Introducción a Python y conceptos básicos de programación

¿Qué es un lenguaje de programación?

Un **lenguaje de programación** es un sistema de instrucciones que permite decirle a una computadora qué hacer. Así como nosotros hablamos en español o inglés, las computadoras entienden lenguajes como Python, C#, Java o JavaScript.
Estos lenguajes permiten crear **programas**, resolver problemas, automatizar tareas y hasta hacer videojuegos.

¿Qué es Python?

Python es uno de los lenguajes de programación más usados del mundo. Es famoso por ser:

- Muy fácil de leer
- Muy fácil de aprender
- Muy utilizado en trabajos reales (web, ciencia de datos, IA, videojuegos, robótica)

Python se destaca porque su sintaxis es clara y corta. Esto hace que sea ideal para quienes están aprendiendo.

Ejemplo rápido en Python:

```
print("Hola, mundo")
```

¿Qué es una variable?

Una **variable** es un espacio donde guardamos un dato.
Podemos imaginarlo como una “caja” con un nombre donde colocamos un valor.

Ejemplos:

```
nombre = "Juan"
edad = 16
altura = 1.72
```

Cada variable guarda un tipo de dato distinto.

Tipos de datos básicos en Python

Python tiene varios tipos de datos importantes:

🔹 str (string) → texto

```
nombre = "Luna"
```

🔹 int (integer) → número entero

```
edad = 15
```

🔹 float → número decimal

```
altura = 1.75
```

🔹 bool (boolean) → verdadero/falso

```
es_estudiante = True
```

🔹 list → lista de datos

```
numeros = [1, 2, 3, 4]
```

¿Cómo usar Python sin instalar nada?

Antes, para usar Python había que descargarlo, pero ahora se puede usar **desde la web**.

Sitios donde podés programar Python online:

- **https://www.onlinegdb.com/online_python_compiler**
- **https://replit.com**
- **https://trinket.io**
- **https://www.programiz.com/python-programming/online-compiler**

Solo necesitás una computadora o celular con navegador.

Si querés instalar Python en tu PC:

Necesitás:

- Descargar Python desde https://www.python.org
- Un editor de código (VS Code, PyCharm o incluso el bloc de notas)

Pero para empezar, la versión online alcanza perfectamente.

Funciones y elementos básicos de Python

A continuación, las herramientas esenciales del lenguaje.

`print()`

Sirve para mostrar cosas en pantalla.

```
print("Hola!")
print(2 + 3)
```

`type()`

Muestra el tipo de dato de una variable.

```
print(type("hola"))   # str
print(type(20))       # int
print(type(3.14))     # float
```

`len()`

Devuelve la longitud (cantidad de caracteres o elementos).

```
len("Hola")          # 4
len([1, 2, 3])       # 3
```

`input()`

Sirve para pedir datos al usuario.

```
nombre = input("Ingresá tu nombre: ")
```

Listas y sus métodos más importantes

Las **listas** son colecciones de elementos, por ejemplo:

```
frutas = ["manzana", "pera", "uva"]
```

A las listas les podemos aplicar métodos para modificarlas.

`append()`

Agrega un elemento al final.

```
frutas.append("banana")
```

`insert()`

Agrega un elemento en una posición específica.

```
frutas.insert(1, "naranja")
```

`remove()`

Elimina la **primera aparición** de un elemento.

```
frutas.remove("uva")
```

`pop()`

Saca un elemento según su posición (o el último si no pongo nada).

```
frutas.pop()      # elimina el último
frutas.pop(1)     # elimina el elemento de la posición 1
```

`reverse()`

Invierte el orden de la lista.

```
frutas.reverse()
```

`sort()`

Ordena la lista (alfabéticamente o numéricamente).

```
numeros = [4, 1, 3, 2]
numeros.sort()
```

Métodos de cadenas de texto (strings)

Los strings tienen muchas funciones para manipular el texto.

`capitalize()`

Primera letra en mayúscula.

```
"juan".capitalize()  # "Juan"
```

`lower()`

Convierte todo a minúsculas.

```
"HOLA".lower()   # "hola"
```

`upper()`

Convierte todo a mayúsculas.

```
"hola".upper()   # "HOLA"
```

`replace()`

Reemplaza partes del texto.

```
"hola mundo".replace("mundo", "Python")
```

`count()`

Cuenta cuántas veces aparece algo.

```
"banana".count("a")   # 3
```

Otros métodos útiles

- `strip()` → elimina espacios al inicio y final
- `split()` → separa texto en “palabras”
- `startswith()` → ¿empieza con…?
- `endswith()` → ¿termina con…?

¿Qué son identificadores?

Los **identificadores** son los nombres que usamos para nuestras variables y funciones.

Reglas importantes:

- No pueden empezar con número
- No pueden tener espacios
- No pueden usar palabras reservadas del lenguaje
- Se recomienda usar minúsculas y guiones bajos:

```
nombre_completo = "Ana Torres"
```

Ejercicios para practicar en Python

EJERCICIO 1

Pedir nombre y apellido y mostrar distintas formas de escribirlos

¿Qué quiere decir el ejercicio?

El programa debe pedir al usuario que escriba su *nombre* y su *apellido*.
Después, debe mostrar esos datos en tres formatos:

1. Todo en MAYÚSCULAS
2. Todo en minúsculas
3. La primera letra en mayúscula y el resto en minúscula (esto se llama *capitalización*)

Paso a paso para entenderlo

1. Pedir datos al usuario

En Python usamos `input()` para pedir que el usuario escriba algo:

```
nombre = input("Ingresá tu nombre: ")
apellido = input("Ingresá tu apellido: ")
```

2 Transformar el texto

Python ya trae funciones para cambiar el formato del texto:

- `upper()` → convierte todo a MAYÚSCULAS
- `lower()` → convierte todo a minúsculas
- `capitalize()` → pone la primera letra en mayúscula

Ejemplo:

```
nombre.upper()
```

3 Mostrar los resultados

Usamos `print()`:

```
print(nombre.upper(), apellido.upper())
print(nombre.lower(), apellido.lower())
print(nombre.capitalize(), apellido.capitalize())
```

Código final del ejercicio 1

```
nombre = input("Ingresá tu nombre: ")
apellido = input("Ingresá tu apellido: ")

print(nombre.upper(), apellido.upper())
print(nombre.lower(), apellido.lower())
print(nombre.capitalize(), apellido.capitalize())
```

EJERCICIO 2

Lista de números impares y verificación de par/impar

¿Qué pide el ejercicio?

1. Crear una **lista** con los números impares del 1 al 10.
2. Pedir al usuario otro número IMPAR entre 1 y 10.
3. Si el número es PAR → mostrar un mensaje diciendo que no cumple.
4. Si el número es IMPAR → decir que está bien.

Paso a paso para entenderlo

1 ¿Qué es una lista en Python?

Es un conjunto de valores guardados entre corchetes `[]`.

Lista de impares:

```
impares = [1, 3, 5, 7, 9]
```

2 Pedir un número al usuario

Recordá que `input()` siempre devuelve texto, así que lo convertimos a número:

```
numero = int(input("Ingresá un número impar del 1 al 10: "))
```

3 Cómo sé si un número es impar?

Usamos el operador `%` → da el resto de una división.

- Si un número % 2 da 0 → es PAR
- Si da 1 → es IMPAR

Ejemplo:

```
if numero % 2 == 0:
    print("El valor ingresado no cumple con el requisito de IMPAR")
else:
    print("El número es impar")
```

4. (Opcional) Verificar además que esté entre 1 y 10

Código final del ejercicio 2

```
# Lista de números impares del 1 al 10
impares = [1, 3, 5, 7, 9]

# Solicitar un número al usuario
numero = int(input("Ingresá un número impar del 1 al 10: "))

# Verificar si es par o impar
if numero % 2 == 0:
    print("El número ingresado es par")
else:
    print("El número ingresado es impar")
```

Alternativa

```
listanumeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print("Ingresá un número impar del 1 al 10:")

numero = int(input())

if numero not in listanumeros:

    print("El número ingresado no está dentro del 1 al 10”)

else:

    if numero % 2 == 0:
        print("El número ingresado es par”)

    else:
        print("El número ingresado es impar y está dentro de la lista")
```

Resumen

- `input()` sirve para pedir datos al usuario.
- `print()` muestra información en pantalla.
- Las listas se escriben como `[1, 2, 3]`.
- Los métodos `.upper()`, `.lower()`, `.capitalize()` cambian el formato del texto.
- El operador `%` permite saber si un número es par o impar.