GUÍA DE LABORATORIO 04
Estructuras de control de flujo para repetición en Python – While -For

Asignatura	Datos del alumno	Fecha y Firma
Algoritmos y solución de problemas	Apellidos:	
	Nombre:	

Instrucciones:

Desarrollar las actividades que indica el docente en base a la guía de trabajo que se presenta.

1. **Objetivos:**
   - Comprender y aplicar la estructura de control de flujo while for en Python para ejecutar ciclos en programas.
   - Escribir programas en Python que utilicen while y for para ejecutar diferentes acciones basadas en ciclos.

2. **Equipos, herramientas o materiales**
   - Computador
   - Software: Python - VSCode
   - Algoritmos

3. **Fundamento teórico**

Estructuras de Control de Flujo para Repetición en Python

while condición: El bloque de código dentro del while se repetirá mientras la condición sea verdadera (True). Es crucial asegurarse de que la condición eventualmente se vuelva falsa para evitar un bucle infinito.

for elemento in secuencia: El bloque de código dentro del for se ejecutará una vez para cada elemento en la secuencia (como una lista, tupla o cadena).

Python tiene dos comandos de bucle primitivos:

- bucles while
- bucles for

3.1. The While Loop

Con el bucle while podemos ejecutar un conjunto de sentencias siempre que una condición sea verdadera.

Ejemplo: Imprima i siempre que i sea menor que 6:

```
01_While_Loop.py
i = 1
while i < 6:
    print(i)
    i += 1
```

Nota: recuerda incrementar i, de lo contrario el bucle continuará para siempre.

El bucle while requiere que las variables relevantes estén listas, en este ejemplo necesitamos definir una variable de indexación, **i** , que establecemos en **1**.

The break Statement

Con la declaración **break** podemos detener el bucle incluso si la condición **while** es verdadera:

Ejemplo: Salir del bucle cuando i es 3:

02_Declaracion_Break.py

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

The continue Statement

Con la declaración **continue** podemos detener la iteración actual y continuar con la siguiente:

Ejemplo: Continúe con la siguiente iteración si i es 3:

03_Declaracion_continue.py

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

The else Statement

Con la declaración **else** podemos ejecutar un bloque de código una vez cuando la condición ya no sea verdadera:

Ejemplo: Imprima un mensaje una vez que la condición sea falsa:

04_Declaracion_else.py

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

3.2. Python For Loops

Un bucle **for** se utiliza para iterar sobre una secuencia (que puede ser una lista, una tupla, un diccionario, un conjunto o una cadena).

Esto se parece menos a la palabra clave **for** en otros lenguajes de programación y funciona más como un método iterador como el que se encuentra en otros lenguajes de programación orientados a objetos.

Con el bucle **for** podemos ejecutar un conjunto de sentencias, una vez para cada elemento de una lista, tupla, conjunto, etc.

Ejemplo: Imprima cada fruta en una lista de frutas:

05_for_Loop.py

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

El bucle **for** no requiere que se establezca una variable de indexación de antemano.

Looping Through a String

Incluso las cadenas son objetos iterables, contienen una secuencia de caracteres:

Ejemplo: Recorre las letras de la palabra "banana":

06_For_Strings.py

```
for x in "banana":
  print(x)
```

The break Statement

Con la declaración **break** podemos detener el bucle antes de que haya recorrido todos los elementos:

Ejemplo: Salir del bucle cuando **x** es "banana":

07_For_Break.py

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Ejemplo: Salir del bucle cuando **x** es "banana", pero esta vez el salto viene antes de la impresión:

08_For_break2.py

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

The continue Statement

Con la declaración **continue** podemos detener la iteración actual del bucle y continuar con la siguiente:

Ejemplo: No imprimir banana:

09_For_Contiue.py

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

The range() Function

Para recorrer un conjunto de código una cantidad específica de veces, podemos usar la función **range().**

La función **range()** devuelve una secuencia de números, comenzando desde 0 de manera predeterminada, y se incrementa en 1 (de manera predeterminada), y finaliza en un número especificado.:

Ejemplo: Usando la función range():

10_For_Range.py

```
for x in range(6):
  print(x)
```

Tenga en cuenta que **range(6)** no son los valores de 0 a 6, sino los valores de 0 a 5.

La función **range()** tiene como valor inicial predeterminado 0, sin embargo es posible especificar el valor inicial agregando un parámetro: **range(2, 6)** , que significa valores de 2 a 6 (pero sin incluir 6):

Ejemplo: Usando el parámetro de inicio:

11_For_Star_Parametro.py

```
for x in range(2, 6):
  print(x)
```

La función **range()** tiene como valor predeterminado incrementar la secuencia en 1, sin embargo es posible especificar el valor de incremento agregando un tercer parámetro: **range(2, 30, 3)**:

Ejemplo: Incrementa la secuencia con 3 (el valor predeterminado es 1):

12_For_incremento_parametro.py

```
for x in range(2, 30, 3):
  print(x)
```

Else in For Loop

La palabra clave **else** en un bucle **for** especifica un bloque de código que se ejecutará cuando finalice el bucle:

Ejemplo: Imprime todos los números del 0 al 5 e imprime un mensaje cuando el bucle haya finalizado:

13_For_else.py

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Nota: El bloque **else** NO se ejecutará si el bucle se detiene mediante una declaración **break**.

Ejemplo: Rompa el bucle cuando x sea 3 y observe qué sucede con el bloque else:

14_For_Else_Break.py

```python
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")
```

Nested Loops

Un bucle anidado es un bucle dentro de un bucle.

El "bucle interno" se ejecutará una vez por cada iteración del "bucle externo":

Ejemplo: Imprime cada adjetivo para cada fruta:

15_For_nested.py

```python
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

}

The pass Statement

Los bucles **for** no pueden estar vacíos, pero si por alguna razón tienes un bucle for sin contenido, ponlo en la declaración **pass** para evitar obtener un error.

16_For_pass.py

```python
for x in [0, 1, 2]:
  pass
```

4. Desarrollo y Actividades

Ejercicio parte 01:

1) Suma de Números Pares:
   Pide al usuario un número entero positivo 'n'. Calcula la suma de todos los números pares desde 2 hasta 'n' (inclusive) utilizando un bucle while.
2) Conjetura de Collatz:
   Pide al usuario un número entero positivo. Aplica la conjetura de Collatz: si el número es par, divídelo por 2; si es impar, multiplícalo por 3 y suma 1. Repite el proceso hasta que el número llegue a 1. Utiliza un bucle while e imprime la secuencia de números generada.
3) Tabla de Multiplicar con for:
   Pide al usuario un número entero positivo. Imprime la tabla de multiplicar de ese número utilizando un bucle for.
4) Imprimir Patrón de Triángulo:
   Pide al usuario un número entero positivo 'n'. Imprime un triángulo de asteriscos con 'n' filas utilizando un bucle for anidado.
5) Contar Vocales en una Frase:

Pide al usuario una frase y cuenta el número total de vocales (mayúsculas y minúsculas) utilizando un bucle for.

6) Suma de Dígitos:
   Pide al usuario un número entero positivo. Calcula la suma de sus dígitos utilizando un bucle while.
7) Invertir una Cadena:
   Pide al usuario una cadena e inviértela utilizando un bucle for.
8) Número de Fibonacci:
   Pide al usuario un número entero positivo 'n'. Calcula el enésimo número de Fibonacci utilizando un bucle for.
9) Verificar si una Cadena es Palíndromo:
   Pide al usuario una cadena y verifica si es un palíndromo (se lee igual de izquierda a derecha que de derecha a izquierda) utilizando un bucle for.
10) Calculadora de Promedio:
    Pide al usuario una serie de números separados por comas. Calcula el promedio de esos números utilizando un bucle for.