6. Estructuras de control

En cualquier programa, es esencial poder controlar el flujo de la ejecución. Aquí es donde entran en juego las estructuras de control, que nos permiten tomar decisiones y repetir acciones en nuestros programas. Antes de adentrarnos en los detalles de estas estructuras, recordemos algunos conceptos fundamentales de tipos de datos y operadores que hemos visto previamente. Los tipos de datos, como enteros, cadenas y booleanos, junto con los operadores de comparación y lógicos, son fundamentales para evaluar las condiciones que rigen nuestras decisiones. Además, la entrada y salida de datos nos permiten interactuar con el usuario, obteniendo información necesaria y mostrando resultados relevantes. Estas habilidades serán cruciales al trabajar con declaraciones condicionales, como if, elif y else, que nos permiten desarrollar programas dinámicos que responden a diferentes situaciones. Vamos a explorar estas estructuras de control en más detalle.

a. Cruce de decisiones: Condicionales

Las estructuras de control son como los signos de tráfico en el camino de la ejecución de nuestro programa. Nos permiten tomar decisiones basadas en diferentes condiciones y repetir ciertas acciones hasta que se cumpla una condición. En Python, los condicionales se manejan con las palabras clave if, elif y else.

Vamos a imaginar que estamos desarrollando un programa para determinar si un estudiante ha aprobado o reprobado un examen basado en su calificación.

```
calificacion = int(input("Por favor ingrese la calificación del
estudiante: "))

if calificacion >= 6:
    print("El estudiante ha aprobado el examen.")
else:
    print("El estudiante ha reprobado el examen.")
```

En este ejemplo, el programa pide al usuario que ingrese la calificación del estudiante. Luego, usa una declaración if para comprobar si la calificación es igual o superior a 6. Si es así, el programa imprime un mensaje diciendo que el estudiante ha aprobado. Si no es así, el programa imprime un mensaje diciendo que el estudiante ha reprobado.

A menudo, cuando programamos, nos encontramos con situaciones donde no basta con preguntar "¿es esto cierto o falso?". La realidad suele ser más compleja y necesitamos considerar múltiples condiciones. Por ejemplo, ¿qué pasa si queremos saber si un número es positivo, negativo o cero? No podemos limitarnos a una sola pregunta. Aquí es donde el uso de elif se vuelve esencial, permitiéndonos manejar varios escenarios posibles con claridad y precisión.

```
numero = float(input("Ingrese un número: "))
```

```python
if numero > 0:
    print(f"{numero} es un número positivo.")
elif numero == 0:
    print(f"{numero} es cero.")
else:
    print(f"{numero} es un número negativo.")
```

A veces, nos encontramos en situaciones donde queremos asignar un valor a una variable dependiendo de una condición específica. Por ejemplo, queremos saber si un número es par o impar y asignar un valor basado en esa condición. En Python, podemos hacer esto de manera concisa utilizando expresiones condicionales en una sola línea. Esta técnica nos permite escribir código más limpio y directo. Aquí está la solución:

```python

```

b. Ciclos terminables: Bucles 'for'

En ocasiones, necesitaremos repetir una acción varias veces. Aquí es donde los bucles entran en juego. Python ofrece dos tipos de bucles: for y while.

Para ilustrar cómo funcionan, vamos a usar un ejemplo simple. Supongamos que queremos imprimir los números del 1 al 5.

```python
for numero in range(1, 6):
    print(numero)
```

En este caso, el bucle for recorre los números del 1 al 5 (el límite superior en range() es exclusivo), imprimiendo cada número en la consola.

Las estructuras de control, los condicionales y los bucles son fundamentales en la programación. Nos permiten crear programas más dinámicos y flexibles, capaces de tomar decisiones y repetir tareas basadas en las condiciones que definamos.

c. Please, stop the 'while'

Primero vamos a ver que es "while", es un bucle que repite un bloque de codigo mientras sea verdadera, en nuestras palabras imaginemos que tenemos 10 ml de agua este estara con agua que es nuestra verdad hasta que lo terminamos ahi paso a ser falso entonces se terminaria el bucle

Vamos a aplicar nuestro conocimiento recién adquirido sobre bucles para resolver un problema real: calcular el promedio de las calificaciones de un estudiante. Supongamos que un estudiante nos proporciona sus calificaciones una por una. El estudiante indicará que ha terminado de proporcionar las calificaciones ingresando el número 0.

```python

```

En este código, el bucle while se ejecuta hasta que el estudiante ingrese 0. Si bien es sencillo y cumple con su objetivo, este código no verifica si las calificaciones ingresadas están dentro del rango válido (1 a 12).

d. Controlando el flujo con 'break' y 'continue'

Ahora, vamos a mejorar nuestro código anterior incorporando las declaraciones break y continue para un mejor control del flujo del programa y la validación de las calificaciones.

En este código, la declaración break se utiliza para detener el bucle cuando el estudiante ingresa 0. Por otro lado, la declaración continue se usa para saltar a la próxima iteración del bucle si la calificación ingresada no está dentro del rango válido, evitando así que calificaciones incorrectas se sumen a suma_calificaciones y se cuenten en contador_asignaturas.

```

```

Desafíos

Desafío 1: Calificaciones aprobadas

Supón que estás analizando las calificaciones de los estudiantes y quieres saber cuántos aprobaron y cuántos desaprobaron. Se considera que una calificación de 7 o superior es aprobatoria y cualquier calificación menor a 7 es desaprobatoria. Utiliza lo que aprendiste sobre bucles y condicionales para resolver este problema.

Desafío 2: Mejora del cálculo

Toma el ejemplo del cálculo del promedio de calificaciones y mejóralo. En lugar de pedir las calificaciones una por una, modifica el código para pedir todas las calificaciones al mismo tiempo (el estudiante puede ingresar las calificaciones separadas por comas) y luego calcular el promedio.

```
# Pedir la calificación del estudiante
for i in range(1,10):
  calificaciones = [5,3,9,2,10,7]
  while True:
    if 0 <= calificacion <= 10:
        calificaciones.append(calificacion)
        break
# Calcular la suma de las calificaciones
suma_calificaciones = sum(calificaciones)
# Calcular el número de asignaturas
contador_asignaturas = len(calificaciones)
# Calcular el promedio
if contador_asignaturas > 0:
```

```
promedio = suma_calificaciones / contador_asignaturas
print(f"El promedio de las calificaciones es: {promedio}")
```

La situación pide tomar como ejemplo el cálculo de promedio y solicitar las calificaciones del estudiante y expresarlas todas juntas. En primer lugar solicité ingresar las calificaciones, separadas por comas. Las ubiqué dentro del rango del 1 al 10. Luego se utilizó el bloque "While" que repite el código y "break" para finalizar. Posteriormente sumar las calificaciones y expresar en la máquina la cantidad de asignaturas. Me dificultó poder unir los bucles, incorproando además condicionales como "if". El material leído sobre Estructura de Control fue de gran apoyo. La funcion "len" fue usada como longitud (cantidad de elementos) Para finalizar calculé el promedio, expresando la división de las calificaciones obtenidas entre la cantidad de asignaturas. Allí se imprimió el promedio.

Webgrafía: Entrada y salida de datos. Estructura de control. Tipos de datos y operadores.

```

```

Desafío 3: Simulación de una carrera de autos

Vas a simular una carrera de autos. Cada auto tiene una velocidad aleatoria (puedes usar la biblioteca random de Python) y cada ciclo del bucle representa un segundo de la carrera. Al final de cada segundo, cada auto avanza una distancia igual a su velocidad. La carrera dura 10 segundos. Al final de la carrera, debes imprimir el auto ganador. Si hay un empate, debes imprimir todos los autos que empataron.

Nota: Este desafío puede requerir que aprendas sobre conceptos adicionales, por ejemplo cómo generar números aleatorios.

Problemas

Problema: Cálculo de gastos de un viaje

Supongamos que estás planeando un viaje y quieres calcular el total de gastos estimados. Tienes un arreglo de los diferentes conceptos de gastos (por ejemplo, transporte, alojamiento, alimentación, actividades, etc.) y quieres solicitar al usuario que ingrese el costo estimado para cada concepto. Al final, quieres mostrar el total de gastos estimados para el viaje.

Planteamiento del problema:

- Crea un arreglo de conceptos de gastos.
- Utiliza un bucle for para iterar sobre el arreglo y solicitar al usuario que ingrese el costo estimado para cada concepto.
- Utiliza una variable para realizar el seguimiento del total de gastos estimados, inicializada en 0.
- Dentro del bucle, agrega el costo ingresado a la variable del total de gastos estimados.
- Al final del bucle, muestra el total de gastos estimados.

Aquí tienes un ejemplo de cómo podrías resolver este desafío utilizando arreglos:

```
import numpy as np

conceptos_gastos = np.array(['Transporte', 'Alojamiento',
'Alimentación', 'Actividades', 'Otros'])

total_gastos = 0

for concepto in conceptos_gastos:
    costo = float(input(f"Ingrese el costo estimado para {concepto}:
"))
    total_gastos += costo

print(f"El total de gastos estimados para el viaje es:
{total_gastos}")
```

En este código, utilizamos un bucle for para iterar sobre el arreglo de conceptos de gastos. En cada iteración, solicitamos al usuario que ingrese el costo estimado para ese concepto y lo sumamos al total de gastos estimados. Al final, mostramos el resultado.

¡Ahora puedes calcular fácilmente el total de gastos estimados para tu próximo viaje utilizando arreglos! ¡Es asombroso cómo Python nos permite resolver problemas de manera eficiente y sencilla! No te preocupes, en el próximo cuaderno exploraremos más a fondo los conceptos y funcionalidades de los arreglos en Python.

Problema: Análisis de calificaciones utilizando Pandas y bucles

Supongamos que tienes un archivo CSV llamado "calificaciones.csv" que contiene las calificaciones de varios estudiantes. Tu objetivo es realizar un análisis de las calificaciones utilizando la biblioteca Pandas y bucles for.

Planteamiento del problema: Importa la biblioteca Pandas. Carga el archivo CSV "calificaciones.csv" en un DataFrame de Pandas. Utiliza un bucle for para calcular el promedio de cada estudiante y mostrarlo en la consola.

Aquí tienes un ejemplo de cómo podrías resolver este desafío utilizando Pandas y bucles for:

```
import pandas as pd

# Carga el archivo CSV en un DataFrame
df = pd.read_csv("calificaciones.csv")

# Inicializa variables para calcular el promedio general
total_calificaciones = 0
numero_calificaciones = 0

# Utiliza un bucle for para sumar todas las calificaciones
```

```
for index, row in df.iterrows():
    nombre = row['Nombre']
    calificacion = int(row['Calificacion'])
    total_calificaciones += calificacion
    numero_calificaciones += 1
    print(f"La calificación de {nombre} es: {calificacion}")

# Calcula el promedio general
promedio_general = total_calificaciones / numero_calificaciones
print(f"El promedio general de las calificaciones es:
{promedio_general:.2f}")
```

En este código, utilizamos Pandas para cargar el archivo CSV en un DataFrame. Luego, utilizamos un bucle for junto con iterrows() para iterar sobre cada fila del DataFrame. Para cada estudiante, extraemos el nombre y la calificación, para luego calcular el promedio.