**PYTHON DESDE CERO**

**Estructura basica**

# Comentarios

"""

Comentario de varias líneas

"""

print("Hola Mundo") No lleva ;

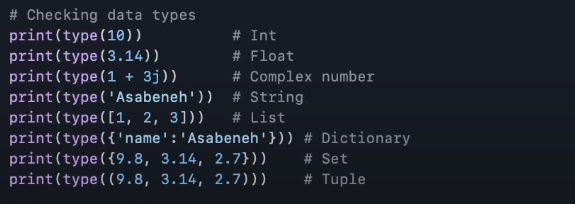
print(‘Hola Mundo’)

Puede llevar comillas dobles o simples.

**Data Types**

Toma un elemento y devuelve el tipo de elemento que es, por eje:

String, Boolean, Int, Float, List, Complex Numbre, Dictionary, Set, Etc.



Python es un lenguaje con tipado dinamico, osea que podemos cambiar el tipo de variable cuando queramos. Una lista se puede transformar en un string, y biseversa. Un numero en una lista, y demas ejemplos. Todos los tipos deobjetos pueden cambiar de tipo cuando queramos.

**Variables**

Las variables deben estar escritas solo en minusculas, y con \_ para separar las palabras.

**mi\_variable = 30**

Podes imprimir multiples variables separandolas con una coma **‘,’ .**

**print (mi\_variable, variable\_dos, variable\_bool)**

Tambien podemos cambiar de tipo a las variables. Por ej: Si recibimos de un input un valor numerico pero necesitamos operarlo, debemos de cambiar el tipo de STR 🡪 INT.

**valor\_recibido = " 30 " Str 🡪 Int**

**cambio\_de\_Valor = int(valor\_recibido)**

O en el caso contrario

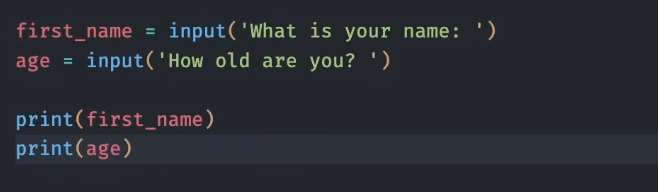
**valor\_variable = 30 Int 🡪 Str**

**cambio\_de\_Valor = str(valor\_variable)**

**Constantes**

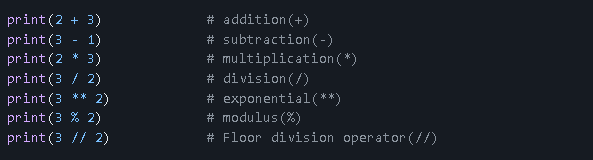
No es posible crear constantes en Python, todos los elementos pueden cambiar su tipo o su valor cuando quieras. Lo que se hace comun mente es llamar a la variable **CONST\_NAME =** esto da a aclarar que no se debe cambiar el valor, pero a nivel sintaxico no se pueden crear,

**Inputs**

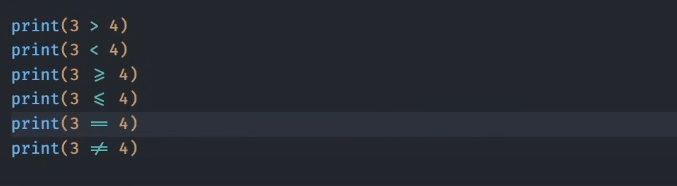


**Operadores**

**Operadores Aritmeticos**



**Operadores Comparativos** Devuelven True o False



**Strings**

**\n** **Salto de linea**

**print("Este es un string\nCon salto de linea")**

**Este es un string**

**Con salto de linea**

**\t** **Tabulacion**

**print("\tEste es un string con tabulación en linea")**

**Este es un string con tabulación en linea**

**\\** **Escapado**

**print("\\t Este string \\n ignora las anteriores secuencias.")**

**\t Este string \n ignora las anteriores secuencias.**

**CONCATENAR VARIABLES Y STRINGS**

Existen varias maneras para concatenar variables dentro de strings.

La mas comun es interrumpiendo y separando el string y las variables, esta forma no se recomienda para textos largos ya que es mas complicado programarlo y es mas propenso a conllevar errores. La mejor manera para concatenar strings y variables es con **Format stings**.

**nombre = "Juan"**

**edad = 30**

**print("Mi nombre es", nombre, "y tengo", edad, "años.")**

**Format string**

Colocando una **f** al inicio del string podras colocar las variables en **{ }** . Esta es la mejor manera para formatos de codigo simple.

**nombre = "Juan"**

**edad = 30**

**print(f"Mi nombre es {nombre} y tengo {edad} años.")**

Colocando operadores para cada variables y luego con .format indicarlas en orden.

**nombre = "Juan"**

**edad = 30**

**print("Mi nombre es %s y tengo %d años."** %**(name,edad))**

**%s** Para los strings. **%d** Para los integers. **%f** Para los floats.

De esta manera se corroborá que cada valor que se le pasa al string es del tipo que se espera. Por ej: Si pasamos un string al **%d** este dará un error ya que **%d** solo admite numeros enteros.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Esta manera es similar a la anterior pero en esta se encierra con **{ }** donde queremos insertar los elementos, esta manera es mas sencilla de programar pero no se asegura que los elementos sean del tipo correcto.

**nombre = "Juan"**

**edad = 30**

**print("Mi nombre es {} y tengo {} años.".format(name,edad))**

**Desempaquetado de Caracteres**

El "desempaquetado" de caracteres en Python se refiere a descomprimir una secuencia de caracteres en variables individuales. Esto se puede hacer fácilmente en Python utilizando una sintaxis simple. Por ejemplo:

**nombre = "Santiago"**

**a, b, c, d, e, f, g, h = nombre**

**print(a)**

**print(d)**

**print(e)**

**print(f)**

**Devuelve : S t g o**

**Division de Caracteres**

Muestra desde el primer indice hasta el degundo sin contar el segundo.

**slice = nombre[1:3]**

**print(slice)**

**Devuelve : an**

Muestra desde el primer indice en adelante.

**slice = nombre[1:]**

**print(slice)**

**Devuelve : antiago**

Muestra el elemento contando desde el final hacia atrás.

**slice = nombre[-1]**

**print(slice)**

**Devuelve : o**

**Reversion de Caracteres**

**reverse = nombre[::-1]**

**print(reverse)**

**Devuelve : ogaitnaS**

**Salteo de Caracteres**

Muestra desde el primer indice hasta el degundo y luego se indica el incremento (de cuanto en cuantos caracteres saltará)

**Abecedario = “ABCDEFGHIJKLMNOPQ”**

**jumped = Abecedario[0:17:2]**

**print(jumped)**

**Devuelve : ACEGIKMOQ**

**Metodos de Strings**

**Capitalize** pone en mayuscula al primer elemento de la cadena.

**texto = "hola mundo"**

**print(texto.capitalize()) # Salida: Hola mundo**

**Upper** pone todos los elemetos en Mayusculas.

**print(texto.upper()) # Salida: HOLA MUNDO**

**Lower** pone todos los elementos en Minusculas.

**print(texto.lower()) # Salida: hola mundo**

**Tittle** pone en mayuscula la primera letra de cada palabra.

**print(texto.tittle()) # Salida: Hola Mundo**

**Count** cuenta la cantidad de veces que aparece una letra en el texto.

**texto = "hola mundo"**

**print(texto.count("o")) # Salida: 2**

**Comprobaciones de Strings**

Los metodos de strings se pueden concatenar con otros metodos de comprobacion de stings.

**Isnumeric** verifica si el elemento dado es un numero o no.

**texto = "hola mundo"**

**print(texto.isnumeric()) # Salida: False**

**print("1".isnumeric()) # Salida: True**

**Isupper** verifica si el texto tiene todas las letras en Mayusculas.

**print(texto.upper().isupper()) # Salida: True**

**Islower** verifica si el texto tiene todas las letras en Minusculas.

**print(texto.lower().islower()) # Salida: True**

**Startswith** verifica si el texto comienza con el elemento proporcionado.

**texto = "Ultimo hola mundo"**

**print(texto.lower().startswith("Ul")) # Salida: True**

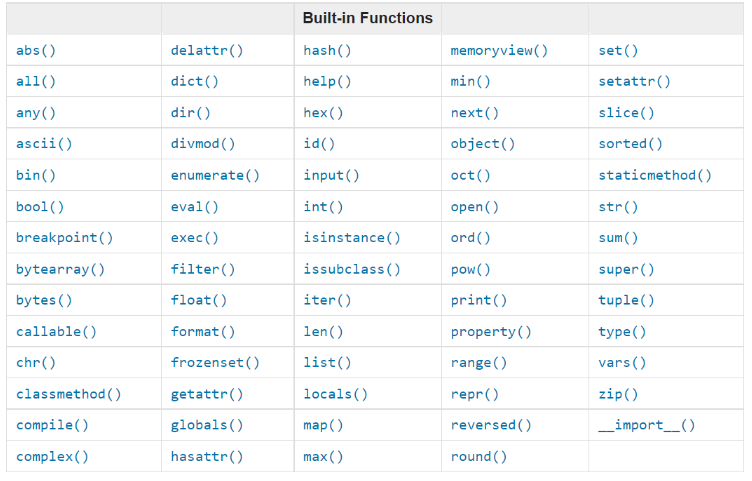
**Endswith** verifica si el texto termina con el elemento proporcionado.

**texto = "Ultimo hola mundo"**

**print(texto.lower().endswith("do")) # Salida: True**

Python diferencia entre mayusculas y minusculas asi que **Ho**, **hO**, **HO** y **ho** no son iguales, cuidado con esto porque puede llevar a errores.

**Funciones del Sistema**



**Listas**

Las listas son una estructura de datos flexible y dinámica que puede contener elementos de diferentes tipos de datos. Pueden cambiar de tamaño dinámicamente, es decir, puedes agregar, eliminar o modificar elementos fácilmente. Pueden contener elementos heterogéneos, es decir, elementos de diferentes tipos de datos. Las listas son parte de la biblioteca estándar de Python y no requieren ningún módulo adicional para ser utilizadas.

Sintaxis para crear una lista vacia

**my\_list = list()**

**my\_list = [ ]**

Sintaxis para crear una lista con elementos

**my\_list = list([1, 2, 3, 4])**

**my\_list = [1, 2, 3, 4]**

Sintaxis para imprimir un elemento de la lista.

**print(my\_list[índice])**

***Funciones de listas***

Esto cuenta cuantas veces aparece el mismo elemento en la lista

**print(my\_list.count(elemento))**

**Count** cuenta cuantas veces aparece el mismo elemento en la lista

**print(my\_list.count(elemento))**

**Append** agrega un elemento al final de la lista

**print(my\_list.append(elemento))**

**Insert** agrega un elemento en la lista con el indice que le indiquemos, haciendo que los demas elementos se desplacen.

**print(my\_list.insert(índice, elemento))**

**Remove** elimina el elemento seleccionado, si el elemento se repite varias veces en la lista, elimina el primero que coincida.

**print(my\_list.remove(elemento))**

**Pop** elimina el ultimo elemento de la lista y lo devuelve, si indicamos el indice elimina y devuelve el elemento en ese indice. Este elemento tambien se puede guardar en una variable extra.

**lista = [1, 2, 3, "Cuatro"]**

**poped = lista.pop() Poped vale = Cuatro**

**print(lista) Salida: [1, 2, 3]**

No importa en que contexto uses el pop, luego de ejecutarse la lista dejara de tener ese elemento, no importa si lo usas para indicar una variable, dentro del print, o por fuera, todos estos modifican directamente a la lista.

**lista = [1, 2, 3, "Cuatro" ]**

**lista.pop() Ahora la lista es: [1, 2, 3]**

**print(lista.pop()) Ahora la lista es: [1, 2]**

**Salida: 3**

**poped = lista.pop() Ahora la lista es: [1]**

**Del** elimina un elemento de una lista y no lo retorna.

**del lista[índice]**

**Clear** elimina todos los elementos de la lista.

**lista = [1, 2, 3, "Cuatro"]**

**lista.clear()**

**print(lista) Salida: []**

**Como modificar un elemento**

**lista = [1, 2, 3, "Cuatro"]**

**lista[1] = "Dos"**

**print(lista) Salida: [1, Dos, 3, Cuatro]**

**Copy** Copia el valor actual de la lista y lo guarda en otra variable. Es una manera eficiente de crear copias de seguridad de la lista actual.

**nueva\_lista = lista.copy()**

**Index** Devuelve el indice del elemento proporcionado.

**print(lista.index(elemento))**

**Reverse** Invierte el orden de los elementos en la lista. Hay que declararla ante de usarla, no se puede ejecutar dentro de un print.

**lista.reverse()**

**print(lista)**

**Salida : [Cuatro, 3, 2, 1]**

**Sort** Ordena los elementos de menor a mayor.

**lista.reverse()**

**print(lista) Salida : [Cuatro, 3, 2, 1]**

**Sub-Listas**

Podemos crear una sub-lista indicando desde que elemento hasta que elemento de la lista padre queremos que tenga. Este ira desde el primer indice indicado hasta un elemento antes del segundo indice indicado.

**lista = [1, 2, 3, 4, 4.5, 6, 7, 8, 9]**

**sub-lista = [1:5]**

**print(sub-lista) Salida : [2, 3, 4, 4.5]**

**Del vs Remove**

**Del** elimina el elemento que esté en el indice indicado, osea que elimina el elemento en el indice independientemente de que valor tenga. **Remove** elimina el primer elemento que encuentre con el valor indicado. En resumen, **DEL** elimina según el indice y **REMOVE** elimina el primer elemento que tenga el valor que queremos eliminar.

**TUPLAS**

Son parecidas a las listas pero, las tuplas son inmutables, lo que significa que una vez creada, no sepueden modificar sus elementos. Esto hace que solo tengan dos metodos para utilizar: **Count** e **Index**.

Sintaxis para crear una tupla vacia

**my\_tuple = tuple()**

**my\_tuple = ( )**

Sintaxis para crear una tupla con elementos

**my\_tuple = tuple((1, 2, 3, 4))**

**my\_tuple = (1, 2, 3, 4)**

Las tuplas estan diseñadas para ser seguras ya que no se pueden cambiar sus elementos ni modificarlos de ninguna manera. Pero en un caso extremo que necesitemos cambiar o alterar un tupla, podemos cambiar tu tipo a lista, modificarla y luego volver a convertirla en una tupla.

**my\_tuple = (1, 2, 3, 4)**

**my\_tuple = list(my\_tuple) # De esta manera podemos**

**my\_tuple[1] = "Dos" # modificar una tupla**

**my\_tuple.insert(2, "Tres")**

**my\_tuple = tuple(my\_tuple)**

**SETS**

Los sets tienen una estructura desordenada, y no permite elementos repetidos. Asi que el orden del set cambiara de forma aleatoria cada vez que agreguemos o modifiquemos algun elemento

Sintaxis para crear una SET vacio

**my\_set = set()**

**my\_set = { } --> Esto en primer lugar dara**

**que es un diccionario**

Sintaxis para crear una tupla con elementos

**my\_set = set({1, 2, 3, 4})**

**my\_set = {1, 2, 3, 4}**

Add añade un elemento al set

**my\_set.add()**

ghjgjghj