



**Nombre:** Mikaela Zurita

**Paralelo:** "B"

**Do – While en Python**

#### **1. ¿Qué es un Do - While?**

El Do – While es una estructura de control de flujo en programación que ejecuta un bloque de código al menos una vez y luego continúa ejecutándolo mientras una condición específica sea verdadera.

En Python, no existe una estructura do-while como en otros lenguajes (C, C++, Java, JavaScript), pero se puede simular su comportamiento usando un **while True** que va a ejecutar por lo menos una vez la condición establecida.

#### **2. Estructura while True (Python).**

```
1  while True:  
2      dato = input("Ingresa un numero mayor a 5: ")  
3      if int(dato) > 5:  
4          break
```

Explicación:

1. Se usa **while True** para garantizar que el código dentro del bucle siempre se ejecute al menos una vez.
2. Se obtiene la entrada del usuario.
3. Se evalúa la condición después de ejecutar el código.
4. Si la condición se cumple, **break** detiene el bucle.

#### **3. Importancia de un while True en la programación de Python.**

El bucle **while True** es una herramienta importante de la programación en Python, ya que permite controlar grupos de código de manera indefinida hasta que se cumpla la condición establecida. Su importancia ya que se tiene mayor flexibilidad y control al momento de gestionar diversas situaciones, desde la validación de datos hasta la implementación de servidores en tiempo real.

#### **4. Utilidad de un while True en la programación de Python.**

##### **- Validación de entrada de usuario**

Asegura que el usuario ingrese los datos correctos antes de continuar con el programa. Es útil en los siguientes casos: formularios, capturas de datos y programas que necesiten entradas seguras.

D: > PROGRAMACION > VALIDACION ENTRADA DE USUARIO (DO WHILE).py > ...

```
1  while True:
2      edad = input("Ingresa tu edad: ")
3      if edad.isdigit() and int(edad) > 0:
4          print(f"Edad valida: {edad}")
5          if edad >= "18":
6              print("Eres Mayor de edad.")
7              break
8          else:
9              print("Eres menor de edad")
10         # Se sale del bucle si la entrada es correcta
11     print("Entrada no valida. Intentalo de nuevo.")
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Det-Pc/AppData/Local/Programs/Python/Python38/python.exe ./VALIDACION ENTRADA DE USUARIO (DO WHILE).py  
Ingresa tu edad: 12  
Edad valida: 12  
Eres menor de edad  
Entrada no valida. Intentalo de nuevo.  
Ingresa tu edad: 15  
Edad valida: 15  
Eres menor de edad  
Entrada no valida. Intentalo de nuevo.  
Ingresa tu edad: 18  
Edad valida: 18  
Eres Mayor de edad.  
PS D:\PROGRAMACION\ProyectoProgramacion>

##### **- Creación de menús interactivos**

El while True es ideal para programas de consola que ofrecen opciones repetitivas. Es útil en los siguientes casos: sistemas de administración, juegos y herramientas interactivas.

D: > PROGRAMACION > CREACION DE MENUS INTERACTIVOS (DO WHILE).py > ...

```
1  while True:
2      print("\nMenu:")
3      print("1. Sumar")
4      print("2. Restar")
5      print("3. Salir")
6
7      opcion = input("Selecciona una opcion: ")
8
9      if opcion == "1":
10         print("Ingrese los numeros para la suma:")
11         a = int(input("Ingrese n1: "))
12         b = int(input("Ingrese n2: "))
13         suma = a + b
14         print(f"La suma es: {suma}")
15     elif opcion == "2":
16         print("Ingrese los numeros para la resta:")
17         a = int(input("Ingrese n1: "))
18         b = int(input("Ingrese n2: "))
19         suma = a - b
20         print(f"La resta es: {suma}")
21     elif opcion == "3":
22         print("Salido del programa...")
23         break # Sale del bucle si el usuario elige salir
24     else:
25         print("Opcion no valida, intenta de nuevo.")
```

Lo que muestra en la terminal es lo siguiente:

```
PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Det-Pc/AppData/Local/Programs/Python/Python37/python Ejemplo.py

Menu:
1. Sumar
2. Resta
3. Salir
Selecciona una opcion: 1
Ingrese los numeros para la suma:
Ingrese n1: 10
Ingrese n2: 11
La suma es: 21

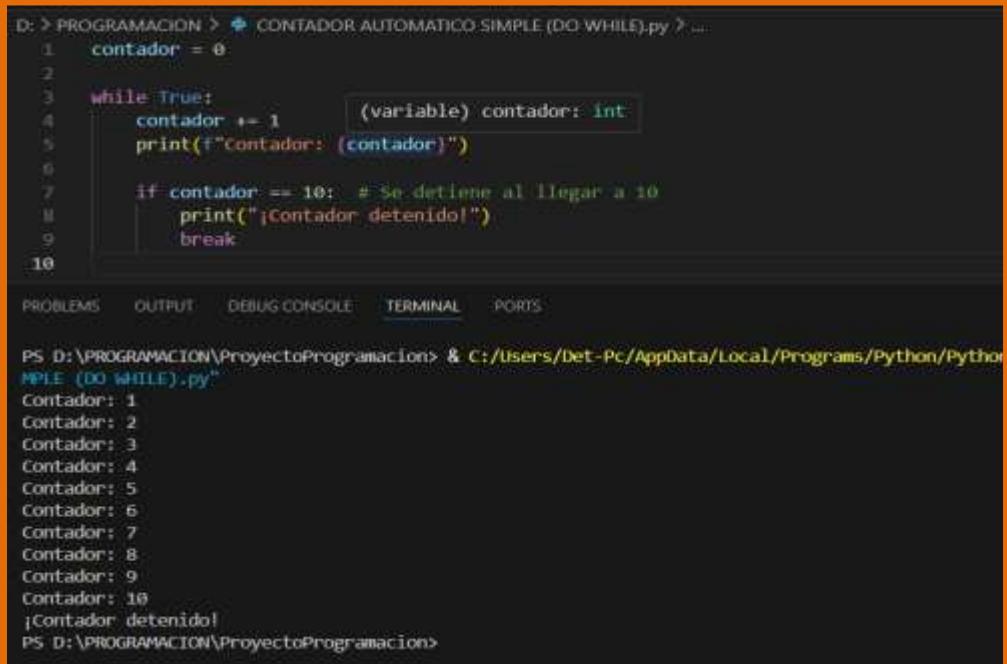
Menu:
1. Sumar
2. Resta
3. Salir
Selecciona una opcion: 2
Ingrese los numeros para la resta:
Ingrese n1: 35
Ingrese n2: 12
La resta es: 23

Menu:
1. Sumar
2. Resta
3. Salir
Selecciona una opcion: h
Opcion no valida, intenta de nuevo.

Menu:
1. Sumar
2. Resta
3. Salir
Selecciona una opcion: 1
```

#### - Contador automático simple

Este contador aumenta indefinidamente hasta que se detenga manualmente con break.



The screenshot shows a code editor window with a Python script named 'CONTADOR AUTOMATICO SIMPLE (DO WHILE).py'. The code uses a do-while loop to print a counter value from 1 to 10, then prints a message and breaks the loop. The code is as follows:

```
D:\> PROGRAMACION > ✎ CONTADOR AUTOMATICO SIMPLE (DO WHILE).py > ...
1     contador = 0
2
3     while True:
4         contador += 1      (variable) contador: int
5         print(f"Contador: {contador}")
6
7         if contador == 10: # Se detiene al llegar a 10
8             print("¡Contador detenido!")
9             break
10
```

Below the code editor, the terminal window shows the execution of the script and its output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Det-Pc/AppData/Local/Programs/Python/Python37/python Ejemplo (DO WHILE).py
Contador: 1
Contador: 2
Contador: 3
Contador: 4
Contador: 5
Contador: 6
Contador: 7
Contador: 8
Contador: 9
Contador: 10
¡Contador detenido!
PS D:\PROGRAMACION\ProyectoProgramacion>
```

## 5. Diferencias entre while True y while con una condición específica en Python.

En Python, existen dos formas principales de usar un bucle while:

**while True:** Es un bucle infinito que se ejecuta hasta que se use break para salir manualmente.

**while con una condición específica:** Se ejecuta mientras la condición dada sea True, terminando automáticamente cuando la condición se vuelve False.

### - **While True: Bucle infinito controlado por break**

- Se ejecuta indefinidamente hasta que se usa break.
- Se usa cuando no sabemos cuántas veces se ejecutará el bucle.
- Necesita una condición interna para detenerse manualmente.

**Ejemplo con while True:**

The screenshot shows a code editor window with a Python script named 'BUCLE INFINITO CONTROLADO POR BREAK (DO WHILE).py'. The code is as follows:

```
D:\> PROGRAMACION > ⚡ BUCLE INFINITO CONTROLADO POR BREAK (DO WHILE).py > ...
1 contraseña_correcta = "miky0107"
2
3 while True:
4     contraseña = input("Ingresa la contraseña: ")
5
6     if contraseña == contraseña_correcta:
7         print("✅ ¡Acceso concedido!")
8         break
9     else:
10        print("❌ Contraseña incorrecta, intenta de nuevo.")
11
12
```

Below the code, there is a terminal window showing the execution of the script. It prompts for a password and prints error messages for incorrect inputs ('Contraseña incorrecta, intenta de nuevo.') and a success message ('¡Acceso concedido!').

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Det-Pc/AppData/Local/Programs/Python/Python38-32/WINDOWS-PYTHON38/python.exe D:\PROGRAMACION\ProyectoProgramacion> BUCLE INFINITO CONTROLADO POR BREAK (DO WHILE).py
Ingresa la contraseña: 3M0HHN
❌ Contraseña incorrecta, intenta de nuevo.
Ingresa la contraseña: 12345
❌ Contraseña incorrecta, intenta de nuevo.
Ingresa la contraseña: a1b2c3d4
❌ Contraseña incorrecta, intenta de nuevo.
Ingresa la contraseña: miky0107
✅ ¡Acceso concedido!
PS D:\PROGRAMACION\ProyectoProgramacion>
```

Explicación:

- **while True:** crea un bucle infinito, ya que la condición siempre es **True**.
- El usuario ingresa una contraseña y se compara con la correcta.
- Si es incorrecta, el bucle sigue ejecutándose.
- Si es correcta, **break** detiene el bucle, permitiendo que el programa continúe o finalice.

- **While con una condición específica**

- Se ejecuta mientras la condición sea True.
- Termina automáticamente cuando la condición cambia a False.
- Más seguro para evitar bucles infinitos accidentales.

Ejemplo con while y una condición:

The screenshot shows a terminal window with the following content:

```
D: > PROGRAMACION > BUCLE CON CONDICION ESPECIFICA (DO WHILE).py > ...
1  contador = 1
2
3  while contador <= 5:
4      print(f"Contador: {contador}")
5      contador += 1
6
7  print("¡Fin del conteo!")
8
```

Below the code, there are tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected. The output section shows the execution of the script:

```
PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Det-Pc/AppData/Local/Temp/DO WHILE.py"
Contador: 1
Contador: 2
Contador: 3
Contador: 4
Contador: 5
¡Fin del conteo!
PS D:\PROGRAMACION\ProyectoProgramacion>
```

Explicación:

- **while contador <= 5:** → El bucle se ejecuta **mientras la condición sea True**.
- Cada iteración, el **contador aumenta en 1** (contador += 1).
- Cuando contador llega a 6, la condición **se vuelve False** y el bucle termina automáticamente.
- **Es más seguro** porque evita un bucle infinito accidental.

## 6. Ejemplo combinado con ambos enfoques:

The screenshot shows a code editor with a dark theme. The code is a Python script named 'EJEMPLO COMBINADO CON AMBOS ENFOQUES (DO WHILE).py'. It uses a standard 'while True' loop to repeatedly ask the user for a number between 1 and 10. Inside this loop, it includes a 'do-while' style check with 'while numero < 1 or numero > 10:' to handle invalid inputs. If the user enters a correct number, it prints a success message and breaks out of the inner loop using 'break'. If the user enters an incorrect number, it prints an error message and loops back to the input prompt. The code also increments a counter 'intentos' (attempts) each time the inner loop runs.

```
D:\> PROGRAMACION > EJEMPLO COMBINADO CON AMBOS ENFOQUES (DO WHILE).py > ...
1 intentos = 0
2 while True:
3     numero = int(input("Adivina un numero entre 1 y 10: "))
4
5     while numero < 1 or numero > 10:
6         print("Número fuera de rango, intenta de nuevo.")
7         numero = int(input("Adivina un numero entre 1 y 10: "))
8
9     intentos += 1
10    if numero == 7:
11        print(f"¡Correcto! Lo lograste en {intentos} intentos.")
12        break
13    else:
14        print("Incorrecto, intenta otra vez.")
15

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Det-Pc/AppData/Local/Programs/Python/Python312/python O WHILE.py"
Adivina un numero entre 1 y 10: 10
Incorrecto, intenta otra vez.
Adivina un numero entre 1 y 10: 9
Incorrecto, intenta otra vez.
Adivina un numero entre 1 y 10: 4
Incorrecto, intenta otra vez.
Adivina un numero entre 1 y 10: 3
Incorrecto, intenta otra vez.
Adivina un numero entre 1 y 10: 2
Incorrecto, intenta otra vez.
Adivina un numero entre 1 y 10: 7
¡Correcto! Lo lograste en 6 intentos.
PS D:\PROGRAMACION\ProyectoProgramacion>
```

Explicación:

- **while True** mantiene el juego en ejecución hasta que el usuario acierte.
- **while numero < 1 or numero > 10** impide que el usuario ingrese números fuera del rango.

## 7. Diferencias entre while True y un for en Python

En la programación de Python, tanto **while True** como **for** se utilizan para repetir código, pero tienen diferencias en uso, control y flexibilidad.

- **while True: Bucle Infinito con control manual**

Es útil cuando el número de iteraciones depende de una entrada del usuario o de un evento externo.

- Se ejecuta infinitamente hasta que se use break para salir.
- Se usa cuando no sabemos cuántas veces se ejecutará el bucle.
- Necesita una condición interna para detenerse manualmente.

### Ejemplo con while True (menú interactivo)

```
D:\> PROGRAMACION > ➜ BUCLE INFINITO CON CONTROL MANUAL (DO WHILE).py > ...
1  while True:
2      print("\nMenu de Comida:")
3      print("1. Comida rápida")
4      print("2. Comida hecha en casa")
5      print("3. Salir")
6
7      opcion = input("Selecciona una opción (1/2/3): ")
8
9      if opcion == "1":
10         print("\nComida rápida:")
11         print("a. Hamburguesa")
12         print("b. Pizza")
13         print("c. Tacos")
14         opcion_rapida = input("Selecciona un plato (a/b/c): ")
15
16         if opcion_rapida == "a":
17             print("Has elegido Hamburguesa. ¡Disfratal!")
18         elif opcion_rapida == "b":
19             print("Has elegido Pizza. ¡Buen provecho!")
20         elif opcion_rapida == "c":
21             print("Has elegido Tacos. ¡Deliciosos!")
22         else:
23             print("Opción no válida en comida rápida.")
24
25     elif opcion == "2":
26         print("\nComida hecha en casa:")
27         print("a. Spaghetti")
28         print("b. Ensalada")
29         print("c. Pollo al horno")
30         opcion_casera = input("Selecciona un plato (a/b/c): ")
31
32         if opcion_casera == "a":
33             print("Has elegido Spaghetti. ¡Qué sabroso!")
34         elif opcion_casera == "b":
35             print("Has elegido Ensalada. ¡Muy saludable!")
36         elif opcion_casera == "c":
37             print("Has elegido Pollo al horno. ¡Exquisito!")
38         else:
39             print("Opción no válida en comida hecha en casa.")
40
41     elif opcion == "3":
42         print("¡Hasta luego! Que tengas un buen día.")
43         break # Sale del bucle cuando elige "Salir"
44
45     else:
46         print("Opción no válida, por favor selecciona una opción válida.")
```

Lo que se muestra en la terminal sería lo siguiente:

```
PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Del-Pc/AppData/Local/Programs/Python/Python312/python.exe .\doWhile.py
Menu de Comida:
1. Comida rápida
2. Comida hecha en casa
3. Salir
Selecciona una opción (1/2/3): 1
Comida rápida:
a. Hamburguesa
b. Pizza
c. Tacos
Selecciona un plato (a/b/c): a
Has elegido Hamburguesa. ¡Disfratal!

Menu de Comida:
1. Comida rápida
2. Comida hecha en casa
3. Salir
Selecciona una opción (1/2/3): 2
Comida hecha en casa:
a. Spaghetti
b. Ensalada
c. Pollo al horno
Selecciona un plato (a/b/c): b
Has elegido Ensalada. ¡Muy saludable!

Menu de Comida:
1. Comida rápida
2. Comida hecha en casa
3. Salir
Selecciona una opción (1/2/3): 3
¡Hasta luego! Que tengas un buen día.
PS D:\PROGRAMACION\ProyectoProgramacion>
```

Explicación:

- **while True:** Se crea un bucle infinito que sigue ejecutándose hasta que el usuario elija salir usando la opción 3.
  - Menú principal:
    - Opción 1: Comida rápida (con subopciones de Hamburguesa, Pizza, o Tacos).
    - Opción 2: Comida hecha en casa (con subopciones de Spaghetti, Ensalada, o Pollo al horno).
    - Opción 3: Salir del menú y terminar el programa.
  - Si el usuario selecciona comida rápida o comida hecha en casa, se le presentan subopciones para elegir un plato específico dentro de esa categoría.
  - Si elige un plato válido, el programa imprime un mensaje de confirmación.
  - Si selecciona una opción no válida, el programa muestra un mensaje de error y vuelve a mostrar las opciones de comida rápida o hecha en casa.
  - Si el usuario elige Salir (opción 3), se rompe el bucle con **break**, y el programa finaliza con un mensaje de despedida.
- **for: Bucle Determinado por un rango**

Es útil cuando sabemos cuántas veces queremos repetir algo.

- Se usa cuando **sabemos de antemano** cuántas veces queremos repetir el código.
- Repite sobre una **secuencia** (lista, cadena, rango, etc.).
- No necesita break, ya que finaliza automáticamente cuando termina la secuencia.

### Ejemplo con for (Imprimir números del 1 al 5)

D: > PROGRAMACION > BUCLE DETERMINADO POR UN RANGO (DO WHILE).py > ...

```
1 for i in range(1, 6):
2     print(i)
3
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Det-Pc/AppData/Local/Programs/Python/Python39/python.exe "D:\PROGRAMACION\ProyectoProgramacion> E.py"
1
2
3
4
5
PS D:\PROGRAMACION\ProyectoProgramacion>

Explicación:

- **for i in range(1, 6):** El range(1, 6) genera una secuencia de números que empieza en 1 y termina en 5. El número 6 no está incluido en el rango.
- El bucle **for** iterará sobre esta secuencia de números, y en cada iteración, el valor de **i** tomará un número dentro de ese rango, comenzando desde 1 hasta 5.
- **print(i):** En cada iteración, el valor actual de **i** se imprimirá en la pantalla.

## Ejemplo combinado: while True y for

The screenshot shows a code editor window with the following content:

```
D: > PROGRAMACION > EJEMPLO COMBINADO WHILE TRUE Y FOR (DO WHILE).py > ...
1  while True:
2      try:
3          cantidad = int(input("¿Cuantos numeros quieres imprimir? "))
4          if cantidad <= 0:
5              print("Por favor ingresa un numero mayor que 0.")
6              continue
7          break
8      except ValueError:
9          print("Por favor ingresa un numero entero valido.")
10
11 for i in range(1, cantidad + 1):
12     print(i)
13
```

Below the code, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing the following terminal session:

```
PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Det-Pc/AppData/Local/Programs/Python/WHILE.py"
¿Cuantos numeros quieres imprimir? 5
1
2
3
4
5
PS D:\PROGRAMACION\ProyectoProgramacion>
```

Explicación:

- **while True:** Se usa para crear un bucle infinito que sigue ejecutándose hasta que se rompe manualmente con un **break**.
- Entrada del usuario: Dentro del **while True**, le pedimos al usuario que ingrese cuántos números quiere imprimir.
- **try y except:** Utilizamos un bloque **try** para manejar posibles errores si el usuario no ingresa un número entero. Si el usuario ingresa algo que no es un número, el programa le pide que ingrese un número válido.
- Condición para detener el **while True**: Si el usuario ingresa un número mayor que 0, el bucle se rompe con **break** y continuamos con el siguiente paso.
- Si el número es menor o igual a 0, el programa vuelve a preguntar por una entrada válida usando **continue**.
- **for:** Después de obtener una entrada válida, usamos un bucle **for** para imprimir los números desde 1 hasta la cantidad ingresada por el usuario.
- **range(1, cantidad + 1):** Esto crea una secuencia de números desde 1 hasta la cantidad indicada por el usuario.

## 8. Simulación de lotería en Python

```
D: > PROGRAMACION > SIMULACION DE LOTERIA (DO WHILE).py > ...
1 import random
2
3 numero_ganador = random.randint(1, 10) # Número ganador entre 1 y 10
4
5 print("¡Bienvenido a la lotería! Adivina el número (1-10).")
6
7 while True:
8     intento = int(input("Ingresa tu número: "))
9
10    if intento == numero_ganador:
11        print(f" ¡Ganaste! El número era {numero_ganador}.")
12        break # Termina el bucle si el usuario acierta
13    else:
14        print(" Número incorrecto. Intenta de nuevo.")
15
16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\PROGRAMACION\ProyectoProgramacion> & C:/Users/Det-Pc/AppData/Local/Programs/Python/Python38-32/python D:\PROGRAMACION\ProyectoProgramacion\SIMULACION DE LOTERIA (DO WHILE).py
¡Bienvenido a la lotería! Adivina el número (1-10).
Ingresa tu número: 1
Número incorrecto. Intenta de nuevo.
Ingresa tu número: 5
Número incorrecto. Intenta de nuevo.
Ingresa tu número: 7
Número incorrecto. Intenta de nuevo.
Ingresa tu número: 2
Número incorrecto. Intenta de nuevo.
Ingresa tu número: 10
Número incorrecto. Intenta de nuevo.
Ingresa tu número: 9
Número incorrecto. Intenta de nuevo.
Ingresa tu número: 8
¡Ganaste! El número era 8.
PS D:\PROGRAMACION\ProyectoProgramacion>
```

Explicación:

- El programa elige un número aleatorio entre 1 y 10.
- Usa un **while True** para seguir pidiendo números hasta que el usuario acierte.
- Si el usuario adivina, muestra un mensaje y usa **break** para salir.
- Si falla, sigue ejecutándose hasta que acierte

## **9. Conclusión:**

El **while True** es una herramienta poderosa en Python, ya que permite la ejecución continua de programas sin necesidad de establecer un número de iteraciones desde el inicio. Su flexibilidad lo hace ideal para validaciones de entrada, creación de menús, manejo de eventos en tiempo real y ejecución de servidores. Sin embargo, debe usarse con precaución para evitar bucles infinitos no controlados.

