# Universidad Tecnológica Equinoccial Tecnología Superior en Desarrollo de Software

# Programación I

**Nombr**e: Mikaela Zurita

**Paralelo:** “B”

# Do – While en Python

## ¿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.

# Estructura while True (Python).

# 

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.

## 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.

1. ***Utilidad de un while True en la programación de Python.***

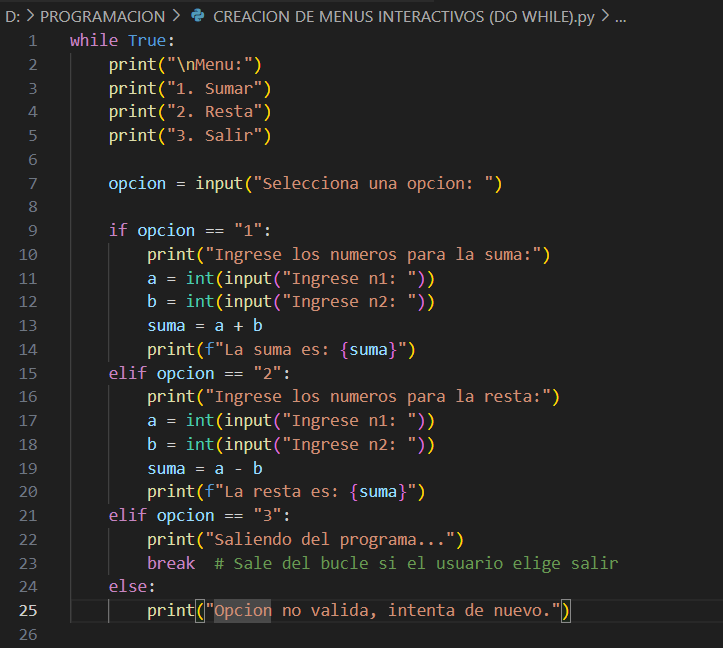
* **Validación de entrada de usuario**

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.

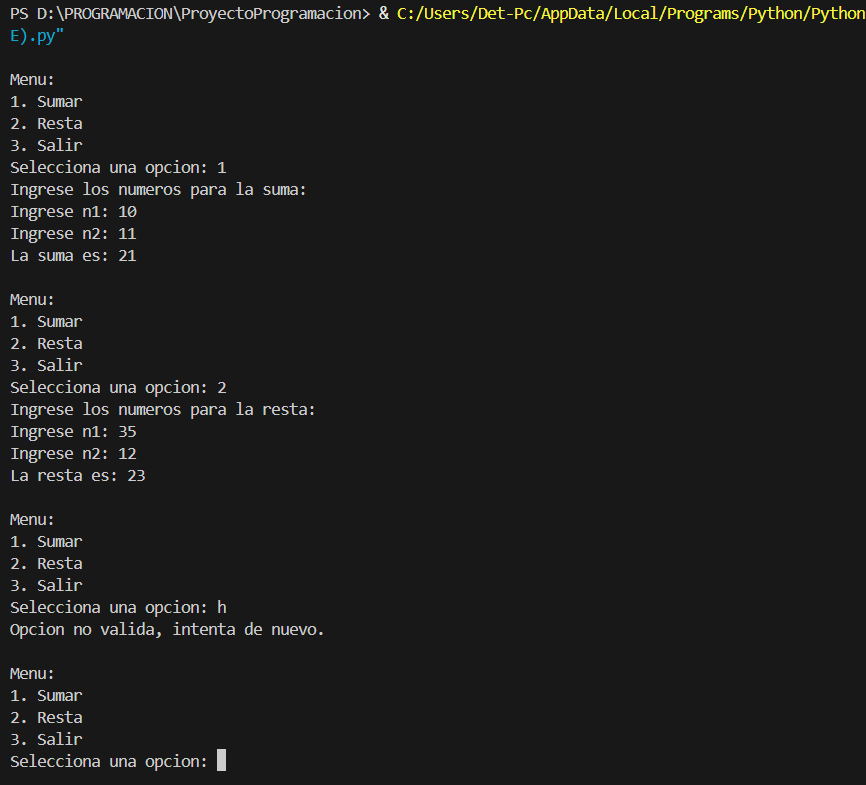


* **Creación de menús interactivos**

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.

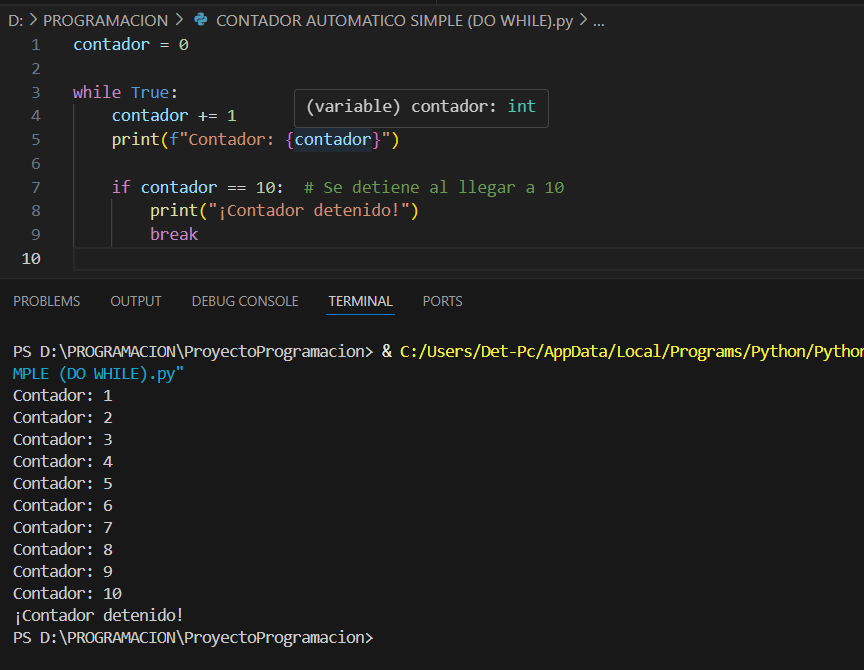


**Lo que muestra en la terminal es lo siguiente:**

****

* **Contador automático simple**

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



1. ***Diferencias entre while True y while con una condición especifica en Python***.

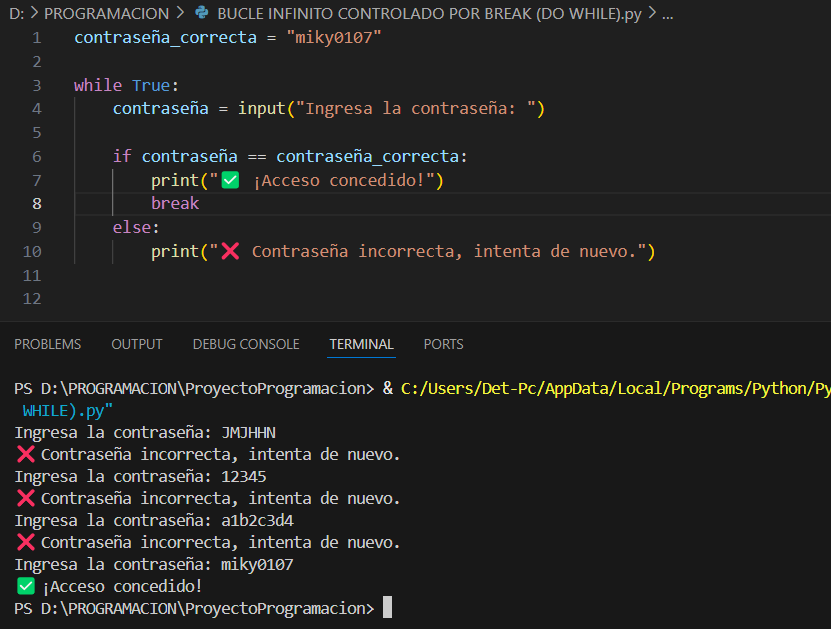
En Python, existen dos formas principales de usar un bucle while:

**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.

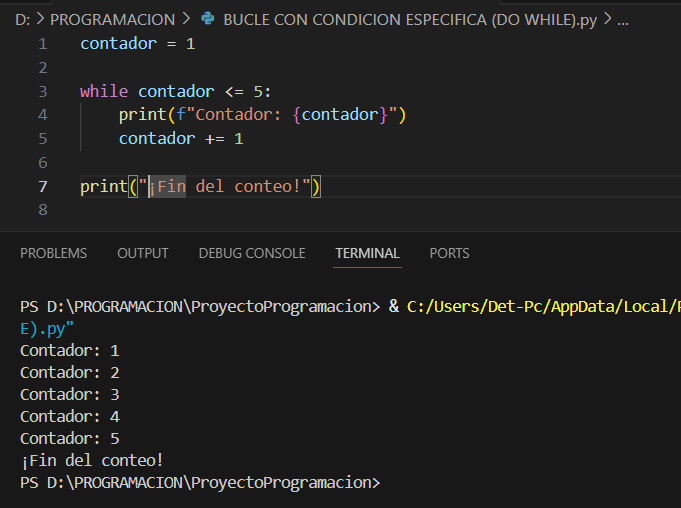
**Ejemplo con while True:**

****

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

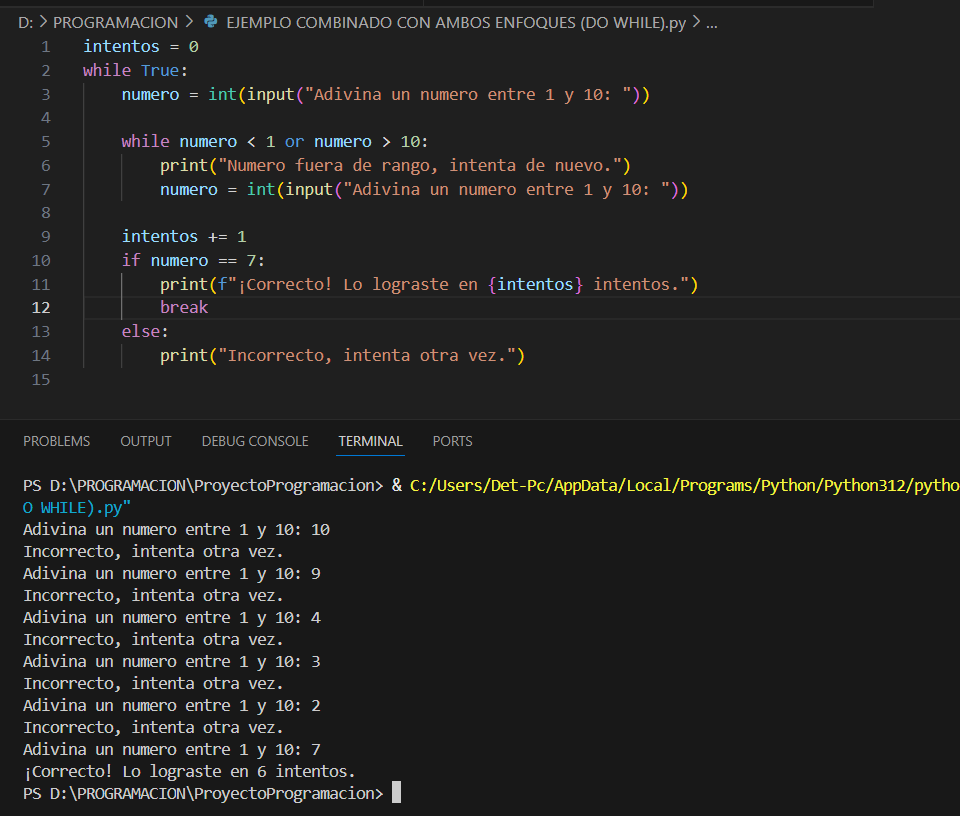
**Ejemplo con while y una condición:**

****

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.

1. ***Ejemplo combinado con ambos enfoques:***



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.

1. ***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**

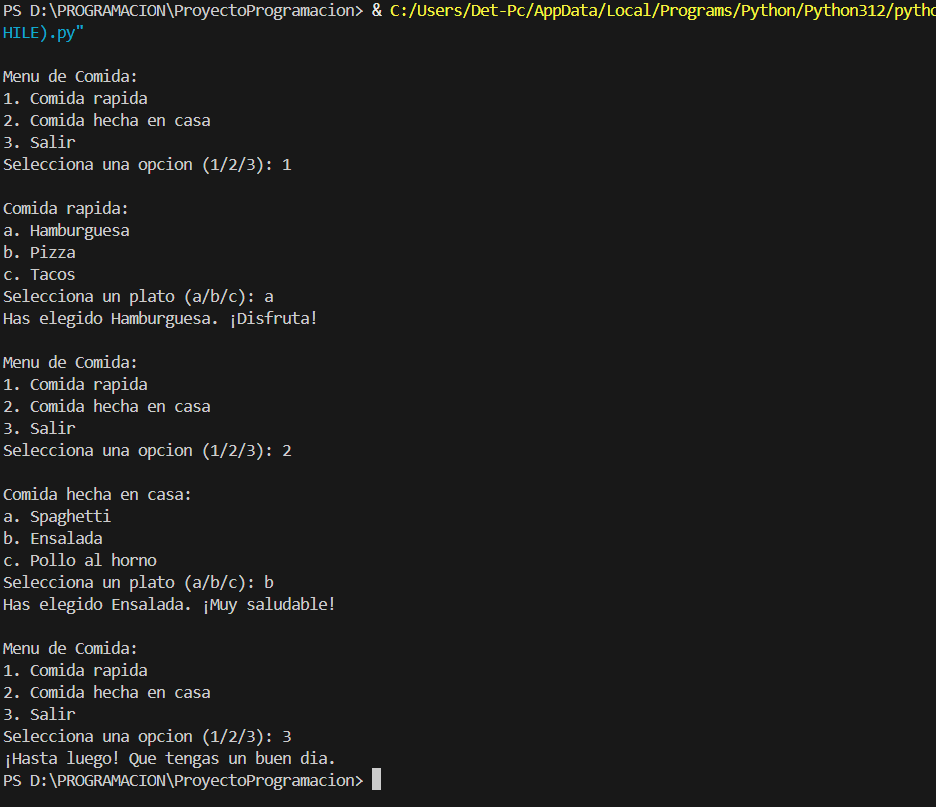
Es útil cuando el número de iteraciones depende de una entrada del usuario o de un evento externo.

* + 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)**

****

**Lo que se muestra en la terminal seria lo siguiente:**

****

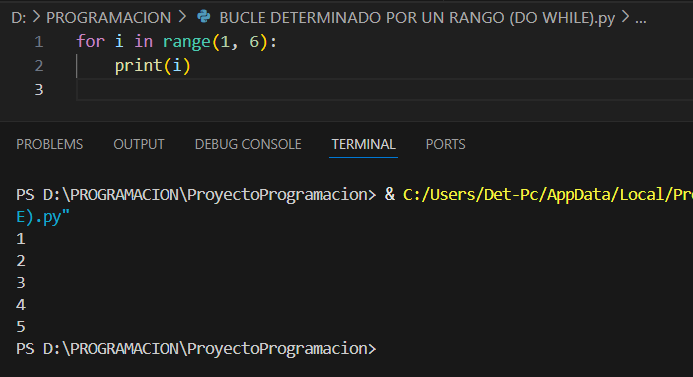
Explicación:

* + **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.

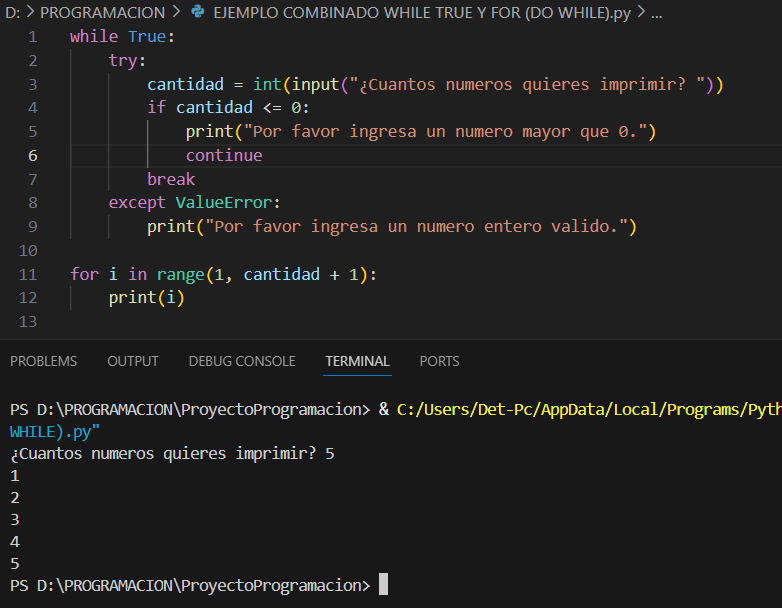
**Ejemplo con for (Imprimir números del 1 al 5)**

****

Explicación:

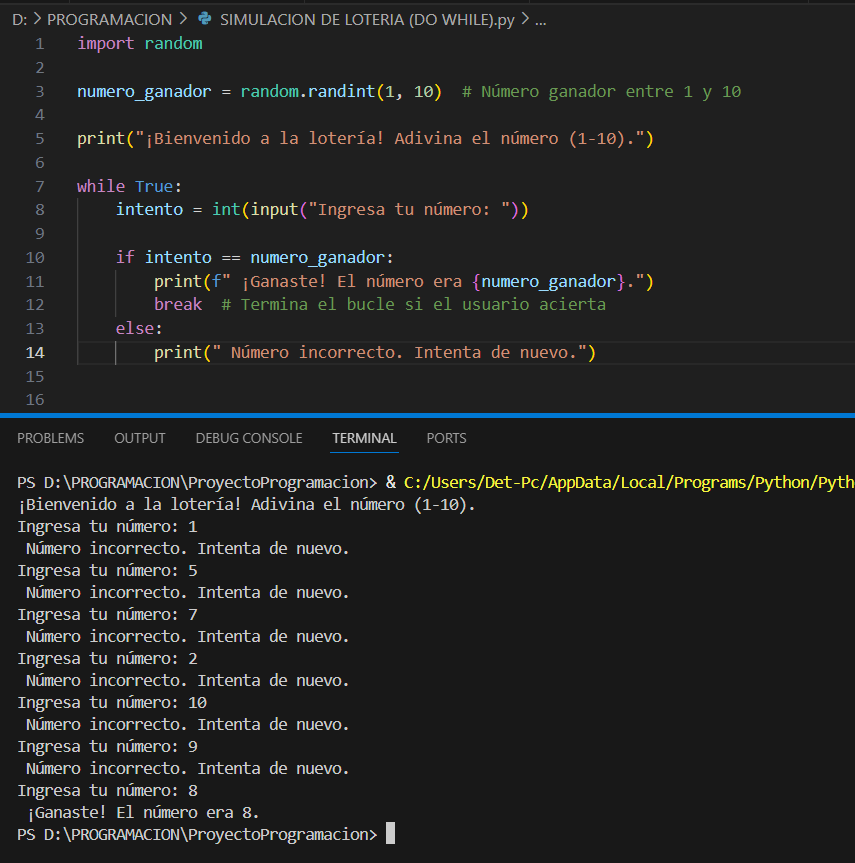
* **for i in range(1, 6):** El range(1, 6) genera una secuencia de números que empieza en 1 y termina en 5. El número 6 no está incluido en el rango.
* El bucle **for** iterará sobre esta secuencia de números, y en cada iteración, el valor de i tomará un número dentro de ese rango, comenzando desde 1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

**Ejemplo combinado: while True y for**

****

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.

1. ***Simulación de lotería en Python***

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

## 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.