



# Introducción a Python

Operadores y librerías

***Distinguir los tipos de datos  
y sentencias básicas  
del lenguaje para la  
construcción de programas.***

- Unidad 1:  
Introducción a Python
- Unidad 2:  
Sentencias condicionales e  
iterativas
- Unidad 3:  
Estructuras de datos y funciones



Te encuentras aquí



## ¿Qué aprenderás en esta sesión?

- *Reconoce los operadores matemáticos, lógicos y de comparación para la construcción de expresiones.*
- *Reconoce las sentencias básicas del lenguaje como condicionales y bucles para la construcción de programas.*

¿Qué entendemos por  
operadores matemáticos?



# **/\* Operadores matemáticos \*/**

# Otros operadores matemáticos

Ahora que tenemos un mayor entendimiento de cómo funciona el lenguaje Python, podemos definir formalmente los operadores nativos en Python, donde además de los operadores de **+**, **-**, **\***, y **/** podemos adicionar los siguientes:

```
2 ** 4 # Exponenciación/Potencia. Ej: 2 elevado a 4 es 16
5 // 2 # División Entera. Ej: 5 dividido 2 es 2.
5 / 2 # División con decimales. Ej: 5 dividido 2 es 2.5
5 % 2 # Módulo. Ej: 5 Módulo 2 es 1. 1 es el resto de 5 // 2.
```

# Precedencia de operadores

*"Saber en qué orden se realiza un grupo de operaciones"*

```
x = 10  
y = 5  
z = 2  
print(x - y * z)
```

10-5\*2

10-10

0

0

# Orden de las operaciones

*Cuando dos operaciones tienen el mismo nivel de prioridad, se resuelven de izquierda a derecha.*

Operador	Nombre
**	Exponenciación (potencia)
*, /, //, %	Multiplicación, división y módulo
+, -	Suma y resta



# Operaciones y paréntesis

*Los paréntesis cambian el orden en que preceden las operaciones dando prioridad a las que estén dentro de los paréntesis.*

```
print((10 - 5) * 2)
```

$(10-5)*2$

$5*2$

10

10

**`/* Librerías */`**

# Importar una librería

Las librerías son extensiones del lenguaje que añaden funcionalidades no disponibles en Python nativo.

```
import os # esta es una librería que permite interactuar con el Sistema Operativo
import math # añade más funcionalidades matemáticas
```

Normalmente las importaciones se definen al inicio de un código. Luego para utilizar una funcionalidad de dicha librería se usa la sintaxis: `libreria.metodo()`:

```
os.getcwd() # entrega la ruta actual en la que me encuentro.
math.sqrt(2) # raíz cuadrada de dos
```

Existe una manera alternativa de llamar una función específica de una librería:

```
from math import ceil # importa sólo la función ceil
ceil(3.14) # ceil aproxima al entero superior
```

# Instalar librerías

Una de las ventajas de utilizar Anaconda, es que ya trae pre instaladas muchas de las librerías más famosas en Python, aún así, es sumamente importante entender cómo instalar librerías adicionales. Para ello, se tendrá que utilizar Anaconda Prompt, si se está en Windows, o la Terminal si se utiliza Mac o Linux.

Digamos que queremos instalar la librería **pandas**, aquí, lo mejor es acudir a su [documentación oficial](#), está indicará la manera más apropiada de instalarla, en este caso, mediante '**pip**'.

Installing from PyPI 📄

pandas can be installed via pip from [PyPI](#).

```
pip install pandas
```

# Instalar librerías

## *pip*

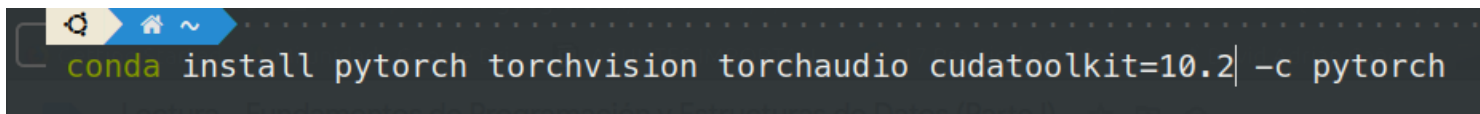
- Es el instalador por defecto de PyPI, el cual es uno de los repositorios de librerías con los que cuenta Python. Mediante **pip install** es posible entonces instalar cualquier librería que resida en este repositorio.
- Pero también conda tiene su propio repositorio, y hay algunas librerías que prefieren este repositorio como por ejemplo Pytorch:

PyTorch Build	Stable (1.8.0)				Preview (Nightly)	
Your OS	Linux		Mac		Windows	
Package	Conda		Pip		LibTorch	
Language	Python				C++ / Java	
Compute Platform	CUDA 10.2		CUDA 11.1		ROCm 4.0 (beta)	
Run this Command:	<b>NOTE:</b> Python 3.9 users will need to add '-c=conda-forge' for installation conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch					

# Instalar librerías

*pip*

Para el caso de conda, el comando escogido es **conda install**.

A terminal window with a dark background. The top bar shows icons for settings, home, and a shell. The command 'conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch' is entered in a light-colored font.

```
conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch
```

*Una de las ventajas de pip es que normalmente es el método que disponibiliza las actualizaciones con mayor rapidez, en cambio Conda tiende a demorar un poco más. La buena noticia es que las últimas versiones de Anaconda son compatibles con ambos métodos, por lo tanto, lo mejor es revisar la documentación de cada librería y ver cuál es su método preferido, porque sin importar cuál sea, éste será compatible con Anaconda.*

# Ejercicio guiado



# Contando calorías

Algunos alimentos también consideran un grado alcohólico, donde cada grado alcohólico aporta 7 calorías.

Modifica el programa de manera tal que se pueda ingresar el grado alcohólico de un alimento y se entregue el número de calorías que este posee.

Para esto, descarguemos el archivo "***Ejecución guiada - Contando calorías.zip***"





# Contando calorías

## Solución - Paso 1

### Solicitar datos para el cálculo

Lo primero que debemos hacer siempre es ingresar en este caso los datos que nos permitirán llevar a cabo el cálculo. En este caso utilizaremos **input()** y dado que los datos pueden contener decimales, es importante transformarlos en el tipo de dato correcto, es decir, **float()**.

```
proteina = float(input("Ingrese los gr de proteina:\n>"))  
carbo = float(input("Ingrese el los de Carbohidrato:\n>"))  
grasa = float(input("Ingrese el los de Grasa:\n>"))
```

# Contando calorías

## *Solución - Paso 2*

### Calcular la fórmula de calorías

Para esta etapa basta con entender los datos. Se entregan 4 calorías por cada proteína y carbohidrato, además de 9 calorías por grasas, y eso se puede expresar de esta manera:

```
calorias = 4 * (proteina + carbo) + 9 * grasa
```



# Contando calorías

## Solución - Paso 3

### Entregar el resultado en el formato correcto

Según lo que se indica las calorías totales deben redondearse al entero superior. Como vimos anteriormente podemos aplicar dicha operación con la librería **math** y la función **ceil**:

```
import math
print(f'Las calorías totales del producto son: {math.ceil(calorias)}')
```



# Contando calorías

## Solución - Paso 3

El código final y ordenado, quedaría de la siguiente manera:

```
# importa librería math
import math

# Solicitud de Inputs
proteina = float(input("Ingrese los gr de proteína:\n>"))
carbohidratos = float(input("Ingrese el los de Carbohidrato:\n>"))
grasa = float(input("Ingrese el los de Grasa:\n>"))

# cálculo de calorías
calorias = 4 * (proteina + carbohidratos) + 9 * grasa

# entregar output en el formato solicitado
print(f'Las calorías totales del producto son: {math.ceil(calorias)}')
```



# Contando calorías

## Solución - Paso 4

Ejecutemos el código

```
python calorías.py  
Ingrese los gr de proteína:  
>1.9  
Ingrese el los de Carbohidrato:  
>9.2  
Ingrese el los de Grasa:  
>7  
Las calorías totales del producto son: 108
```



# Contando calorías

## Solución - Paso 4

Si ingresamos los datos por porción (segunda columna) notamos que los cálculos cuadran a la perfección.

```
python calorías.py
Ingrese los gr de proteína:
>9.6
Ingrese el los de Carbohidrato:
>46
Ingrese el los de Grasa:
>35
Las calorías totales del producto son: 538
```

Realizando los mismos cálculos para los valores cada 100 gr (primera columna), también obtenemos resultados correctos.

¿Qué son las librerías  
y para qué sirven?





## Próxima sesión...

- *Desafío evaluado.*



**{desafío}**  
**latam\_**

*Academia de  
talentos digitales*

