FPP-PE-P7. Estructura de repetición while

ALUMNO: Alejandro Bermúdez Gorozabel

1) Crea un script que solicite un valor número entero por teclado al usuario entre 1 y 10. Si el usuario no introduce un valor correcto (fuera de rango o no numérico) se seguirá pidiendo el número hasta que sea válido.

```
a = 0;

while not (1<= a <= 10):

    a = int(input("Introduzca un numero entre 1 y 10: "))

    if not (1 < a < 10):

        print("El número debe estar entre 1 y 10.")

    else:

        print(f"Número introducido: {a}.")
```

2) Crea un script que solicite un valor numérico entero. A continuación el script calculará internamente un número aleatorio entre 0 y ese número y solicitará de forma repetida números al usuario hasta que acierte el número.

- Cuando el usuario acierte el número se saldrá del programa dando la enhorabuena.
- En cada intento fallido el script debe mostrar si el número introducido es mayor o menor al calculado y volver a solicitar un nuevo número.
- Si el usuario presiona la letra `q` el script debe salir y mostrar el número secreto.

```python
import random



numero = int(input("Introduce un número mayor que 0: "))

numeroSecreto = int(random.randint(0, numero))



while True:

    intento = input("Adivina el número (o pulsa 'q' para salir): ")



    if intento.lower() == 'q':

        print(f"Has salido del juego. El número secreto era
{numeroSecreto}.")

        break



    intento = int(intento)



    if intento < numeroSecreto:

        print("El número es mayor.")

    elif intento > numeroSecreto:

        print("El número es menor.")

    else:

        print(f"Has acertado el número {numeroSecreto}.")

        break
```

3) Crea un script que reciba como parámetro una dirección IP.

- El script comprobará utilizando una expresión regular que el parámetro tiene el formato de una dirección IP.
- Cada 10 segundos el script comprobará la disponibilidad de la IP con un ping (un único envío).
- El bucle terminará cuando la IP se encuentre disponible (con código de salida 0) mostrando un mensaje informativo.
- Si han transcurrido 10 intentos el script terminará (con código de salida `1`) informando del error.

```
# REALIZAR UN PING EN LINUX
import subprocess
exec = subprocess.run(["ping", "-c", "1", "8.8.8.8"],
stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)
# exec.returncode será igual a 0 si el ping fue correcto y distinto
en otro caso

# ESPERAR 10 SEGUNDOS
import time
time.sleep(10)   # Pausa de 10 segundos
```

4) ¡Implementa tu versión del juego de piedra-papel-tijera en Python!

- El juego solicitará al usuario una acción por teclado que podrá ser: `piedra`, `papel`, `tijera` o `salir`.
- Entrará en un bucle mientras que la opción no sea `salir`.
- En cada iteración solicitará una opción al usuario y calculará aleatoriamente una de las 3 opciones y la comparará con la opción introducida, mostrando según el caso:
  - ❌ Opción no válida. Intenta de nuevo
  - 👋 Gracias por jugar. ¡Hasta la próxima!
  - 🤝 ¡Empate!
  - 🏆 ¡Ganaste!
  - 💀 Perdiste.

```python
import random



opciones = ["piedra", "papel", "tijera"]



while True:

    jugador = input("Elige piedra, papel, tijera o salir: ").lower()



    if jugador == "salir":

        print(f"\U0001F44B Gracias por jugar. ¡Hasta la próxima!")

        break



    if jugador not in opciones:

        print(f"\u274C Opción no válida. Intenta de nuevo.")

        continue



    pc = random.choice(opciones)

    print(f"El ordenador eligió: {pc}")



    if jugador == pc:

        print(f"\U0001F91D ¡Empate!")

    elif (

        (jugador == "piedra" and pc == "tijera") or
```

```
        (jugador == "papel" and pc == "piedra") or
        (jugador == "tijera" and pc == "papel")
    ):
        print(f"\U0001F3C6 ¡Ganaste!")
    else:
        print(f"\U0001F480 Perdiste.")
```