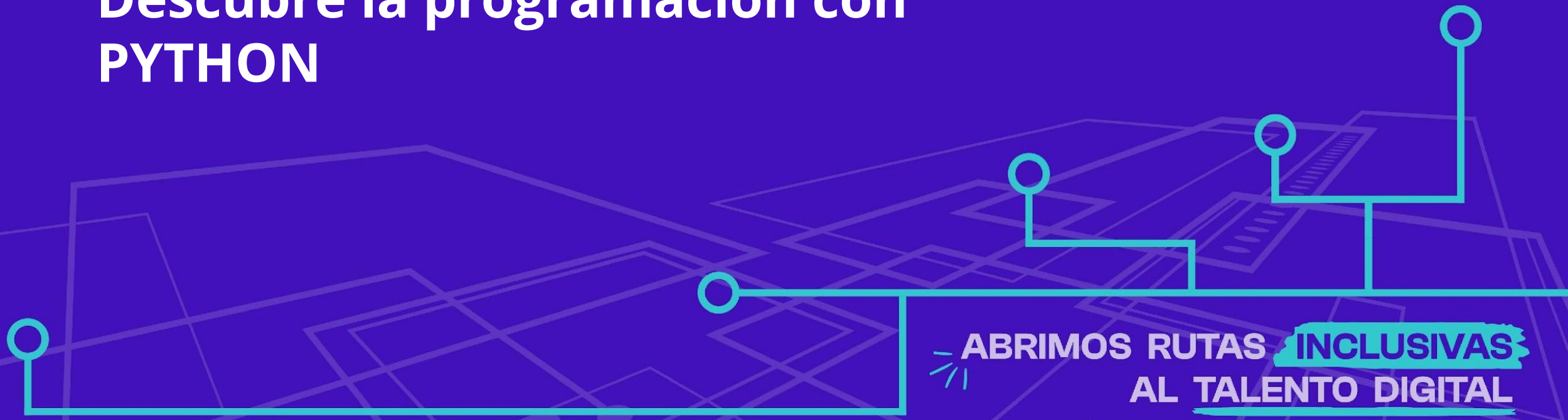
Somos F5

Sage

Descubre la programación con PYTHON

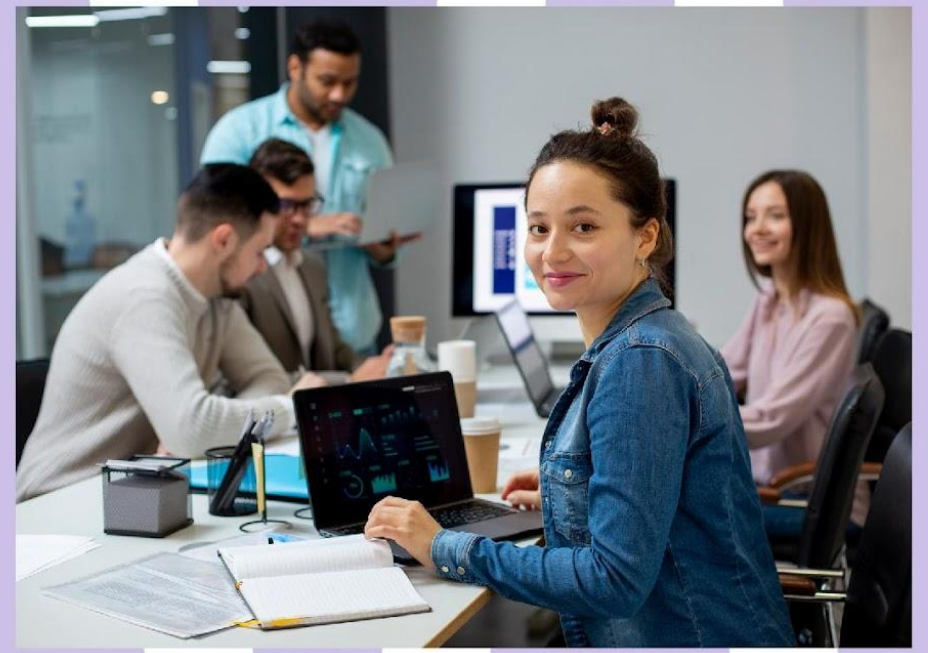


ABRIMOS RUTAS INCLUSIVAS AL TALENTO DIGITAL

Rompehielos

Condicionales

if - Sí
else - entonces



¿Qué es la programación?

Es como una receta de cocina:

Definen los ingredientes (datos), sigues pasos específicos (algoritmos) y el resultado es un platillo listo para servir (output). Si omites un paso o te equivocas, el plato no sale como esperas (bug).

Palabras clave

Sintaxis

Bug

Semántica

Base de datos

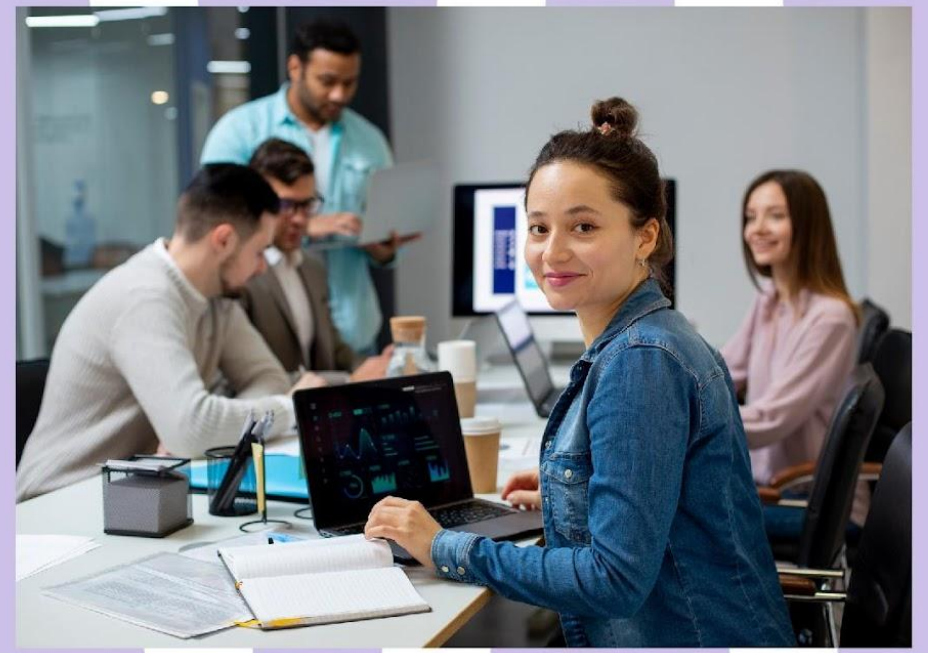
Lenguaje binario

Algoritmo

Repositorio

Variables

Front y Back



Sintaxis y Semántica

🔧 **Piensa en la sintaxis como**:

el conjunto de reglas que te dice *cómo decir lo que quieres decir* en un lenguaje específico.

🧠 **Piensa en la semántica como**:

el *mensaje que estás transmitiendo*, más allá de que uses las palabras correctas.

Sintaxis y Semántica - mostrar ejemplo general

```
# Correcto
for i in range(5):
    print(i)
# Incorrecto (error de sintaxis por falta de dos puntos)
for i in range(5)
    print(i)
```

```
# Código con sintaxis correcta, pero error semántico
lista = [1, 2, 3]
print(lista[5])  # Intentamos acceder a un índice que no existe
```

Ejemplo de Bug en Python

```
def suma_lista(lista):
    total = 0
    for num in lista:
        total += num
    return total
print(suma_lista([1, 2, "3", 4]))  # Error aquí
```

- **Tipo de bug**: **Error de tipo de datos**
- **Error**: `TypeError: unsupported operand type(s) for +=: 'int' and 'str'`
- **Causa**: La lista contiene una cadena (`"3"`) en lugar de un número.
- **Solución**: Convertir todos los elementos a enteros antes de sumarlos.

Frontend o Backend?

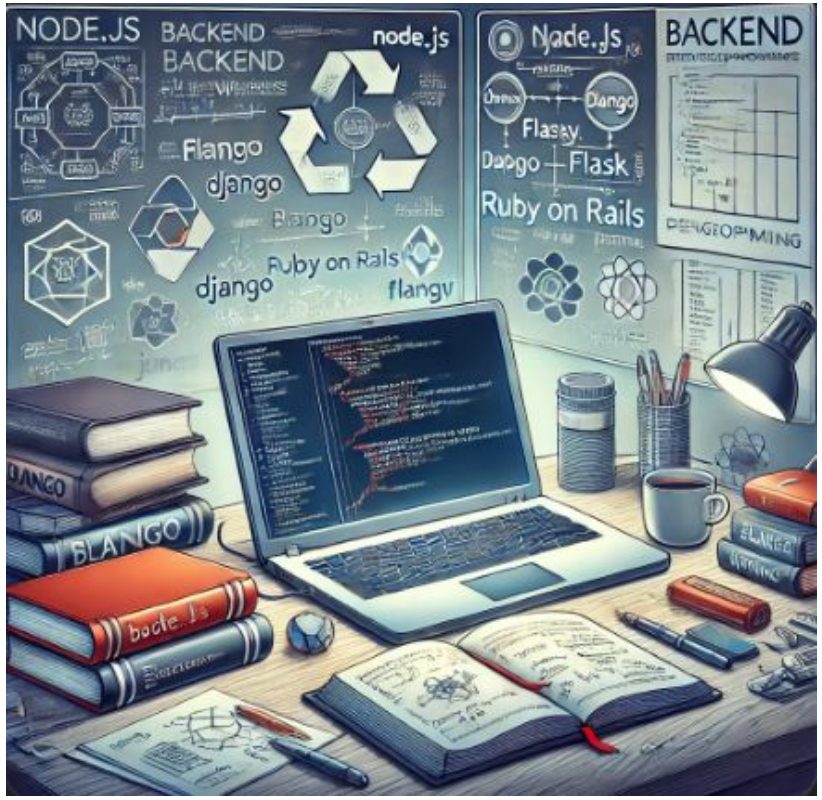
Frontend-Backend-Buckets

En búsqueda del Front y el Back

Investigación acerca de

- Los lenguajes de front y los lenguajes de Back,(10)

- Ejemplos reales de uso(5)



CLICK AQUÍ

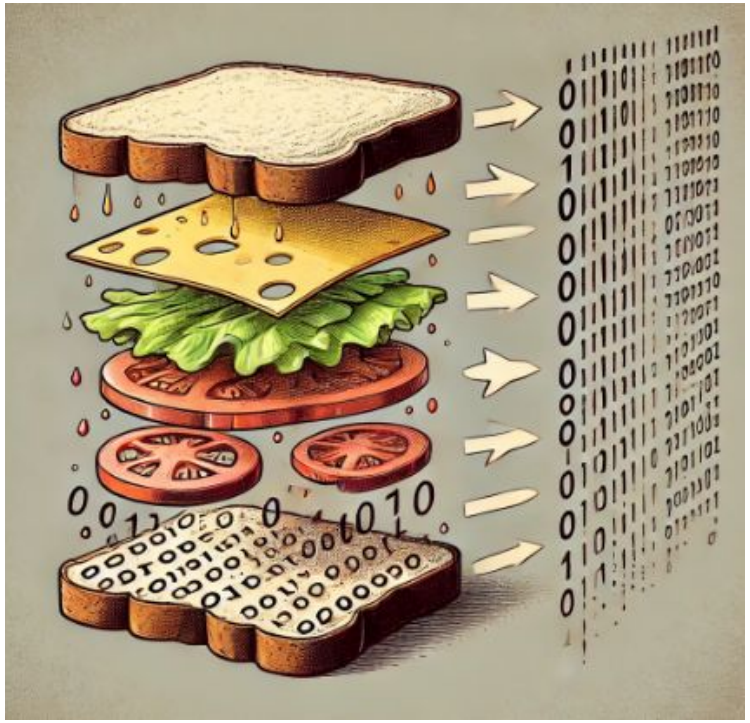
Diferencias entre frameworks y librerías

Característica	Framework	Librería
Definición	Un conjunto estructurado de herramientas y reglas que define una arquitectura para desarrollar aplicaciones.	Un conjunto de funciones reutilizables que ayudan a resolver tareas específicas en el desarrollo.
Control del flujo	El framework dicta la estructura y el flujo del programa.	El programador tiene control total y usa la librería cuando la necesita.
Flexibilidad	Menos flexible, ya que impone una manera específica de trabajar.	Más flexible, ya que se puede usar solo lo necesario.
Ejemplo	Django (para desarrollo web en Python)	NumPy (para cálculos matemáticos en Python)

Buscar frameworks y librerias usadas en python ¿Para qué sirven?
Hacer un documento compartido grupal
-Reto que no se repita ninguno.

Un algoritmo

Un **algoritmo** es una secuencia de pasos definidos y ordenados para resolver un problema o realizar una tarea.



Actividad: Programa un robot!

- Ejercicio: Describe los pasos para hacer **Un robot paseador de perros**

🐕 Planifica la ruta

🐕 Recoge las bolsitas

🐕 Evita charcos

🐕 Regresa a casa a tiempo.

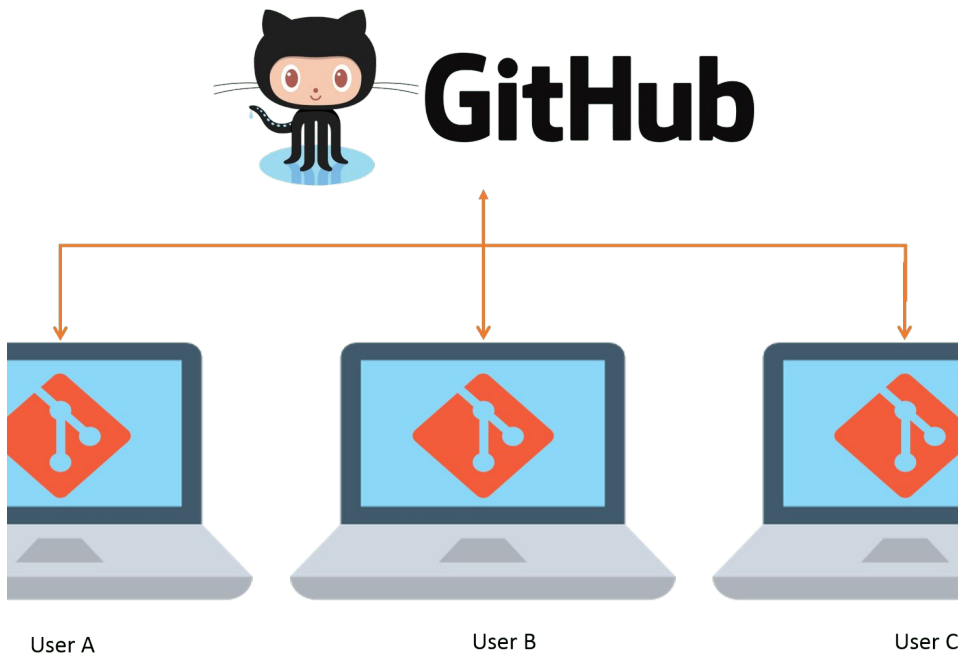
Descomposición de problemas: Haz una receta!

PASOS:

1. Identificar la Tarea Principal
2. Descomponer en Subtareas
3. Descomponer las Subtareas en Pasos Más Específi
4. Convertir los Pasos en Pseudocódigo

Un repositorio

Un **repositorio de código** es un almacenamiento centralizado donde se guarda y gestiona el código fuente de un proyecto. Se suele utilizar un sistema de control de versiones como **Git** y plataformas como **GitHub, GitLab o Bitbucket**.



[Intro Git y Github]

Desafío de Lógica: "El Reto de los Ascensores" 🚀🏢

- **Duración:** Aproximadamente **40 minutos**.
- **Objetivo:** Desarrollar **pensamiento lógico y secuencial**, clave en la programación.
- **Materiales:** (opcional: pizarra digital o Miro para trabajo en equipo).

📜 Enunciado del Desafío

Imagina un edificio con **tres ascensores**: **A, B y C**. Cada ascensor tiene reglas específicas de funcionamiento:

- 🔹 **Ascensor A**: Solo se detiene en pisos **pares**.
- 🔹 **Ascensor B**: Solo se detiene en pisos **impares**.
- 🔹 **Ascensor C**: Se detiene en **todos los pisos**, pero tarda el doble de tiempo en moverse.

El edificio tiene **10 pisos** y hay 3 personas en diferentes pisos que quieren ir a otros pisos.

📌 **Situación:**

- Persona **1** está en el piso **3** y quiere ir al piso **8**.
- Persona **2** está en el piso **6** y quiere ir al piso **1**.
- Persona **3** está en el piso **9** y quiere ir al piso **2**.

Pregunta: 🤔
¿Qué ascensor debe tomar cada persona para llegar lo más rápido posible a su destino? Explica tu razonamiento paso a paso.

Metodología de resolución

1️⃣ **Analizar restricciones** de cada ascensor.
2️⃣ **Determinar opciones** para cada persona.
3️⃣ **Comparar tiempos** y seleccionar la mejor opción.
4️⃣ **Justificar la elección** con lógica secuencial.

💡 **Variante extra:** ¿Qué pasaría si hubiera una emergencia y todos los ascensores deben ser usados de la manera más eficiente?

Este ejercicio fomenta el **pensamiento algorítmico** y la resolución de problemas, esenciales en programación. 🚀💡

Herramientas para escribir código de python

1️⃣ ¿Qué es Jupyter Notebook?

Jupyter Notebook es una **herramienta interactiva** para escribir y ejecutar código en un navegador. Permite mezclar código, texto, imágenes y gráficos en un mismo documento.

- **Año de creación**: 2014
- **Creadores**: Proyecto Jupyter, evolucionado de IPython
- **Uso**:

✅ Ciencia de datos y machine learning
✅ Desarrollo y prueba de código en Python
✅ Creación de documentación interactiva

2️⃣ ¿Qué es Google Colab?

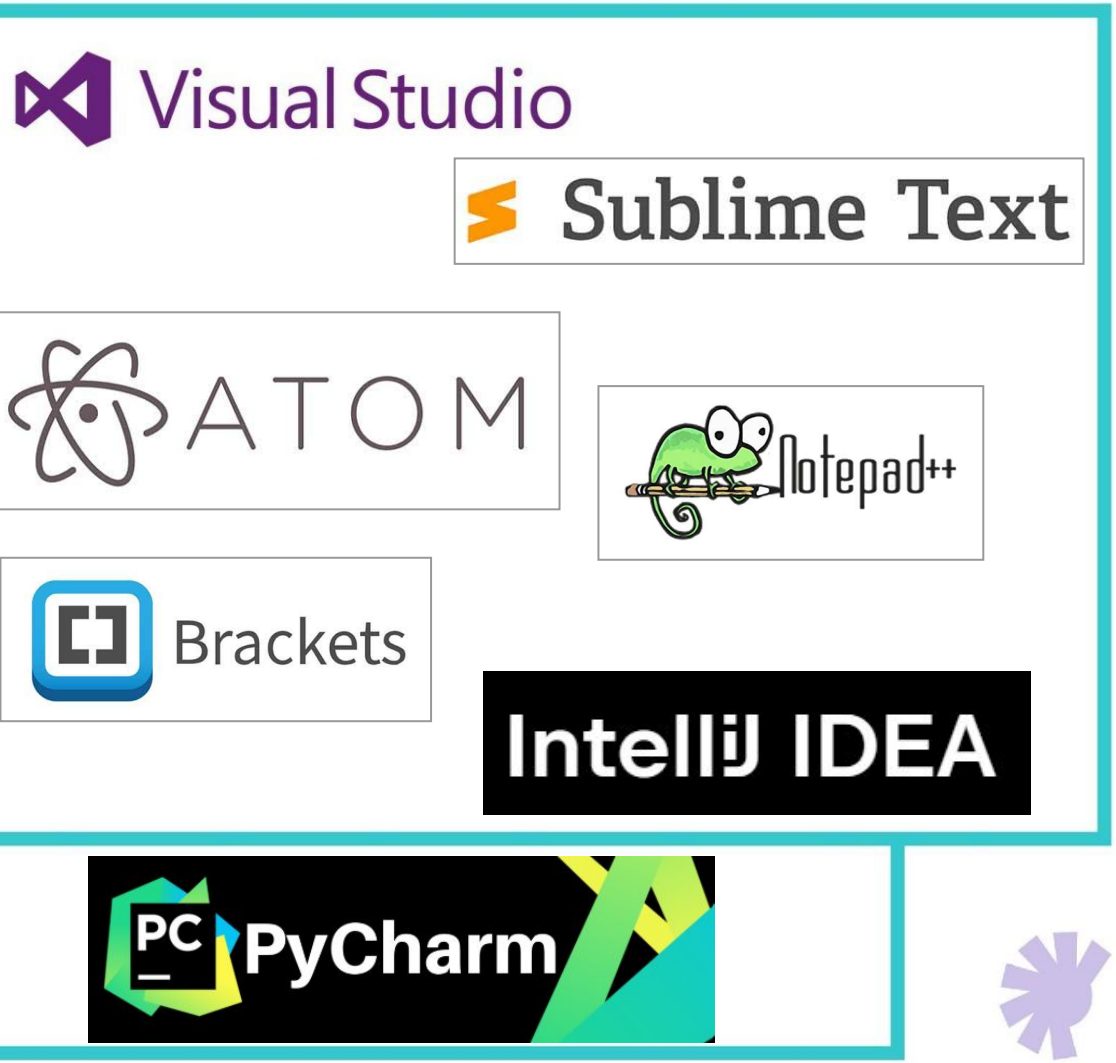
Google Colab (**Colaboratory**) es una versión basada en la nube de Jupyter Notebook, creada por Google.

- **Año de creación**: 2017
- **Ventajas sobre Jupyter Notebook**:

✅ No requiere instalación, se usa en Google Drive
✅ Soporta GPU gratis para tareas de IA
✅ Permite compartir y colaborar en tiempo real

Editores de código

Un **editor de código** es un programa diseñado para escribir y editar código fuente de manera eficiente. Ofrece herramientas como resaltado de sintaxis, autocompletado y gestión de archivos para facilitar la programación.



Intro a Python 🐍

¿Qué es Python y por qué es tan popular?

Python es un lenguaje de programación de alto nivel, fácil de leer y con una sintaxis muy amigable.

Se usa en desarrollo web, inteligencia artificial, ciencia de datos, automatización, ciberseguridad, ¡y mucho más!

Python es un lenguaje de programación de alto nivel que permite:

- Programación imperativa
- Programación funcional
- Programación orientada a objetos

Kahoot: **¿Qué sabemos de...?**

Intro a Python 🐍

📜 Un poco de historia:

- Creado en 1990 por Guido van Rossum.
- Su nombre no viene de la serpiente 🐍, sino del grupo de comedia británico "Monty Python" 🎭.
- Su lema: "Fácil de leer, fácil de escribir, poderoso".
- Actualmente es desarrollado y mantenido por la Python Software Foundation
- Mas info wikipedia
- Zen de Python

Intro a Python 🐍

💡 Aplicaciones reales de Python:

- Netflix: Usa Python para su sistema de recomendaciones.
- Instagram: Backend en Python.
- Tesla: Python en IA para conducción autónoma.
- NASA: Procesamiento de datos espaciales con Python.

🎯 Mini Reto 1: ¿Dónde crees que podrías usar Python en tu día a día?

Primer programa en Python:

Escribir un programa que muestre por pantalla la frase ¡Hola mundo!

```
print("¡Hola Mundo!")
```

print() es una función que muestra texto en pantalla.

Los textos van entre comillas.

🎯 Mini Reto 2:

Modifica el mensaje de salida para que diga:
🚀 "Hola, me llamo [su nombre] y estoy aprendiendo Python"

Sintaxis básica y primeros pasos

VARIABLES y TIPOS DE DATOS

```
nombre = "Ana"          # Tipo string (cadena) - comillas simples (') o comillas dobles (")
edad = 25               # Tipo integer (entero) - Puede ser positivo o negativo
altura = 1.68           # Tipo float (flotante) - Puede ser positivo o negativo
es_estudiante = True    # Tipo boolean (booleano) - True o False (verdadero o falso)
```

Datos primitivos simples

- No hace falta declarar el tipo de variable (Python lo detecta solo).
- Puedes unir una cadena con otra, a ello se le llama **concatenar**
- puedes convertir valores entre diferentes tipos utilizando funciones como `int()`, `float()`, y `str()`
- no todas las conversiones están permitidas - por ejemplo, no puedes convertir el string "hola" a un entero.

🎯 Mini Reto 3:

- crear 3 variables: nombre, edad y si les gusta la pizza(True/False)
- Imprimir mensaje como: "Hola, soy Ana, tengo 25 años y ¿me gusta la pizza? True"

Sintaxis básica y primeros pasos

Datos primitivos compuestos (contenedores)

VARIABLES y TIPOS DE DATOS

lista = [“manzana”, “banana”, “cereza”] # Colecciones de objetos que representan secuencias ordenadas de objetos de distintos tipos.Son mutables(pueden alterarse durante la ejecución de un programa)

tupla = (1, “dos”, 3) # Tienen orden. Pueden contener elementos de distintos tipos. Son inmutables, es decir, no pueden alterarse durante la ejecución de un programa.Se usan habitualmente para representar colecciones de datos una determinada estructura semántica, como por ejemplo un vector o una matriz.

diccionario = {'manzana': 'roja', 'plátano': 'amarillo', 'cereza': 'roja'} # Es una colección no ordenada de pares clave-valor (key:value), donde cada clave debe ser única. Este tipo de datos es útil para almacenar relaciones entre pares de datos.

🎯 Mini Reto 3:

- crear 3 variables: nombre, edad y si les gusta la pizza(True/False)
- Imprimir mensaje como: "Hola, soy Ana, tengo 25 años y ¿me gusta la pizza? True"

Entrada y Salida por Terminal

Entrada por terminal **input()** - es una función incorporada que se utiliza para leer una línea de texto del ingreso del usuario

Salida por terminal **print()** - es una función incorporada que envía salida a la consola". En lenguaje no técnico, significa que se utiliza para mostrar información al usuario.

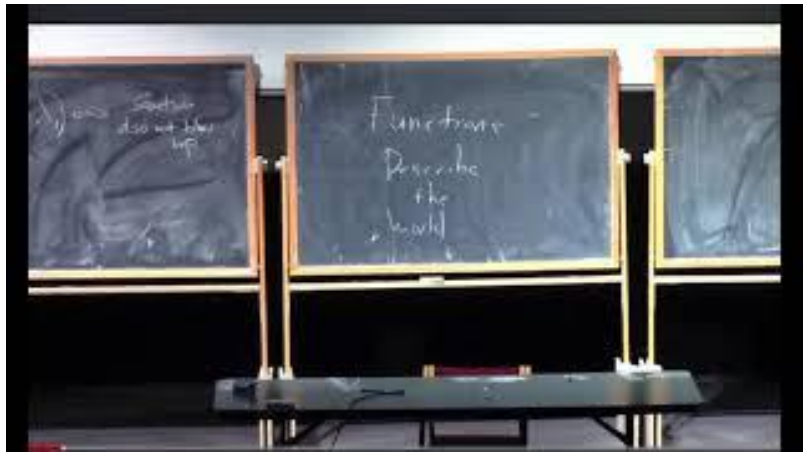
🎯Ejercicio: Crear una variable para almacenar el nombre del usuario y otra para su edad. Imprimir un mensaje de bienvenida.

```
nombre = input("¿Cuál es tu nombre? ")
edad = int(input("¿Cuántos años tienes? "))
print(f"¡Hola, {nombre}! Tienes {edad} años.")
```

f-string, puede incluir varios tipos de datos dentro de una cadena

Funciones

Ejercicio: Crear una función que salude al usuario



definimos una función

parámetro

```
def saludar(nombre):
    return “¡Hola, ” + nombre + “! Bienvenid@ al workshop de Python."

saludar(nombre)
```

devuelve un valor

argumento

llamamos la función

Condicionales if-elif-else

Ejercicio: Crear una función que determine si el usuario es mayor de edad

```
def es_mayor_de_edad(edad):
    if edad >= 18:
        return "Eres mayor de edad."
    else:
        return "Eres menor de edad."

print(es_mayor_de_edad(edad))
```

Otro ejemplo:

```
x = 5
if x > 0:
  print("x es positivo")
elif x < 0:
  print("x es negativo")
else:
  print("x es cero")
```

Bucle for y while

Los bucles en Python se utilizan para ejecutar un bloque de código de forma repetida.

para

```
for i in range(5):
  print(i)

# Output:
# 0
# 1
# 2
# 3
# 4
```

```
i = 0
while i < 5:
  print(i)
  i += 1
```

condición

mientras se cumple la condición

Bucle for y while

Ejercicio con bucle While: Control de Inventario

Eres el gerente de una tienda de zapatos y necesitas controlar tu inventario. Cada vez que vendes un par de zapatos, disminuyes tu inventario. Quieres tener un programa que te permita ingresar la cantidad inicial de pares de zapatos en tu tienda y luego, sucesivamente, te pregunte cuántos pares de zapatos se vendieron. El programa debe continuar preguntando cuántos pares de zapatos se vendieron hasta que el inventario llegue a cero. Finalmente, el programa debe decirte cuántas transacciones (ventas individuales) tuvieron lugar antes de que se acabara el inventario.

Instrucciones

- Inicia con un inventario de X pares de zapatos (donde X es un número que ingresas).
- Pregunta al usuario cuántos pares de zapatos se vendieron.
- Deduce esa cantidad del inventario.
- Repite el proceso hasta que el inventario sea 0.
- Imprime el número total de transacciones realizadas.

Solución ejercicio

```
# Ingreso de la cantidad inicial de zapatos en inventario
inventario = int(input("Ingrese la cantidad inicial de pares de zapatos en la tienda: " ))
transacciones = 0  # Inicializa el contador de transacciones

# Mientras haya zapatos en inventario
while inventario > 0:
    venta = int(input("¿Cuántos pares de zapatos se vendieron en esta transacción? " ))

    # Deduce la venta del inventario y suma al contador de transacciones
    inventario -= venta
    transacciones += 1

    # Verifica que no se ingrese una venta mayor al inventario
    if inventario < 0:
        print("La venta excede el inventario actual. Por favor, ingrese un número
válido.")
        inventario += venta   # Devuelve la cantidad vendida al inventario
        transacciones -= 1    # Resta la transacción inválida

print(f"El inventario ha llegado a cero. Se realizaron  {transacciones} transacciones.")
```

Mini proyecto: Juego de adivinanza de números

Descripción del Proyecto: El programa generará un número aleatorio entre 1 y 100, y el usuario tendrá que adivinarlo. El programa dará pistas si el número es mayor o menor hasta que el usuario acierte.

Mini proyecto: Juego de adivinanza de números

```
import random

def juego_adivinanza():
    numero_secreto = random.randint(1, 100)
    intentos = 0
    adivinado = False

    print("¡Bienvenido al juego de adivinanza de números!")
    print("Estoy pensando en un número entre 1 y 100.")

    while not adivinado:
        experimento = int(input("Adivina el número: "))
        intentos += 1

        if experimento < numero_secreto:
            print("El número es mayor.")
        elif experimento > numero_secreto:
            print("El número es menor.")
        else:
            adivinado = True
            print(f"¡Felicidades! Adivinaste el número en {intentos} intentos.")

juego_adivinanza()
```

Calculadora.py

```
number1 = int(input("Inserta el numero 1 : "))
number2 = int(input("Inserta el numero 2 : "))

def sum(number1, number2):
    result = number1 + number2
    print(number1, "+", number2, "=", result)

def sub(number1, number2):
    result = number1 - number2
    print(number1, "-", number2, "=", result)

def mult(number1, number2):
    result = number1 * number2
    print(number1, "*", number2, "=", result)

def div(number1, number2):
    if (number2 != 0):
        result = number1 / number2
        print(number1, "/", number2, "=", result)
    else:
        print("El Número 2 es igual a 0 y no se puede dividir por 0")


sum(number1,number2)
sub(number1,number2)
mult(number1,number2)
div(number1,number2)
```

Hamburgueseria.py

```
inventario = 100
print(f"El inventario contiene {inventario} hamburgesas")
while inventario > 0:
    if inventario <= 10:
        print("🚨Advertencia: El inventario está casi vacío." )
    num_hamburguesas = int(input("Cuantas hamburguesas quiere el cliente?
"))
    if num_hamburguesas > inventario:
        print("No hay suficientes hamburguesas en stock. Solo hay" ,
inventario, "hamburguesas disponibles." )
    else:
        inventario -= num_hamburguesas
        print(f"El cliente ha pedido {num_hamburguesas} hamburguesas. El
inventario ahora tiene {inventario} hamburguesas.")
```

🚨

RETOS: Te toca a ti!

RETOS: Te toca a ti!

A continuación verás 3 retos que debes superar:

- Refresca la memoria! (**Individual**)
- Opciones opcionales (**Equipo**)
- ¿Qué hora es? (**Equipo**)

RETOS: Refresca la memoria!

Individual, 30 min

ABRIMOS RUTAS INCLUSIVAS
AL TALENTO DIGITAL

RETOS: Refresca la memoria (30min)

Somos F5

Como ya has aprendido.
Cuando creas una función tienes que llamarla para ejecutarla.

Crea un script que te pregunta:

- Nombre
- Año de nacimiento

Ahora con un **cálculo**

- Calcula la edad a partir del año de nacimiento :)

RETOS: Opciones opcionales!

En equipo, 1h30 min

RETOS: Opciones opcionales (1h30m)

Comprobamos si recuerdas las condiciones, bucles y funciones:

Crea un script con un menú:

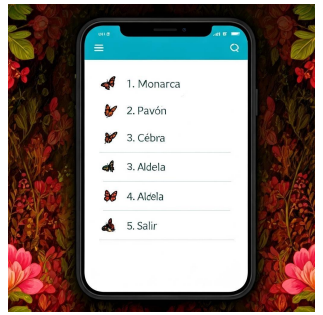
- Del tema que os guste
- Varias opciones
- Varias funciones

RETOS: Opciones opcionales (1h30m)

Por ejemplo, si tu app es de mariposas

Crea opciones distintas:

- Tipo de mariposas existentes
- Descripciones
- Colores



Lo que más te guste pero que
tengas opciones de ejecutar distintas
funciones

RETOS: Opciones opcionales (1h30m)

Ejemplos:

Crea una app de:

- Cálculos: sumar, restar, multiplicar…
- Descripciones: Películas, animales…
- Calculadora de IMC…

Ayúdate de *bucles*, *condiciones* y *funciones*
para conseguir tu app **básica** de **opciones**

RETOS: ¿Qué hora es?

En equipo, 1h30 min

RETOS: Qué hora es? (1h30m)

Ahora vamos a jugar con la vida real!

Dependiendo de la hora, tu app te dirá si es la hora de:

- Desayuno
- Comida
- Merienda
- Cena

Por cada tiempo de comida te ofrece 2 o 3 platos a elegir

RETOS: Qué hora es? (1h30m)

Recuerda que debes utilizar algunas bibliotecas en especial

Tu app debe saber la hora y en consecuencia ofrecerte comida.

- Recuerda las condiciones
- Debes poder elegir
- Trabajas en equipo, comparte las tareas
- Utiliza herramientas agile
- Divertíos mucho!

Kahoot!

Kahoot Retos! - Documentos de Google

Sage

Somos F5

Gracias!

ABRIMOS RUTAS INCLUSIVAS AL TALENTO DIGITAL