Python

Historia de Python

- Creado por el Holandés Guido Van Rossum a comienzos de los 90's
- El trabajaba en un sistema operativo Amoeba mediante el uso de un bourne Shell, debido a las dificultades que presentaban en la compatibilidad creo Python para usarlo como una interface

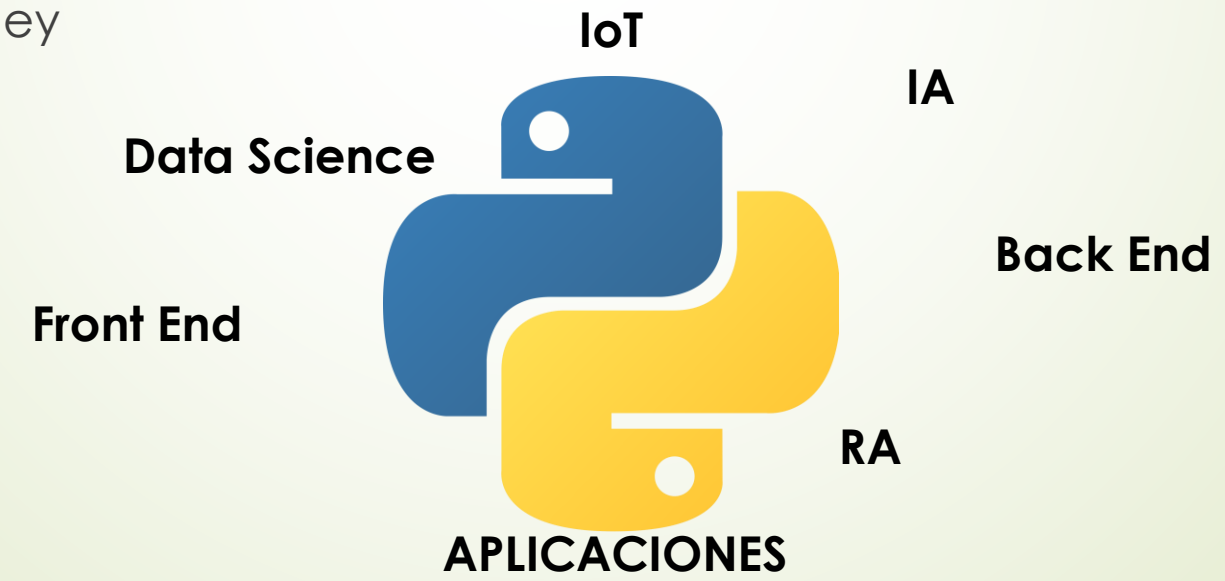
Características

- Lenguaje de alto nivel. Gramática sencilla, clara, muy legible
- Tipado dinámico y fuerte
- Orientado a objetos
  - Sobrecarga de constructores. Herencia múltiple. Encapsulación. Interfaces. Polimorfismo
- Open Source
- Fácil de aprender
- Librería estándar muy amplia
- Interpretado
- Versátil
  - Aplicaciones de escritorio, aplicaciones de servidor, aplicaciones web
- Multiplataforma

¿Por qué Python?

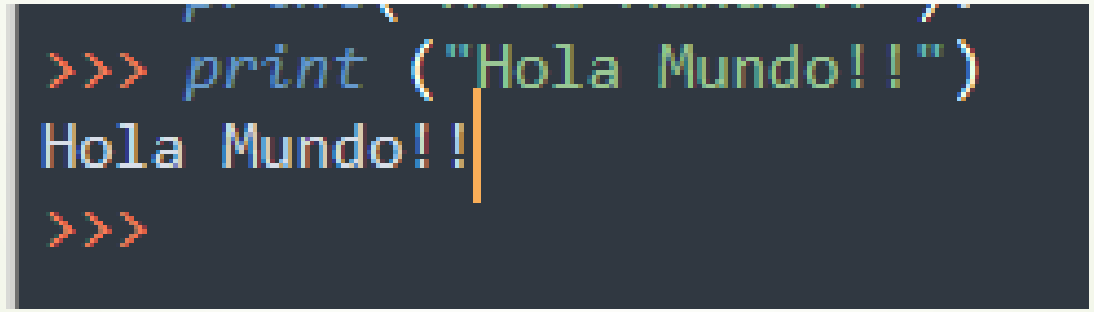
- **Python** es un lenguaje de programación de alto nivel, con aplicaciones en numerosas áreas, incluyendo programación web, scripting, computación científica e inteligencia artificial.

- **Es muy popular** y usado en organizaciones como Google, NASA, la CIA o Disney



¡Bienvenido a Python!

- Comencemos creando un programa corto que muestre “Hola Mundo!!”
- En Python se utiliza la declaración **print** para generar el texto, el cual debe ir entre comillas simples o dobles.



- Escribe un programa que imprima “Python es divertido y fácil.”. Utiliza la declaración **print** para generar el texto.

Operaciones simples

- Python tiene la capacidad de realizar cálculos directamente en la declaración de impresión.
- Suma, resta, multiplicación y división se pueden realizar y utilizar paréntesis para dar prioridad a ciertas operaciones.

```
>>> print(2 * (3 + 4))
14
>>> print(10 / 2)
5.0
>>> print(5 + 4 - 2)
7
```

Floats

- Los **floats** se usan para representar números que nos son enteros. Por ejemplo 0.5 y -7.823.
- Se pueden crear directamente ingresando el numero con punto decimal o como resultado de alguna operación.
- También puede ser el resultado de la operación entre un entero y un float o dos floats.

```
>>> print (3/4)
0.75
>>> print (0.42)
0.42
>>> print (1/3)
0.3333333333333333
```

```
>>> print (4 + 1.65)
5.65
>>> print (3.1 + 2.5)
5.6
```

Exponenciación

- Además de las operaciones básicas también es posible manejar **exponentes**, es decir elevar un numero a una potencia. Esta operación se realiza utilizando dos asteriscos **
- Es posible encadenar exponentes, para elevar un numero a multiples potencias.

```
>>> print (2**5)
32
>>> print(2**3**2)
512
>>> print (4**0.5)
2.0
>>> print (9**(1/2))
3.0
```

Cociente y resto

- La **división entera** se realiza mediante dos barras oblicuas y se utiliza para determinar el cociente de una división.
- El operador de **modulo** se lleva a cabo con el símbolo % y nos permite sacar el **resto** de una división.

```
>>> print (20/6)
3.3333333333333335
>>> print(20//6)
3
>>> print (20%6)
2
>>> print(1.25%0.5)
0.25
```

Cual es el resultado

- print (1+4*3)
- print ((3**2)//2)
- Genere un código que nos de 10 elevado a la potencia 5.
- Cual es el cociente de 100 entre 42.
- Se te ofrece una opción de $1,000,000 o $0.01 que se duplicara cada día por 30 días. Escribe un programa que calcule la cantidad que resultara.

Cadenas de texto

- Una cadena de texto se crea introduciendo texto entre comillas simples o dobles
- Algunos caracteres no se pueden incluir directamente en una cadena, Por ejemplo, las comillas dobles.
- Para agregar algunos de estos caracteres es necesario incluir una barra diagonal inversa antes que ellos.
- **\n** significa un salto de línea. **\t** es una tabulación

```
>>> print ('Brian\'s mother')
Brian's mother
>>> print ("Brian's mother")
Brian's mother
>>> print ("Mama de Brian")
Mama de Brian
>>> print ("uno\ndos\ntres")
uno
dos
tres
```

```
>>> print (""" Esta
... es una
... multilinea
... de texto""")
 Esta
es una
multilinea
de texto
```

Concatenación

- Unión de dos cadenas.
- Multiplicación por enteros

Variables

- Una variable permite almacenar un valor asignándole un nombre (Identificador) que se utiliza para referirse al valor mas adelante en el programa.

- Hay restricciones al momento de asignar nombre a una variable, Solo permite usar letras, números y guion bajo. No puede iniciar con numero ni incluir espacios.

- Python es sensible a mayúsculas y minúsculas. **Apellido** y **apellido** son dos nombres de variable distintos
- Las variables pueden reasignarse

```
>>> user="Roger"
>>> password="123456"
>>> User="Juan"
>>> print(user)
Roger
>>> print(User)
Juan
>>> este_nombre_es_valido="Hola Mundo"
>>> 1este_no="si"
  File "<stdin>", line 1
    1este_no="si"
     ^
SyntaxError: invalid syntax
>>>
```

```
>>> user=2538
>>> print (user)
2538
>>> user="Hugo"
>>> print (user)
Hugo
```

Entrada y salida sencilla

- Para obtener información del usuario se utiliza el comando **input()**
- Esta función solicita al usuario una entrada y devuelve lo que ingresa como una cadena.
- Se puede proporcionar una cadena entre los paréntesis a modo de mensaje de solicitud.
- Para convertir la cadena en numero usamos **int()** y para convertir un numero en cadena la función **str()**, útil para concatenar cadenas

```
>>> x=input()
7
>>> nombre=input("Dame un numbre:")
Dame un numbre:Roger
>>> print (nombre)
Roger
>>> print(x)
7
>>> print(x*3)
777
>>> edad=int(input("Introduce tu edad:"))
Introduce tu edad:6
>>> print(edad*5)
30
```

Operadores in place y Walrus

- Los operadores in situ te permiten escribir **x=x+3** como **x+=3**
- Es posible utilizar cualquier operación matemática (+, -, *, /, **, //, %)
- El Operador **Walrus :=** permite asignar valores dentro de una expresión
- Por ejemplo:
  - num = int(input())
  - print (num)
- Se puede expresar como:
  - print(num := int(input())

Listas

- Son Estructuras de Datos que nos permiten almacenar varios valores (Array)
- Permiten guardar diferentes tipos de valores
- Lista=[elem1, elem2, elem3, elem,4]
- Funciones básicas:
  - Lista[índice] //puede ser negativo
  - Lista[:] //Rango
  - **.append** (elem) //agrega un elemento al final
  - **.insert** (índice, elem) //agrega un elemento en una posición especifica
  - **.extend**([elementos]) //permite agregar varios elementos
  - **.index**(elemento) //permite conocer el índice de un elemento
  - Elemento **In** lista //Nos responde si existe algún elemento en la lista
  - **.remove**(elemento) // Eliminar un elemento de una lista
  - **.pop**() //elimina el ultimo elemento de una lista
  - **len**(lista) //nos da el total de elementos en la lista

Tuplas

- Son listas inmutables, no se puede modificar después de su creación
- Permite extraer porciones, pero no permite búsqueda (**index**), permite **in.**
- Son mas rápidas y ocupan menos espacio que una lista.
- Tupla=(elem1, elem2, elem3)
- Funciones Básicas:
  - Tupla[índice] //referencia a un elemento
  - **list**(tupla) //Convertir tupla en lista
  - **len**(tupla) **//**
  - **tuple**(lista) //Convertir lista en tupla.
  - Elemento **in** tupla //Permite saber si un elemento existe en la tupla
  - **.count**(elemento) //contar cuantas veces existe elemento en tupla

Diccionarios

- Estructura de datos que permite almacenar valores de diferentes tipos.
- La principal característica es que los datos se almacenan asociados a una clave. **clave:valor**
- Los elementos almacenados no están ordenados.
- Midiccionario={"clave_1":"valor_1","clave_2":"valor_2","clave_3":"valor_3"}
- Funciones básicas:
  - Dicc["clave_x"]=valor_x //agregar o modificar un valor
  - **Del** dicc["clave_x"] //eliminar un dato
  - Dicc.**keys()** // Imprime claves
  - Dicc.**values()** // imprime valores
  - **Len**(dicc) //devuelve el tamaño del diccionario