

Científico de Datos

Nivel Básico

Aliados:



Microsoft

Vigilada Mineducación



Advanced analytics for business

Tema:

Variables y tipos de datos

Aliados:



Microsoft

Vigilada Mineducación



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.

Aliados:



Microsoft

Vigilada Mineducación



Advanced analytics for business

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**.



Microsoft

Vigilada Mineducación



Advanced analytics for business

Variables y tipos de datos

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

```
In [ ]: a = 5
```

————→ Entero

```
In [ ]: un_nombre_cualquiera = 12.7
```

————→ Float

```
In [ ]: b = 'Hola!'
```

————→ String

```
In [ ]: nueva_variable = True
```

————→ Boolean

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

Aliados:



Microsoft

Vigilada Mineducación



Advanced analytics for business

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.

```
In [6]: numero_entero = int(2.0)  
        type(numero_entero)
```

Podemos especificar el tipo de dato que queremos

```
Out[6]: int
```

```
In [8]: numero_en_texto = '5'  
        type(numero_en_texto)
```

Podemos consultar el tipo de dato de una variable

```
Out[8]: str
```

Aliados:



Vigilada Mineducación



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]: <code>type(3)</code> [1]: <code>int</code>	[1]: <code>type(3.0)</code> [1]: <code>float</code>	[1]: <code>type("Hola")</code> [1]: <code>str</code>	[1]: <code>type(2==2)</code> [1]: <code>bool</code>

Aliados:

Operaciones básicas entre Enteros y Floats

```
In [42]: x = 3  
y = 1.5  
print(x/y)  
2.0
```

```
In [43]: x = 2  
y = 3  
print(x**y)  
8
```

```
In [44]: x = 10  
y = 3  
print(x%y)  
1
```

Operación	Operador	Ejemplo
Suma	+	$3 + 5.5 = 8.5$
Resta	-	$4 - 1 = 3$
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:



Vigilada Mineducación



Listas y Loops

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

```
In [47]: lista_1 = [2, 4.7, True, 'Texto']  
         type(lista_1)
```

→ Los elementos pueden ser de distintos tipos.

```
Out[47]: list
```

```
In [49]: lista_2 = [0, lista_1, 'Mas texto']  
         print(lista_2)  
[0, [2, 4.7, True, 'Texto'], 'Mas texto']
```

→ Incluso puede haber listas dentro de listas.

Las **listas** se definen con corchetes []

Aliados:



Microsoft

Vigilada Mineducación



Advanced analytics for business

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']
```

Aliados:



Microsoft

Vigilada Mineducación



Advanced analytics for business

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.

Aliados:



Microsoft

Vigilada Mineducación



Advanced analytics for business

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:

```
In [62]: lista_1 = [0, 1, 2, 3]
         for item in lista_1:
             print('Hola.')
```

Hola.
Hola.
Hola.
Hola.

Lo que está dentro del cuerpo del loop se identifica por su **indentación**. En este caso se repite 4 veces, porque la lista tiene 4 elementos

Aliados:



Microsoft

Vigilada Mineducación



Advanced analytics for business

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.

```
In [64]: lista_1 = [10, 20, 30,]
```

```
for item in lista_1:  
    doble = 2*item  
    print(doble)
```

```
20  
40  
60
```

→ Todo lo que está indentado se repite 3 veces, pero en cada repetición el valor de **item** va cambiando

Aliados:



Microsoft

Vigilada Mineducación



Advanced analytics for business

Listas y Loops

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

```
In [66]: lista_nombres = ['Ernesto', 'Camilo', 'Violeta']  
nueva_lista = []  
  
for item in lista_nombres:  
    oracion = 'Mi nombre es ' + item  
    nueva_lista.append(oracion)  
  
print(nueva_lista)  
['Mi nombre es Ernesto', 'Mi nombre es Camilo', 'Mi nombre es Violeta']
```

Estas dos líneas se repiten 3 veces. Le agregamos texto a cada elemento y lo agregamos a una nueva lista.

Aliados:



Microsoft

Vigilada Mineducación



Advanced analytics for business

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
```

1
2
3
4

→ Se cumple hasta que número
vale 5 y ya no entra al loop.

El resultado de esta
comparación es un booleano:

```
In [69]: 4 < 5
```

```
Out[69]: True
```

```
In [70]: 5 < 5
```

```
Out[70]: False
```

Aliados:



Vigilada Mineducación



Tips

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

Aliados:



Vigilada Mineducación



Material complementario (DS)

- <https://learnxinyminutes.com/docs/python3/>
- <https://www.tutorialsteacher.com/python>

Aliados:



Vigilada Mineducación



Contenido asincrónico

- [Azure Cloud Concepts](#)
- [Azure Cloud Services](#)
- Resto de actividades en plataforma Interactiva Virtual

Aliados:



Vigilada Mineducación



¡Gracias!

Aliados:



Microsoft

Vigilada Mineducación

