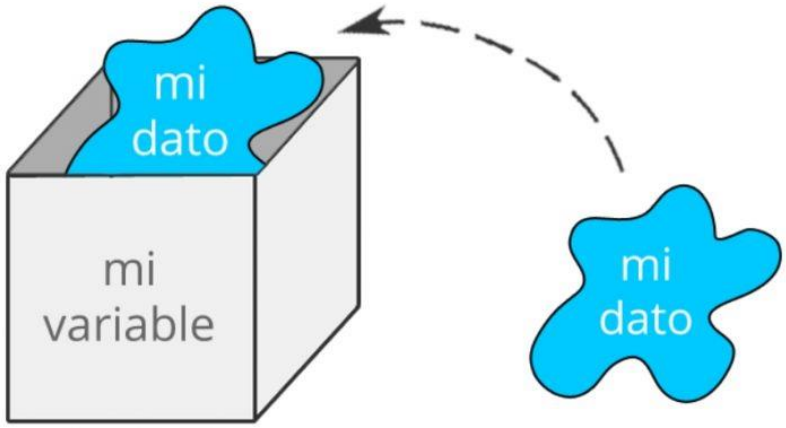
VARIABLES Y TIPOS DE DATOS

Variable

Una variable es un espacio en memoria, donde podemos guardar y recuperar un dato que utilizara nuestro programa. En ella se pueden almacenar distintos tipos de datos.

Las variables pueden llamarse de cualquier forma, siempre y cuando no tengan espacios entre ellas, o no pertenezcan a palabras reservadas del sistema.



Ejemplo

entero -> esta variable no contiene espacio entre ella ni es una palabra reservada.

nombre -> esta variable no contiene espacios entre ella.

print -> **esta variable es una palabra reservada del sistema.**

ciudad **SI**

codigo postal **NO (tiene un espacio entre las 2 palabras)**

chocolate **SI**

yerba_buena **SI (tiene un guión bajo entre medio de las 2 palabras)**

Print **SI (tiene la letra "p" en mayúscula)** → A pesar de que se llame igual que la palabra reservada. En python las variables se pueden diferenciar con mayúsculas y minúsculas.

¿Qué son los tipos de datos?

Es la **clasificación de un dato** según sus **características**.

Una **hoja de cuaderno** y una **hoja A4** para una **impresora**, **ambas** pertenecen a los **elementos de papel**.

Lo mismo sucede cuando alguien quiere reciclar: “voy separando cada elemento según su material.”

¿Qué tipos de datos hay en Python?

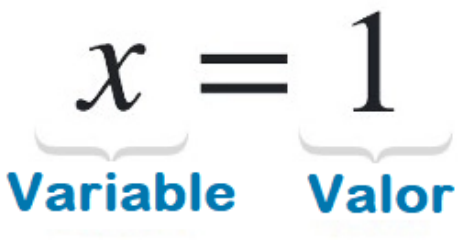
Enteros (Integer} y Largos (Long): **números** que no tienen decimales en sus cifras. Pueden ser tanto negativos como positivos. Los largos pueden almacenar más dígitos.

Flotantes o Reales (Float}: En este caso, estos números si admiten **decimales**.

Cadenas de Caracteres (String}: Es una **cadena de caracteres** de los cuales pueden contener letras, números, signos y símbolos.

Booleanos (Boolean}: Pueden sólo tener **2 valores**, *true* (Verdadero} y *false* (Falso}. Sirven para condicionales.

En Python los tipos de datos pueden ser directamente instanciados sin necesidad previa de decidir qué **tipo de dato** va a ser.



Ejemplo

```
entero = 5
cadena  = "Hola mundo"
flotante  =     2.3
booleano =    False
```

Podemos ver su valor utilizado la función **print**.

```
print(entero)
print(cadena)
print(flotante)
print(type(booleano)) // Devuelve: boolean
```

Podemos utilizar la función **type** para averiguar el **tipo de dato**.

Operaciones sobre números

Para cada tipo de dato tenemos diferentes operaciones que podemos realizar a cada uno.

Números: integer, float, long.

Símbolo	Significado	Ejemplo	Resultado
+	Suma	a = 10 + 5	a es 15
-	Resta	a = 12 – 7	a es 5
-	Negación	a = -5	a es -5
*	Multiplicación	a = 7 * 5	a es 35
**	Exponente	a = 2 ** 3	a es 8
/	División	a = 12.5 / 2	a es 6.25
//	División entera	a = 12.5 / 2	a es 6.0
%	Módulo	a = 27 % 4	a es 3

Operaciones sobre Cadenas de Caracteres

Dentro de estas operaciones, la única operación aritmética que es válida es la "+". Lo que hace es concatenar 2 cadenas de caracteres (unir).

```
cadena_1  = " hola "

cadena_ 2 =  "  mundo "

cadena    = cadena_1 + cadena_ 2   #     Resultado:      " hola mundo "
```

Estructuras complejas

Python, posee además de los tipos ya vistos, 4 tipos más complejos, que admiten una colección de datos.

Estos tipos son:

Tuplas

Conjunto

Listas

Diccionarios

Estos tres tipos, pueden almacenar colecciones de datos de diversos tipos y se diferencian por su sintaxis y por la forma en la cual los datos pueden ser manipulados.

Tuplas

Una tupla es una variable que permite almacenar varios datos inmutables (no pueden ser modificados una vez creados) de tipos diferentes. Es una estructura ordenada y puede haber elementos repetidos:

mi_tupla = (‘cadena de texto’, 15, 2.8, ‘otro dato’, 25, 15)

Listas

Una lista es similar a una tupla con la diferencia fundamental de que permite modificar los datos una vez creados. Son estructuras ordenadas y pueden contener elementos repetidos.

mi_lista = [‘cadena de texto’, 15, 2.8, ‘otro dato’, 25]

Conjuntos

Un conjunto es una colección NO ordenada de objetos únicos. No mantienen un orden específico de elementos, y no se pueden acceder mediante índices.

mi_conjunto = {‘cadena de texto’, 15, 2.8, ‘otro dato’, 25}

Diccionarios

Mientras que a las listas y tuplas se accede solo y únicamente por un número de índice, los diccionarios permiten utilizar una clave para declarar y acceder a un valor. Los diccionarios no mantienen un orden específico para sus elementos:

```
mi_diccionario = { ‘clave_ 1’ : valor_1 , ‘clave_ 2’ : valor_2 , \
‘clave_7’ : valor_7}
```

Asignación Múltiple

Otra de las ventajas que Python nos provee, es la de poder asignar en una sola instrucción, múltiples variables.

```
a, b, c    = ‘string’, 15', True
print(a)   # Salida: string
print(b)   # Salida: 15
print(c)   # Salida: True
```

La asignación múltiple de variables, también puede darse utilizando como valores, el contenido de una tupla o una lista