



Conociendo Python (Parte I)

Clase sincrónica

Mapa general de la carrera

Revisión modular - Data Science

SQL para el
análisis de
datos

Programación
con Python
para el análisis
de datos

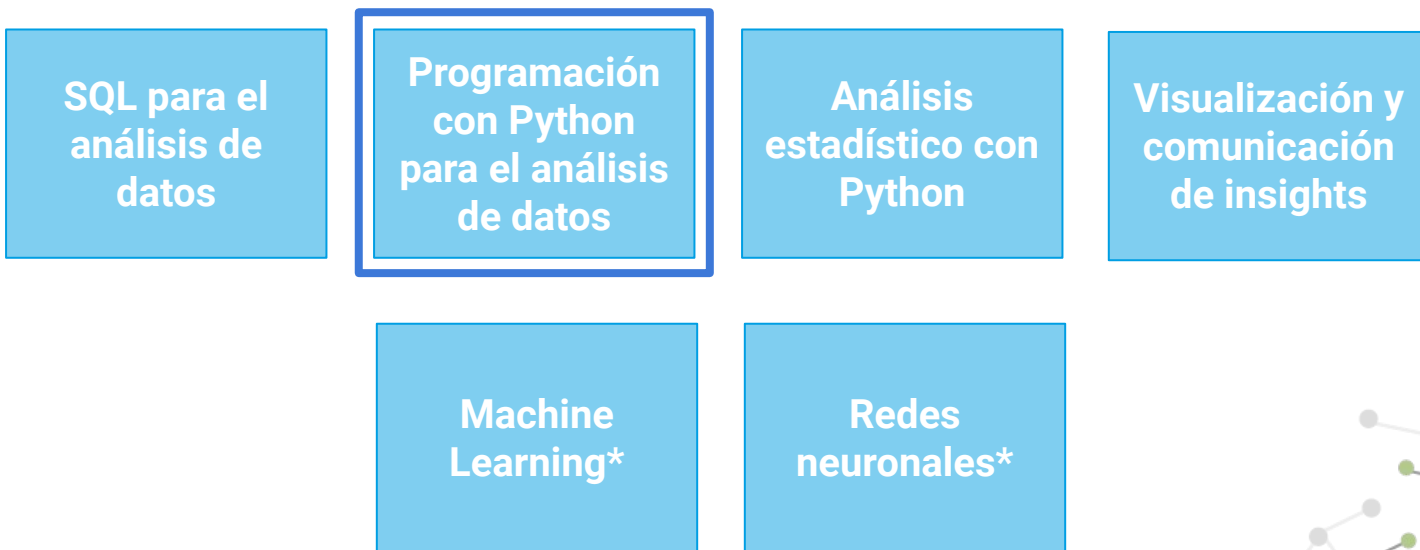
Análisis
estadístico con
Python

Visualización y
comunicación
de insights



Mapa general de la carrera

Revisión modular - Data Science



Programación con Python para el análisis de datos

Unidad	Clases (sincrónico)	Autoaprendizaje (asincrónico)	Tutoría (sincrónico)
Conociendo Python (Parte I)	2 horas	Desde 6 horas	2 horas
Conociendo Python (Parte I)	2 horas	Desde 6 horas	2 horas
Tipos y estructura de datos (Parte I)	2 horas	Desde 6 horas	2 horas
Tipos y estructura de datos (Parte II)	2 horas	Desde 6 horas	2 horas
Manipulación y transformación de datos (Parte I)	2 horas	Desde 6 horas	2 horas
Manipulación y transformación de datos (Parte I)	2 horas	Desde 6 horas	2 horas
Manipulación y transformación de datos (Parte II)	2 horas	Desde 6 horas	2 horas
<i>Prueba</i>	<i>0 horas</i>	<i>Desde 6 horas</i>	<i>0 horas</i>
<i>Receso</i>	<i>0 horas</i>	<i>0 horas</i>	<i>0 horas</i>

¿Qué aprenderemos en este módulo?

Al finalizar el módulo serás capaz de aplicar técnicas de obtención, limpieza y preparación de datos a través de Python para resolver problemas.



**Utiliza herramientas,
comandos y estructuras
básicas de Python para la
creación de programas
sencillos.**

- Unidad 1: Conociendo Python
(Parte I)
(Parte II)
- Unidad 2: Tipos y estructura de datos
(Parte I)
(Parte II)
- Unidad 3: Manipulación y transformación de datos
(Parte I)
(Parte II)
(Parte III)



Te encuentras aquí



¿Qué aprenderás en esta sesión?

- *Utilizar Python para resolver problemas simples de acuerdo a requerimientos.*

¿Qué es Python?
¿Lo han oído nombrar?
¿Conocen algunas de
sus características?

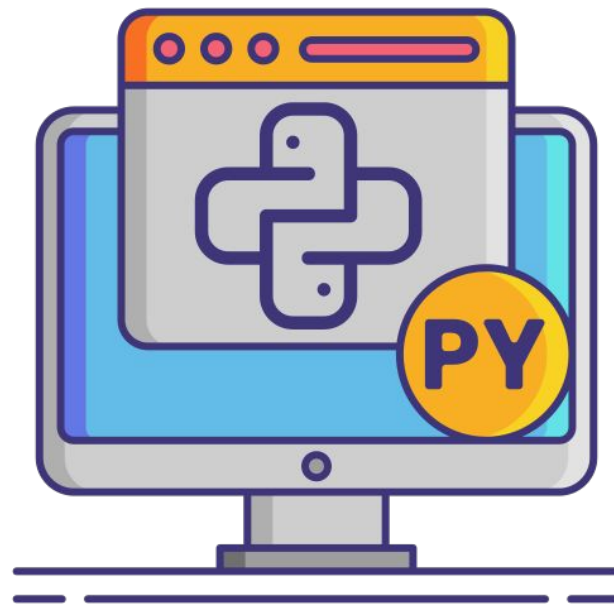


/*Aproximándonos a Python*/

¿Qué es Python?

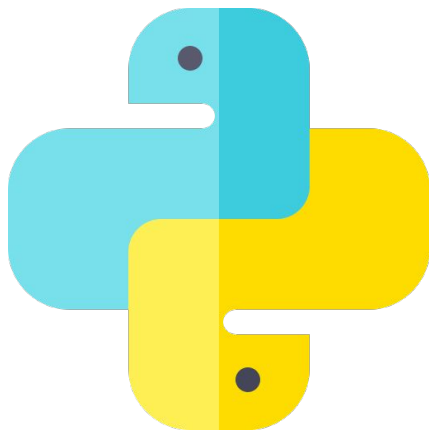
Aspectos esenciales

1. Orientado a objetos.
2. Alto nivel.
3. Gran cantidad de bibliotecas de análisis de datos (NumPy, Pandas, Scikit-Learn) y de tipos variados, por lo que cuenta con gran versatilidad.
4. Escalabilidad y portabilidad (Multiplataforma).
5. Gratuito y colaborativo.
6. Fácil de aprender.



Fuente: FlatIcon

Python permite



Fuente: FlatIcon

- Construir de forma sencilla aplicaciones web con manejo de bases de datos.
- Hacer análisis de datos y visualización de estos.
- Realizar web-scraping (Captura de datos de una página web).
- Crear videojuegos.
- Crear aplicaciones de escritorio.

Áreas donde se utiliza Python

Desarrollo Web

- Páginas como YouTube, Instagram y Google implementan Python en sus servicios.
- El lenguaje ofrece librerías como Django y Flask que permiten desarrollar servicios webs complejos dentro de un marco de trabajo sencillo.

Análisis y Ciencia de Datos

- Dada la simpleza sintáctica de Python, existe un gran desarrollo de librerías de análisis y preprocesamiento de datos por parte de la academia e industria para agilizar las rutinas de análisis.
- Librerías como Scikit-Learn y Tensorflow dominan la implementación de modelos predictivos en distintas áreas.

Empresas que utilizan Python

Python en el mundo real



Fuente: LinkedIn - Power of Python

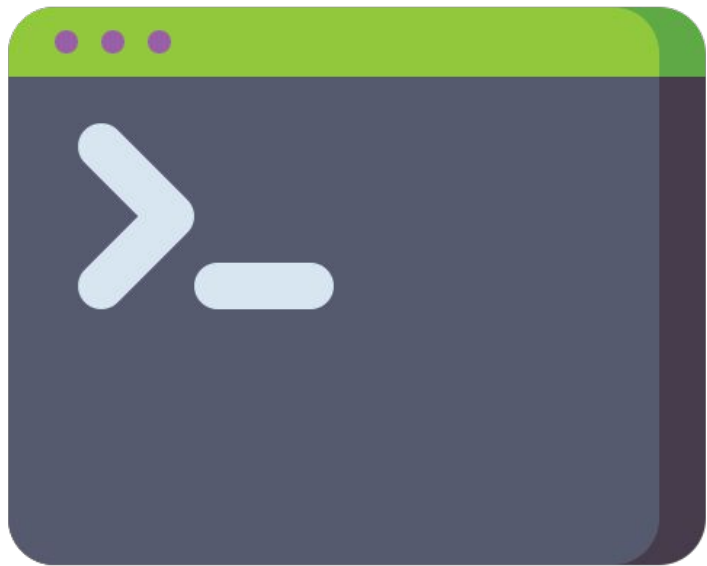
/*¿Cómo utilizar Python?*/

¿Cómo utilizar Python?

Uso en consola

Una primera alternativa para utilizar Python es directamente en consola o a través de un Entorno de Desarrollo, que puedes descargar (hay variadas opciones gratuitas).

El uso en consola es el más “primitivo” para Python y se suelen preferir otros entornos, pero es necesario saber utilizarlo ya que es el más básico.



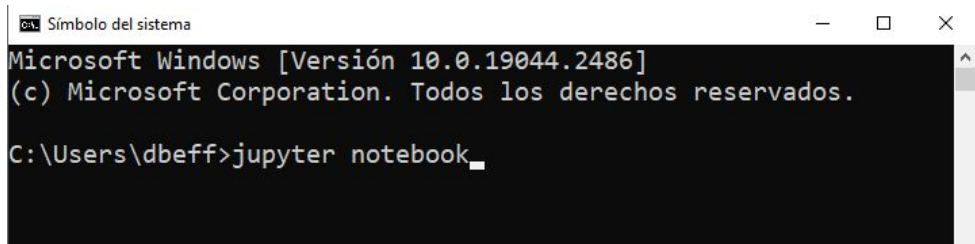
¿Cómo utilizar Python?

Jupyter Notebook

Jupyter Notebook es una **aplicación web** que permite crear y compartir documentos que contienen **código ejecutable, texto, visualizaciones** y otros elementos interactivos.



Fuente: www.jupyter.org

A screenshot of a Windows command prompt window. The title bar says 'Símbolo del sistema'. The text inside the window shows the Windows version and copyright information, followed by the command 'C:\Users\dbeff>jupyter notebook_'.

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19044.2486]
(c) Microsoft Corporation. Todos los derechos reservados.

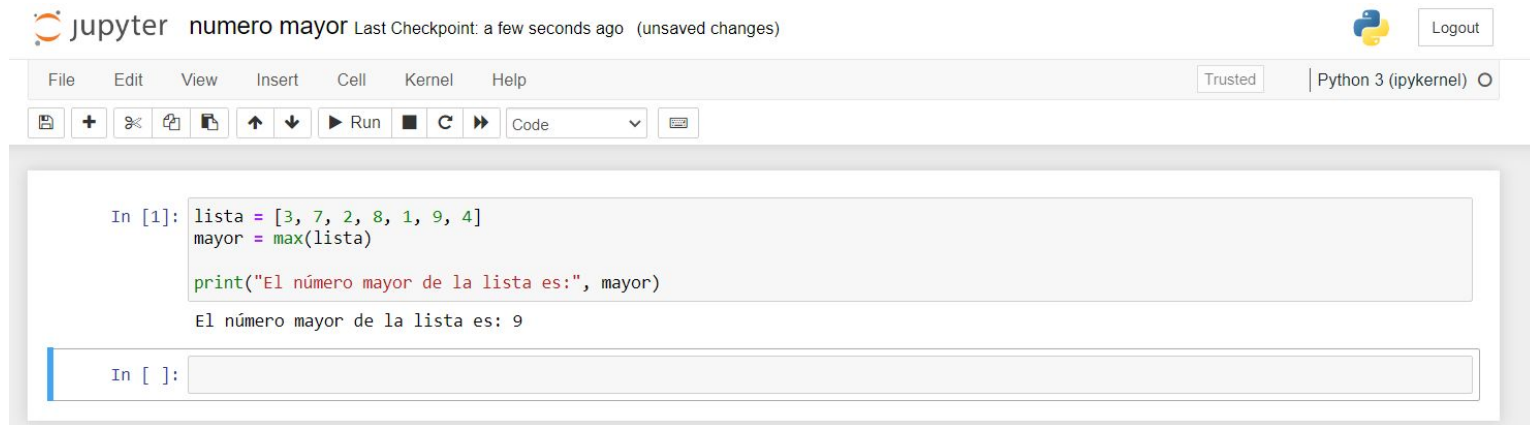
C:\Users\dbeff>jupyter notebook_
```

Para lanzar la aplicación desde el terminal o cmd.

Jupyter Notebook

Celdas de código

Las celdas de código se utilizan para escribir y ejecutar código en Jupyter Notebook, mientras que las celdas de markdown se utilizan para agregar texto descriptivo, comentarios y anotaciones.



The screenshot displays the Jupyter Notebook interface. At the top, the title bar shows 'jupyter numero mayor' and 'Last Checkpoint: a few seconds ago (unsaved changes)'. The right side of the title bar includes a Python logo, a 'Logout' button, and a 'Trusted' status indicator. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. A toolbar below the menu bar contains icons for saving, adding, deleting, and running cells, as well as a dropdown menu set to 'Code'. The main area shows a code cell with the following Python code:

```
In [1]: lista = [3, 7, 2, 8, 1, 9, 4]
        mayor = max(lista)

        print("El número mayor de la lista es:", mayor)
```

Below the code, the output is displayed: 'El número mayor de la lista es: 9'. At the bottom, there is an input prompt 'In []:' followed by an empty text box for entering new code.

Fuente: Desafío Latam

Google Colab

Jupyter Notebooks en la nube



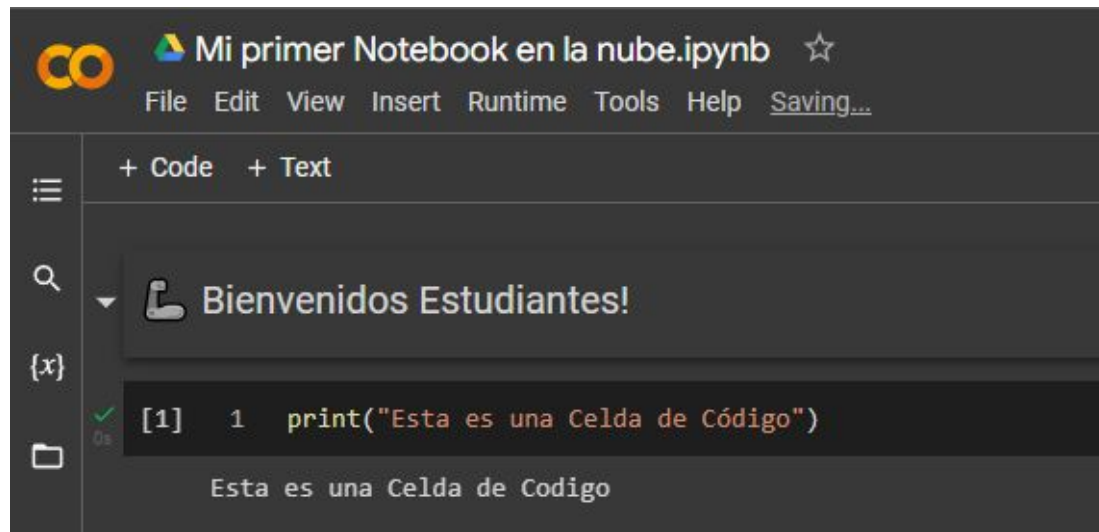
- Se ejecuta localmente en el equipo.
- Solo utiliza la CPU local.
- Requiere la configuración de un servidor y la exposición de ese servidor a través de una dirección IP o un nombre de dominio.



- Se ejecuta en la nube.
- Proporciona acceso gratuito a GPUs para acelerar los cálculos.
- Permite compartir y colaborar en notebooks con otros usuarios a través de un enlace compartido.

Google Colab

Cómo se ve la interfaz



Fuente: Desafío Latam

/*Interacción*/

Interactuando con el usuario

input y print



Recibe información o datos que el usuario ingresa desde el teclado.



Muestra información o datos en la pantalla

Ejercicio: Interactuando con el usuario



Ejercicio

print / input

1. Abre una terminal o línea de comandos.
2. Crea un archivo de texto con la extensión **.py**, por ejemplo, **hola_mundo.py**.
3. Abre el archivo de texto y escribe el siguiente código:

```
nombre = input("Introduce tu nombre")  
print("Hola " + nombre + "!")
```

4. Guarda el archivo.
5. En la terminal, navega hasta la carpeta donde se encuentra el archivo **hola_mundo.py**.
6. Ejecuta el siguiente comando:

```
python hola_mundo.py
```



Interactuando con el usuario

Una alternativa

Podemos intercalar valores de variables en un comando print, como se muestra:

```
nombre=input("Ingrese su nombre")
edad=input("Ingrese su edad")
print(f"Hola, {nombre}! Tienes {edad} años")
```


Interactuando con el usuario

Comentarios

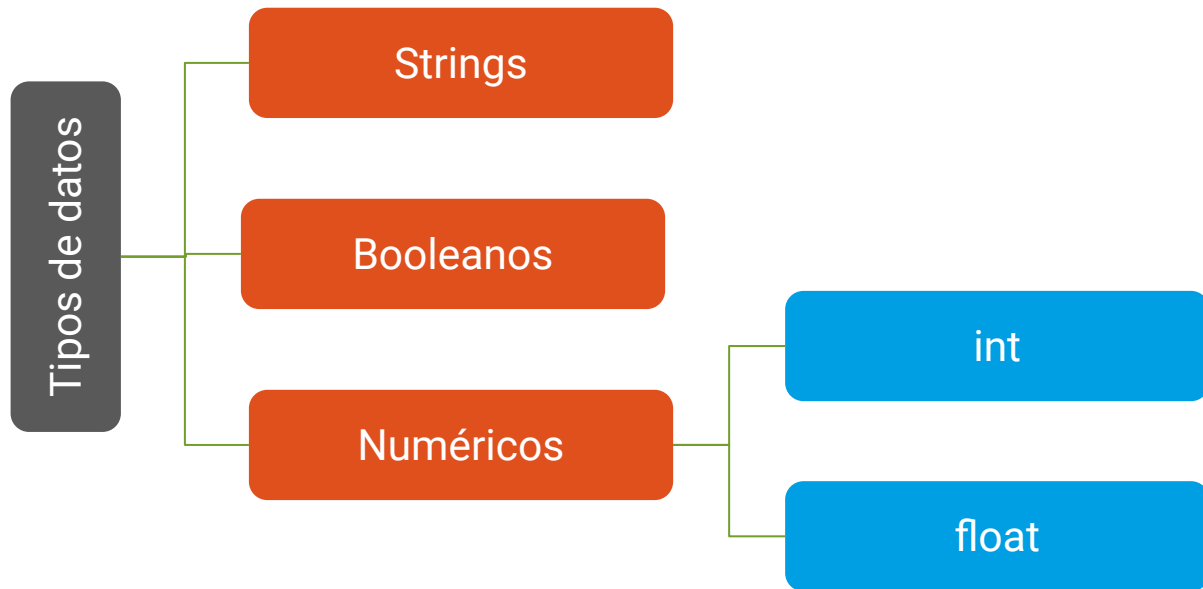
- Los comentarios ayudan a precisar qué estamos haciendo, no se “ejecutan”.
- Permiten explicar la lógica o pasos del programa.
- Ubicarnos y “navegar”.
- Trabajar entre dos o más personas.

```
edad=input("Ingrese su edad") #solicitamos su edad
print(f"Tienes {edad} años") #esto inserta la variable edad
```

/*Tipos de datos*/

Tipos de datos

Tipos básicos



Tipos de datos

Arreglos

- **Listas (lists):** Secuencia de elementos ordenados, que pueden ser de cualquier tipo de datos.
- Las listas se crean usando corchetes `[]` y los elementos se separan por comas.
 - Ejemplos: `[1, 2, 3]`, `["manzana", "pera", "naranja"]`, `[1, "hola", True]`, etc.
- Para acceder a los elementos se usa el **índice**, que siempre **comienza en cero**.

```
mi_lista = ["manzana", "banana", "naranja", "piña"]  
print(mi_lista[0])  
print(mi_lista[2])  
print(mi_lista[3])
```

```
manzana  
naranja  
piña
```

Tipos de datos

Diccionarios

- Colección de pares clave-valor, donde cada clave se asocia con un valor.
- Los diccionarios se crean usando llaves {} y cada par clave-valor se separa por comas.

```
mi_diccionario = {"nombre": "juan", "edad": 25, "ciudad": "santiago"}  
print(mi_diccionario["nombre"])  
print(mi_diccionario["edad"])  
  
#Agreguemos un nuevo par clave - valor  
mi_diccionario["profesion"]="cocinero"
```

/*Operaciones y métodos*/

Operaciones y métodos

Strings

```
print("Carlos" + "Santana")  
print("Carlos " + "Santana")  
print("15" + "23")  
print(3 * "Carlos")  
print(5 * "12")  
print("Santana".count("a"))  
print("Santana".upper())  
print(len("Carlos Santana"))
```

```
CarlosSantana  
Carlos Santana  
1523  
CarlosCarlosCarlos  
1212121212  
3  
SANTANA  
14
```

Operaciones y métodos

Aritméticas

```
print(3+5)
print(7-3)
print(3*5)
print(3/5)
print(9//4)
print(15%6)
print(2**3)
```

8
4
15
0.6
2
3
8



¿Cómo podemos encontrar el número más grande dentro de una lista?

Solución

```
lista = [3,7,2,8,1,4]  
mayor = max(lista)  
print(f"El número mayor es {mayor}")
```

El número mayor es 8



Errores con operaciones

Mezclando variables

Es importante mencionar que las listas no exigen que todos sus elementos sean del mismo tipo, lo que puede generar un error al aplicar funciones que operan sobre datos numéricos:

```
lista = [3,7,2,8,1,4, "perro"]
mayor = max(lista)
print(f"El número mayor es {mayor}")
```

```
-----
TypeError                                Traceback (most recent call last)
/tmp/ipykernel_35290/4059540410.py in <module>
      1 lista = [3,7,2,8,1,4, "perro"]
----> 2 mayor = max(lista)
      3 print(f"El número mayor es {mayor}")
      4
```

```
TypeError: '>' not supported between instances of 'str' and 'int'
```

Ejercicio guiado “El vendedor”



El vendedor

1. El ejercicio del vendedor es un ejemplo clásico utilizado para enseñar programación en diferentes lenguajes.
2. Pueden existir infinitas variaciones de este ejercicio, dependiendo de los requerimientos.
3. En este caso, crearemos un programa en el que se conoce el valor de un producto y el porcentaje de descuento que se le aplica. El usuario debe ingresar la cantidad de artículos que desea, y el programa le muestra el valor a pagar.



El vendedor

Veremos el siguiente caso:

1. **Definición de variables:** Se establecen dos **variables**, una para el precio del producto y otra para el descuento.
2. **Solicitud de información:** Se utiliza la función "**input**" para solicitar al usuario la cantidad de productos que desea comprar.
3. **Cálculo del precio final:** Se realiza el cálculo multiplicando el precio del producto por la cantidad y luego se aplica el descuento.
4. **Imprimiendo resultados:** Finalmente, se utiliza la función "**print**" para mostrar el precio final.



El vendedor

```
In [1]: # Ejercicio del vendedor

# 1.Definición de variables
precio_producto = 10
descuento = 0.15

# 2.Solicitud de información
cantidad = int(input("¿Cuántos productos quieres comprar? "))

# 3.Cálculo del precio final
precio_final = precio_producto * cantidad * (1 - descuento)

# 4.Imprimiendo resultados
print("El precio final es", precio_final)

¿Cuántos productos quieres comprar? 10
El precio final es 85.0
```



/*Librerías*/

Librerías

Importación de librerías

En Python, las librerías son conjuntos de módulos y funciones predefinidos que se pueden importar en un programa para agregar funcionalidades adicionales.

Para importar una librería en Python, se utiliza la palabra clave **import** seguida del nombre de la librería:

```
import random

num_aleatorio = random.randint(1,10)
print(f"El número generado es {num_aleatorio}")
```

El número generado es 5

Ejercicio: Lanzando una moneda



Ejercicio

Lanzando una moneda

Podemos simular el lanzamiento de una moneda, eligiendo al azar un elemento dentro de una lista, por ejemplo:

```
import random

opciones = ["cara", "sello"]

resultado = random.choice(opciones)

print(f"El resultado es {resultado}")
```

El resultado es sello

Ejercicio Guiado "Club de Mascotas"



Club de mascotas

- Debes descargar el notebook “Club de Mascotas.ipynb” desde la plataforma.
- Ábrelo en tu computador o en Google Colab.
- ¡Sigue las instrucciones para inscribir a tu mascota en el club!



Desafío - Conociendo Python



Desafío

“Conociendo Python”

- Descarga el archivo “Desafío”.
- Tiempo de desarrollo asincrónico: desde 4 horas.
- Tipo de desafío: individual.

¡AHORA TE TOCA A TI! 💪



Ideas fuerza



Python es un **lenguaje de programación de alto nivel**, fácil de usar, y que permite complementar el uso de SQL en bases de datos.



Admite **interacción con el usuario** y trabajar con **diferentes tipos de datos**. Es fundamental distinguirlos para no cometer errores.



Permite utilizarse combinado con el **lenguaje de marcas**. Podemos trabajar offline con **Jupyter Notebook**, o de manera online gracias a **Google Colab**.

¿Qué contenidos de la clase
no te quedaron totalmente
claros?



Recursos asincrónicos

¡No olvides revisarlos!

Para esta semana deberás revisar:

- Guía de contenidos “Conociendo Python (Parte I)”
- Tutorial “Markdown”
- Desafío “Conociendo Python (Parte I)”





Próxima sesión...

- *Construir y utilizar diagramas de flujo y algoritmos.*
- *Utilizar operadores matemáticos y lógicos.*
- *Utilizar estructuras de control de flujo para crear programas.*

{desafío}
latam_

*Academia de
talentos digitales*

