print("Esto es un valor logico:")

print(logico)
```

```
Esto es un texto:
Programación Competitiva
Esto es un entero:
10
Esto es un decimal:
10.0
Esto es un valor logico:
True
```

1.5 Concatenar en Python

[ ]:
```python
# Definicion de variables:

cad1 = "Esto"
cad2 = "es"
cad3 = "Python"

# Mostrar variables con print:

print("Pasando las cadenas como parámetros al print.")
print(cad1, cad2, cad3)

# Concatenar una sola cadena:

cadena = cad1 + " " + cad2 + " " + cad3
print("Utilizando una variable")
print(cadena)

# También puede hacerse desde el print:
print("Formando la cadena en el print")
print(cad1 + " " + cad2 + " " + cad3)

# Tercera manera, utilizando f:
print("Utilizando formateo - interpolado")
print(f"{cad1} {cad2} {cad3}")
```

```
# Cuarta forma, con format:
print("Utilizando la cláusula format:")
print("Cadena 1: {}, Cadena 2: {}, Cadena 3: {}".format(cad1,cad2,cad3))
```

```
Pasando las cadenas como parámetros al print.
Esto es Python
Utilizando una variable
Esto es Python
Formando la cadena en el print
Esto es Python
Utilizando formateo - interpolado
Esto es Python
Utilizando la cláusula format:
Cadena 1: Esto, Cadena 2: es, Cadena 3: Python
```

1.6 Mas acerca de los tipos de datos en Python

[ ]:
```
# Tipos de datos de variables:
nada = None
cadena = "Programación Ucatec"
entero = 7
decimal = 10.50
logico = False
listaNumeros = [10, 20, 30, 40, 50, 60]
listaNombres = ["Juan", "Carlos", "María", "Fernanda"]
# La tupla  es similar a una lista pero no cambia
tupla = ("Programación", "en", "Python")
# Un diccionario es una lista con clave y valor.
diccionario = {
    "nombre": "Juan",
    "apellido": "Perez",
    "ocupacion": "Programador"
}
rango = range(9)
dato_byte = b"Ucatec"

# Imprimiendo datos de variables:
print(nada)
print(cadena)
print(entero)
print(decimal)
print(logico)
print(listaNumeros)
print(listaNombres)
print(tupla)
print(diccionario)
```

```
print(rango)
print(dato_byte)

# Mostrando tipos de datos:
print(type(nada))
print(type(cadena))
print(type(entero))
print(type(decimal))
print(type(logico))
print(type(listaNumeros))
print(type(listaNombres))
print(type(tupla))
print(type(diccionario))
print(type(rango))
print(type(dato_byte))
```

```
None
Programación Ucatec
7
10.5
False
[10, 20, 30, 40, 50, 60]
['Juan', 'Carlos', 'María', 'Fernanda']
('Programación', 'en', 'Python')
{'nombre': 'Juan', 'apellido': 'Perez', 'ocupacion': 'Programador'}
range(0, 9)
b'Ucatec'
<class 'NoneType'>
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'list'>
<class 'list'>
<class 'tuple'>
<class 'dict'>
<class 'range'>
<class 'bytes'>
```

1.7 Conversión de Tipos de Datos

[ ]:
```
texto = "Ucatec Programación"
anio = 2023

# Se puede mostrar enviando parametros:
print(texto, anio)
```

```
# Pero para concatenar se debe convertir el tipo de dato:
print(texto + " " + str(anio))

# Otras conversiones de tipos:
numero = int(7)

print(numero)
print(type(numero))
print(type(str(numero)))

# Reasignando numero:
numero = float(7)

print(numero)
print(type(numero))
print(type(str(numero)))
```

```
Ucatec Programación 2023
Ucatec Programación 2023
7
<class 'int'>
<class 'str'>
7.0
<class 'float'>
<class 'str'>
```

1.8 Strings

[ ]:
```
texto1 = "Programación"
texto2 = "\"Ucatec\""

cadena1 = texto1 + " " + texto2

print(cadena1)

texto3 = 'Club'
texto4 = '"Python"'
cadena2 = texto3 + " " + texto4

print(cadena2)

# Valores especiales:
print(cadena1 + '\t' + cadena2)
print(cadena1 + '\n' + cadena2)
print(cadena1 + '\r' + cadena2)
```

```
Programación "Ucatec"
Club "Python"
```

```
Programación "Ucatec"   Club "Python"
Programación "Ucatec"
Club "Python"
Club "Python""Ucatec"
```

1.9 Operadores

[ ]:
```python
# Operadores de asignación:
a = 2
b = 2

# Operadores aritméticos:
print("Operaciones Aritméticas")
print(f"La suma es: {a + b}")
print(a - b)
print(a * b)
print(a / b)
print(a // b)
print(a % b)
print(a ** b)

# Asignaciones complejas, incremento y decremento:
num = 7

print("Asignaciones complejas")
num += 1
print(num)
num -= 1
print(num)
num *= 2
print(num)
num //= 2
print(num)
num /= 2
print(num)

print("incremento y decremento")
anio = 2023
print(anio)
anio = anio + 1
print(anio)
anio = anio - 1
print(anio)
```

```
Operaciones Aritméticas
La suma es: 4
0
4
```

```
1.0
1
0
4
Asignaciones complejas
8
7
14
7
3.5
incremento y decremento
2023
2024
2023
```

1.10 Entrada y Salida de Datos

[ ]:
```python
# Entrada de datos:
num1 = input("Ingresa un número: ")
num2 = input("Ingresa otro número: ")

suma = int(num1) + int(num2)

# Salida de datos:
print("La suma es: " + str(suma))
```

```
La suma es: 12
```