Laboratorio 1B - Programas Secuenciales en Python

Objetivos

1. Comprender y aplicar la secuencialidad en los programas.
2. Escribir programas que procesen entradas y generen salidas de manera interactiva.
3. Resolver problemas prácticos mediante programación en Python.

Parte 1: Secuencialidad

Introducción

La secuencialidad significa que las instrucciones de un programa se ejecutan una después de otra, en el orden en que están escritas. Es el flujo más básico de control en programación.

Esto quiere decir que si tenemos, por ejemplo, el siguiente programa:

```
numero1 = 10 # Paso 1: asignar el número 10 a la variable numero1
numero2 = 5 # Paso 2: asignar el número 5 a la variable numero2
suma = numero1 + numero2  # Paso 3: Sumar
producto = suma * 2       # Paso 4: Multiplicar
print("El resultado es:", producto)  # Paso 5: Mostrar el resultado
```

Cada línea se ejecuta de manera ordenada desde la primera hasta la última, y el resultado de cada línea se utiliza en las siguientes.

Actividad 1: Analizando el error

1. Analiza el siguiente código y responde: ¿si lo ejecutamos funcionará? ¿por qué?

   ```
   y = x + 5
   x = 10
   print(y)
   ```

   Luego ejecuta el código y analiza si es que funcionó como pensabas.

2. Arregla el código de arriba para que funcione sin errores.

   ▼ 👀 **Pista**

   El orden de las instrucciones importa. Si `x` no se asigna antes de usarlo en `y`, el programa genera un error.

   Respuesta correcta: hay que invertir el orden de las líneas 1 y 2

   ```
   x = 10
   y = x + 5
   print(y)
   ```

Actividad 2: Cambiando el Resultado

1. Analiza los siguientes códigos: ¿mostrarán el mismo resultado? ¿por qué?

```
a = 2
b = a + 3
a = 5
print(b)
```

```
a = 2
a = 5
b = a + 3
print(b)
```

Parte 2: Entrada y Salida de Datos

Introducción

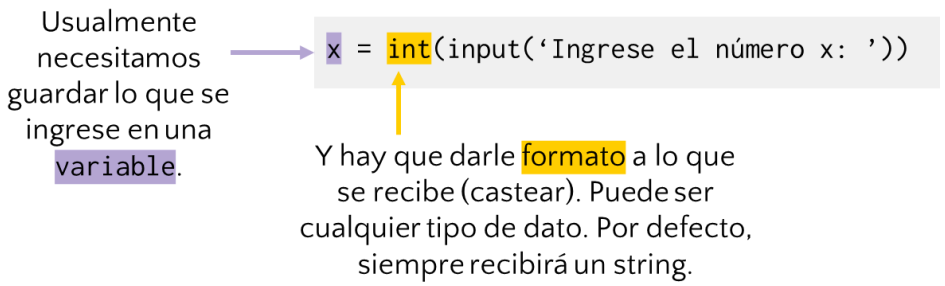
Ya hemos estado utilizando la función `print()` para mostrar cosas por consola, como los resultados de las operaciones que calculamos en los diferentes códigos. A esto lo llamamos **salida de datos**, ya que estamos mostrando cosas al usuario que ejecuta el programa.

Además, hemos visto que **se pueden imprimir varias cosas en una misma línea poniendo comas entre los valores:**

```
print("El resultado es:", resultado, ":D")
```

Sin embargo, también habrán ocasiones en as que necesitaremos que el usuario que ejecuta el programa ingrese algunos valores para poder procesarlos. En este caso hablamos de entrada de datos.

Para leer datos desde la consola se utiliza la función `input()` de la siguiente manera:



⚠️ Si usas la función `input()` sin hacer **casteo**, es decir, sin darle formato a lo que se recibe, el valor ingresado por el usuario siempre será un **string (`str`)**, aunque parezca un número.
Por ejemplo, si en el código queremos recibir un número como entrada de tipo int, pero escribimos esto:

```
numero = input("Ingresa un número: ")
```

Aunque el usuario ingrese un `2`, en el código se guardará como el string `"2"`.

Por eso es muy importante que, si en tu programa necesitas operar con números, **conviertas el input a `int` o `float`**:

```
numero = int(input("Ingresa un número: "))
```

Actividad 1: Entendiendo cómo funcionan las entradas

1. Analiza el siguiente código y responde:

```
numero = int(input("Ingresa un número: "))
cuadrado = numero ** 2
print("El cuadrado de ", numero, " es ", cuadrado)
```

   a. ¿Qué ocurre si ingresas un número decimal? Ejecuta el código y verifica.

   b. Modifica el programa para aceptar números decimales (`float`) en lugar de enteros. ¿Qué ocurre ahora si ingresas un número decimal? ¿y si ingresas un entero?

2. Analiza el siguiente código y responde:

```
base = input("Ingresa la base del triángulo: ")
altura = input("Ingresa la altura: ")
area = base * altura / 2
print("El área es:", area)
```

a. Si se ejecuta el código, ¿qué ocurrirá?

b. Copia y pega el código, y ejecútalo para ver si se cumple tu predicción o no.

c. Arregla el código para que funcione correctamente.

Actividad 2: Usando entradas y salidas

1. Escribe un programa que reciba dos números **enteros** como entrada, los sume y muestre el resultado. **Guíate con el ejemplo a continuación**, que muestra cómo se debería ver el resultado de la ejecución del programa en la consola:

```
Ingresa el primer número: 2
Ingresa el segundo número: 5
La suma resultante de ambos números es: 7
```

> 💡 Cuando el enunciado de un ejercicio tiene ejemplos de ejecución, significa que ese es el **resultado esperado de tu programa**. La idea es que **tu programa contenga los mismos mensajes** de entrada y salida de los datos, **con el mismo formato** que aparece en el ejemplo.

2. Escribe un programa que pregunte al usuario su nombre y edad, y luego muestre un mensaje saludando a la persona y mostrando su edad el próximo año. Guíate con el siguiente ejemplo de ejecución:

```
Ingresa tu nombre: Juan
Ingresa tu edad actual: 20
¡Hola Juan!
El próximo año tendrás 21 años
```

⚠️ Si quieres hacer un salto de línea dentro de una sola instrucción `print()`, puedes usar el carácter especial `\n`, que indica un salto de línea. Por ejemplo, si imprimes:

```
print("Hola\nMundo")
```

En la consola la salida se verá así:

```
Hola
Mundo
```

El `\n` es un carácter de escape que se usa para representar el salto de línea dentro de las cadenas de texto. Si lo pones entre palabras, se imprimen en líneas separadas. Puedes usarlo cuantas veces necesites en un `print()`.

Actividad 3: Parte entera y decimal de un número

Vamos a aplicar lo aprendido hasta ahora realizando un pequeño programa.

Escribe un programa que entregue la parte entera y la parte decimal de un número real ingresado por el usuario.

Para probar tu programa, guíate por estos ejemplos que representan el resultado de lo que debería mostrar tu programa por consola cuando se ejecute:

```
Ingrese un numero real: 4.5
La parte entera es: 4
La parte decimal es: 0.5
```

```
Ingrese un numero real: -1.56
La parte entera es: 1
La parte decimal es: 0.56
```

▼ 👀 **Pista**

Si es que obtienes la parte entera del número podrás obtener la parte decimal haciendo un pequeño cálculo ➖.

Actividad 4: Pseudo-Fibonacci

En 1843 el matemático francés Jacques Philippe Marie Binet publicó su trabajo acerca de la Fórmula de Binet, que permite calcular el **enésimo elemento de la sucesión de Fibonacci**, usando la siguiente fórmula:

$$F_n = \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n \sqrt{5}}$$

Escribe un programa en Python que solicite un número *n* y calcule la enésima sucesión de Fibonacci usando la Fórmula de Binet. Debe mostrar el resultado por pantalla.

> 💡 Para probar tu programa, recuerda que la sucesión de Fibonacci es:
>
> 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,...
>
> Por lo que, por ejemplo, el cuarto número de la sucesión de Fibonacci es 2. Por lo que si en tu programa se ingresa el número 4, el resultado que se muestre por consola debería ser 2.

▼ 👀 **Pista**

Hay ocasiones en las que deberemos programar fórmulas que se ven un poco complejas y puede que no entendamos del todo las matemáticas que hay detrás.

Sin embargo, lo que debemos hacer es **analizar cómo pasar la fórmula que está escrita en términos matemáticos a las operaciones que conocemos en Python**.

Si es que la ecuación es muy compleja, algo que puede ayudar a programarla es dividirla en cálculos más pequeños. Por ejemplo, calcular el numerador (parte de arriba de la división) en una variable, el denominador (parte de abajo de la división) en otra, y luego dividir.

Recuerda el **uso de paréntesis**, ya que ayudan a asegurarte de que se hagan los cálculos en el orden adecuado.

Parte 3: Desafío

Introducción

Vamos a resolver un último ejercicio de código con un contexto un poco más largo que los anteriores, que se asemeja un poco más a lo que serán los ejercicios de los certámenes. Recuerda:

- **Leer todo el enunciado antes de comenzar** a programar, para entender bien el contexto del problema.
- Analizar qué datos tienes que recibir como entrada, qué cálculos debes realizar con ellos, y qué se espera como salida (ten en cuenta los tipos de datos y los formatos de los mensajes que haya en los ejemplos).

Actividad 1: La dieta de Grogu 🍲🍪

Durante su vida, la criatura denominada "Grogu" ha comido diversos alimentos, algunos más polémicos que otros, lo cual ha impulsado a su cuidador a analizar el valor nutricional de lo que consume.

Se te ha solicitado escribir un programa que **reciba las cantidades de alimentos consumidos por Grogu y determine los gramos de grasas, carbohidratos y proteínas consumidos, así como las calorías totales consumidas.** Para esto, guíate de la siguiente información.

Los alimentos que consume Grogu son 3:

- Nevarro Nummies (valores nutricionales por unidad de 11 [g]):
  - 1.90 [g] Grasas
  - 6.00 [g] Carbohidratos
  - 0.80 [g] Proteínas
- Sapce Soup (valores nutricionales por porción de 285 [ml]):
  - 10.0 [g] Grasas
  - 12.0 [g] Carbohidratos
  - 11.0 [g] Proteínas
- Carne de Rana (valores nutricionales por cada 100 [g]):
  - 0.30 [g] Grasas
  - 0.00 [g] Carbohidratos

- 16.0 [g] Proteínas

Desarrolla un programa que solicite el **ingreso de la cantidad de cada uno de los alimentos consumida por Grogu**. Ten en cuenta que Las “Nevarro Nummies” se pueden consumir enteras o por fracción, la sopa se reporta en [ml] y la carne de rana se reporta en gramos [g].

Una vez ingresados los datos, el programa debe calcular y mostrar por pantalla las **cantidades totales de grasas, carbohidratos y proteínas consumidas por Grogu en [gr] redondeadas a 2 decimales.**

Un ejemplo de ejecución del programa sería:

```
Nevarro Nummies consumidas (en unidades): 1.5
Space Soup consumida (en [ml]): 420
Carne de rana consumida (en [g]): 210
Grogu ha consumido:
18.22 [g] de grasas
26.68 [g] de carbohidratos
51.01 [g] de proteinas
```

💡 Recuerda siempre revisar el ejemplo de ejecución del programa, para ver el formato en que se deben imprimir los resultados.

▼ 👀 **Pista**

Guíate por el siguiente paso a paso para resolver el desafío

1. **Entrada de datos**:
   - Pide al usuario que ingrese las cantidades de los tres alimentos consumidos por Grogu: *Nevarro Nummies* (en unidades), *Space Soup* (en ml), y *Carne de Rana* (en gramos).
   - Recuerda convertir las entradas a los tipos de datos adecuados.
2. **Cálculo de grasas, carbohidratos y proteínas**:
   - Define las constantes de nutrientes para cada alimento.
   - Calcula los gramos de grasas, carbohidratos y proteínas consumidos para cada alimento, teniendo en cuenta las unidades

ingresadas.
    - Luego suma el total de grasas, carbohidratos y proteínas entre los 3 alimentos.
3. **Redondeo de resultados**:
    - Redondea los gramos de cada nutriente a dos decimales.
    - Redondea las calorías totales al entero más cercano.
4. **Salida del programa**:
    - Muestra los resultados de manera clara y ordenada al usuario.

Actividad 2: ¿Cuántas calorías consumió Grogu? ⚖️

Para poder ver si la dieta de Grogu es adecuada, su cuidador necesita conocer la cantidad de calorías totales que está consumiento. Para eso cosigue la tablas de calorías alimentarias, con las siguientes relaciones entre nutrientes y la cantidad de calorías:

- 1 [g] Grasas = 9 [calorías]
- 1[g] Carbohidratos = 4 [calorías]
- 1 [g] Proteínas = 4 [calorías]

Modifica tu programa anterior para que al final imprima la **cantidad total de calorías consumidas por Grogu redondeadas al entero más cercano**.

Un ejemplo de ejecución del programa final sería:

```
Nevarro Nummies consumidas (en unidades): 1.5
Space Soup consumida (en [ml]): 420
Carne de rana consumida (en [g]): 210
Grogu ha consumido:
18.22 [g] de grasas
26.68 [g] de carbohidratos
51.01 [g] de proteinas
Dando un total de 475 [calorias]
```

▼ 👀 **Pista**

Recuerda que la función de redondeo `round(x)` redondea el número `x` al entero más cercano.

Actividad 3: El entrenamiento de Grogu ✨

Grogu no solo se alimenta bien, también está entrenando para convertirse en un **gran Jedi**. Su maestro Luke, le ha dicho que necesita mantener un buen equilibrio entre su dieta y su entrenamiento físico para poder desarrollar sus habilidades en la Fuerza.

Grogu realiza dos tipos de entrenamientos para mantener su físico y estar listo para cualquier desafío:

- **Entrenamiento de Fuerza**: Este tipo de ejercicio lo prepara para mantener la concentración y desarrollar la resistencia física.
- **Entrenamiento de Resistencia**: Fundamental para mejorar la agilidad y la rapidez con que mueve su sable de luz.

El Maestro Luke necesita calcular cuántos **minutos de entrenamiento** debe realizar Grogu para quemar las calorías consumidas en su dieta, según el tipo de entrenamieto que haga. Para eso, Luke tiene la siguiente información:

- **Entrenamiento de Fuerza**: Quema 8 calorías por minuto.
- **Entrenamiento de Resistencia**: Quema 12 calorías por minuto.

Modifica tu programa anterior, para que también calcule y muestre cuántos minutos de cada tipo de entrenamiento se deben realizar Grogu para quemar las calorías consumidas.

Un ejemplo de ejecución del programa final sería:

```
Nevarro Nummies consumidas (en unidades): 1.5
Space Soup consumida (en [ml]): 420
Carne de rana consumida (en [g]): 210
Grogu ha consumido:
18.22 [g] de grasas
26.68 [g] de carbohidratos
51.01 [g] de proteinas
Dando un total de 475 [calorias]
```

Si Grogu hace entrenamiento de fuerza, debe entrenar 59 minutos
Si Grogu hace entrenamiento de resistencia, debe entrenar 40 minutos