



Variables y tipos de datos



UNIVERSIDAD EAFIT

Predeterminados del lenguaje

tipos de datos básicos Números, texto, variables de verdad (bool), etc.

estructuras de datos Podemos hacer "conjuntos" de cosas y agruparlas de formas específicas. ¡Y vienen con funcionalidades propias! Ejemplo: listas.

funciones propias

Ejemplo: print(), type(), etc.







Variables

En un lenguaje de programación, a los datos se los guarda en forma de variables. A cada variable debemos darle un nombre único que la identifique:

```
In [ ]: a = 5
In [ ]: un_nombre_cualquiera = 12.7
In [ ]: b = 'Hola!'
In [ ]: nueva_variable = True
```

A estas **variables** pueden se le pueden asignar distintos **tipos de datos**.







Variables y tipos de datos

Python identifica automáticamente el **tipo de dato** de cada variable. Esto resulta muy cómodo para trabajar.

Pero debemos ser cuidadosos, **a veces** el tipo asignado automáticamente **no es el que esperamos ...**



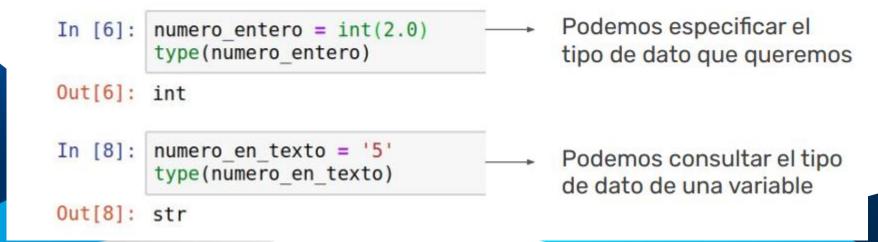




Tipos de datos

¿Podemos pasar de un tipo de dato a otro?

¡Sí! La **solución** es ser explícitos si deseamos que nuestra variable sea de algún tipo en particular.











Tipos de datos

Enteros	Floats	Strings	Booleanos
Son los números que usamos para contar, el 0 y los negativos	Son los números "con coma" Se introducen usando puntos	Texto Se introducen entre comillas dobles, "", o simples, ".	Variables de "verdad": verdadero o Falso
-1 0 1 2	5.1 -1.3 1.0 10.0	"Hola Mundo" "A" 'Mi nombre es Esteban'	True False 1 == 2 1 == 1
[1]: type(3)	[1]: type(3.0)	[1]: type("Hola")	[1]: type(2==2)

Aliados:







Operaciones básicas entre Enteros y Floats

In	[42]:	<pre>x = 3 y = 1.5 print(x/y)</pre>
		2.0

In [43]:
$$x = 2$$

 $y = 3$
print(x**y)

Operación	Operador	Ejemplo
Suma	+	3 + 5.5 = 8.5
Resta	(4)	4 - 1 = 3
Multiplicación	*	3 * 6 = 18
Potencia	**	3 ** 2 = 9
División (cociente)	/	15.0 / 2.0 = 7.5
División (parte entera)	//	15.0 // 2.0 = 7
División (resto)	%	7 % 2 = 1

Aliados:





UNIVERSIDAD EAFIT®

Listas y Loops

Una estructura de dato muy importante en Python son las **listas**. Una lista consiste en una serie de elementos ordenados:

Las listas se definen con corchetes []









Operaciones con listas

Las listas se pueden **sumar** entre sí (se **concatenan**). También se les puede agregar un elemento nuevo mediante el método '.append()'

```
In [52]: lista_1 = [2, 4.7, True, 'Texto']
    lista_2 = [42, 42]
    lista_1 + lista_2

Out[52]: [2, 4.7, True, 'Texto', 42, 42]

In [53]: lista_1 = [2, 4.7, True, 'Texto']
    lista_1.append('Un nuevo elemento')
    lista_1

Out[53]: [2, 4.7, True, 'Texto', 'Un nuevo elemento']
```









Operaciones con listas

```
In [55]: lista 1 = [2, 4.7, True, 'Texto']
len(lista_1)

Out[55]: 4

In [56]: lista_2 = [0, lista_1, 'Mas texto']
len(lista_2)

Out[56]: 3
```

Las listas tienen un largo determinado por su cantidad de elementos. Se consulta mediante la función **len()**.

```
In [59]: lista_vacia = []
len(lista_vacia)

Out[59]: 0

In [60]: lista_vacia.append(42)
lista_vacia.append('un segundo item')
print(lista_vacia)
        [42, 'un segundo item']
```

Se pueden generar listas vacías y luego ir agregándole elementos a medida que una lo precise.







LOOPS

Los **Loops** en programación son bloques de código que, dadas ciertas condiciones, se repiten una cierta cantidad de veces.

El **For** es un tipo de **Loop** que repite un bloque de código tantas veces como elementos haya en una **lista** dada:





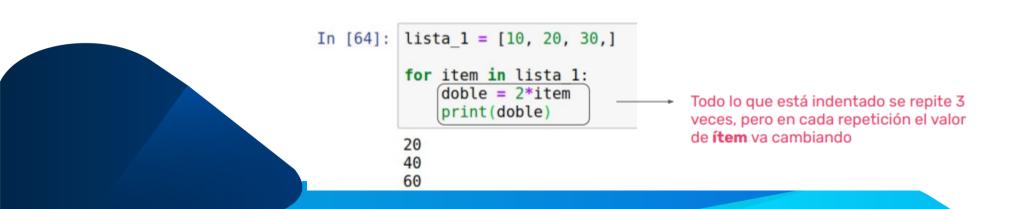




UNIVERSIDAD EAFIT

Listas y Loops

En cada repetición, la variable **item** (podría tener cualquier nombre) va tomando el valor de cada un de los elementos de la lista dada.









UNIVERSIDAD EAFIT®

Listas y Loops

Las listas pueden contener texto. Veamos un ejemplo donde creamos una nueva lista.







UNIVERSIDAD EAFIT

Listas y Loops

El **While** es un tipo de **Loop** que repite un bloque de código hasta que una dada condición se deje de cumplir. Esta condición debe expresarse como una variable **Booleana**.

```
In [68]: numero = 1

while (numero < 5):
    print(numero)
    numero = numero + 1

In [69]: 4 < 5
Out[69]: True

3    Se cumple hasta que número
4    Vale 5 y ya no entra al loop.

In [70]: 5 < 5
Out[70]: False</pre>
```









- 1. Comentar el código en voz alta
- 2. Modificar los códigos
- 3. Visitar websites de comunidades: Github, StackOverflow, Kaggle, Medium.







Recursos adicionales

https://learnxinyminutes.com/docs/python3/

https://www.tutorialsteacher.com/python







¡Gracias!

Aliados:



