Introducción: Instrucciones básicas en Python

1. Entrada de datos desde el teclado (`input`)

La función `input()` permite al usuario introducir datos manualmente durante la ejecución del programa. Esta entrada se toma siempre como una cadena de texto (`str`).

Ejemplo:

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

2. Conversión y tipos de datos (`int`, `float`, `str`.)

Como `input()` devuelve texto, es necesario convertirlo si se quiere trabajar con números. Python ofrece funciones para transformar cadenas en tipos numéricos:

- `int()` convierte a número entero.

- `float()` convierte a número decimal.

- `str()` convierte a cadena de texto.

Ejemplos:

```
edad = int(input("Introduce tu edad: "))
```

```
altura = float(input("Introduce tu altura: "))
```

En Python `float` representa números decimales de doble precisión.

3. Salida de datos (`print`) y f-strings

La función `print()` muestra información por pantalla. Las f-strings permiten incluir variables dentro del texto fácilmente.

Ejemplo:

```
print(f 'Tu edad es {edad} años.')
```

4. Estructuras condicionales (`if`, `elif`, `else`)

Permiten ejecutar diferentes bloques de código según condiciones lógicas.

Ejemplo:

```
if a < b:
    print("a es menor que b")
elif a == b:
    print("a y b son iguales")
else:
    print("a es mayor que b")
```

5. Bucles (`for`, `while`)

Sirven para repetir instrucciones:

- `for` recorre elementos de una secuencia (lista, cadena...).

- `while` repite instrucciones mientras se cumpla una condición lógica.

Ejemplos:

```
lista=[1,2,3,4,5,6]
for i in lista:
    print(i)
```

```
while contador < 5:
    print(contador)
    contador += 1
```

6. Listas

Una lista es una estructura que almacena varios valores en orden. Se define con corchetes `[ ]` y puede contener cualquier tipo de dato (números, cadenas, booleanos, otras listas, etc.).

Ejemplo de lista simple:

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

Listas vacías:

```
vacia = []
```

Listas predefinidas:

```
ceros = [0] * 5          # [0, 0, 0, 0, 0]
vacias = [""] * 4        # ["", "", "", ""]
valor_fijo = [3.14] * 3  # [3.14, 3.14, 3.14]
```

Estas listas se pueden modificar posteriormente accediendo a sus elementos por índice:

```
ceros[2] = 10            # cambia el tercer elemento por 10 → [0, 0, 10, 0, 0]
```

7. Definición de funciones (`def`)

Las funciones se definen con `def`, seguidas del nombre y los parámetros. El cuerpo de la función se escribe indentado tras dos puntos `:`.

Ejemplo:

```
def saludar(nombre):
    print(f'Hola, {nombre}')
```

nombre: variable de entrada de la función, tengo que pasársela desde el programa principal.

saludar: nombre de la función

Otro Ejemplo:

```
def minimo(a,b):
        if  a<b:
                num_minimo=a
        elif a>b:
                num_minimo=b
        return num_minimo
```

a y b: variables de entrada de la función, tengo que pasárselas desde el programa principal.

minimo: nombre de la función

return: aquello que la función devuelve al programa principal cuando esta sea llamada. En este caso devuelve el valor guardado en la variable num_minimo

En el programa principal llamaría a la función minimo así:

```
minimo(a, b)
```

8. Uso de la biblioteca `time`

`import time` importa la librería `time`, que permite trabajar con el tiempo. Se usa frecuentemente la función `sleep()` para pausar el programa.

Ejemplo:

```
import time

time.sleep(2)  # Detiene el programa durante 2 segundos
```