ej_3

September 29, 2025

0.1 Flujos de Control

1) Crear una variable que contenga un elemento del conjunto de números enteros y luego imprimir por pantalla si es mayor o menor a cero

[1]:
```
# Crear una variable con un número entero
numero_entero = 10

# Imprimir si es mayor o menor a cero
if numero_entero > 0:
  print("El número es mayor a cero")
elif numero_entero < 0:
  print("El número es menor a cero")
else:
  print("El número es cero")
```

```
El número es mayor a cero
```

2) Crear dos variables y un condicional que informe si son del mismo tipo de dato

[2]:
```
# Crear dos variables
variable1 = 10
variable2 = "hola"

# Verificar si son del mismo tipo de dato
if type(variable1) == type(variable2):
  print("Las variables son del mismo tipo de dato")
else:
  print("Las variables no son del mismo tipo de dato")
```

```
Las variables no son del mismo tipo de dato
```

3) Para los valores enteros del 1 al 20, imprimir por pantalla si es par o impar

[3]:
```
# Iterar sobre los números del 1 al 20
for numero in range(1, 21):
  # Verificar si es par o impar
  if numero % 2 == 0:
    print(f"{numero} es par")
  else:
```

```
    print(f"{numero} es impar")
```

```
1 es impar
2 es par
3 es impar
4 es par
5 es impar
6 es par
7 es impar
8 es par
9 es impar
10 es par
11 es impar
12 es par
13 es impar
14 es par
15 es impar
16 es par
17 es impar
18 es par
19 es impar
20 es par
```

4) En un ciclo for mostrar para los valores entre 0 y 5 el resultado de elevarlo a la potencia igual a 3

[5]:
```
# Iterar sobre los números del 0 al 5
for numero in range(6):
  # Elevar el número a la potencia 3 e imprimir el resultado
  print(f"{numero} elevado a la potencia 3 es: {numero**3}")
```

```
0 elevado a la potencia 3 es: 0
1 elevado a la potencia 3 es: 1
2 elevado a la potencia 3 es: 8
3 elevado a la potencia 3 es: 27
4 elevado a la potencia 3 es: 64
5 elevado a la potencia 3 es: 125
```

5) Crear una variable que contenga un número entero y realizar un ciclo for la misma cantidad de ciclos

[6]:
```
# Crear una variable con un número entero
cantidad_ciclos = 5

# Realizar un ciclo for la misma cantidad de ciclos
for i in range(cantidad_ciclos):
  print(f"Ciclo número: {i+1}")
```

```
Ciclo número: 1
Ciclo número: 2
```

```
Ciclo número: 3
Ciclo número: 4
Ciclo número: 5
```

6) Utilizar un ciclo while para realizar el factorial de un número guardado en una variable, sólo si la variable contiene un número entero mayor a 0

[7]:
```python
# Crear una variable para el factorial
numero_factorial = 5
factorial = 1

# Verificar si es un número entero mayor a 0
if isinstance(numero_factorial, int) and numero_factorial > 0:
  # Realizar el factorial con un ciclo while
  i = 1
  while i <= numero_factorial:
    factorial *= i
    i += 1
  print(f"El factorial de {numero_factorial} es: {factorial}")
else:
  print("La variable debe contener un número entero mayor a 0")
```

```
El factorial de 5 es: 120
```

7) Crear un ciclo for dentro de un ciclo while

[8]:
```python
# Crear un ciclo while
contador_while = 0
while contador_while < 3:
  print(f"Ciclo while: {contador_while}")
  # Crear un ciclo for dentro del while
  for contador_for in range(2):
    print(f"  Ciclo for: {contador_for}")
  contador_while += 1
```

```
Ciclo while: 0
  Ciclo for: 0
  Ciclo for: 1
Ciclo while: 1
  Ciclo for: 0
  Ciclo for: 1
Ciclo while: 2
  Ciclo for: 0
  Ciclo for: 1
```

8) Crear un ciclo while dentro de un ciclo for

[9]:
```python
# Crear un ciclo for
for contador_for in range(3):
  print(f"Ciclo for: {contador_for}")
```

```
  # Crear un ciclo while dentro del for
  contador_while = 0
  while contador_while < 2:
    print(f"  Ciclo while: {contador_while}")
    contador_while += 1
```

```
Ciclo for: 0
  Ciclo while: 0
  Ciclo while: 1
Ciclo for: 1
  Ciclo while: 0
  Ciclo while: 1
Ciclo for: 2
  Ciclo while: 0
  Ciclo while: 1
```

9) Imprimir los números primos existentes entre 0 y 30

[10]:
```
# Función para verificar si un número es primo
def es_primo(numero):
  if numero < 2:
    return False
  for i in range(2, int(numero**0.5) + 1):
    if numero % i == 0:
      return False
  return True

# Imprimir los números primos entre 0 y 30
print("Números primos entre 0 y 30:")
for numero in range(31):
  if es_primo(numero):
    print(numero)
```

```
Números primos entre 0 y 30:
2
3
5
7
11
13
17
19
23
29
```

10) ¿Se puede mejorar el proceso del punto 9? Utilizar las sentencias break y/ó continue para tal fin

[11]:
```python
# Función optimizada para verificar si un número es primo
def es_primo_optimizado(numero):
  if numero < 2:
    return False
  if numero == 2:
    return True
  if numero % 2 == 0:
    return False
  for i in range(3, int(numero**0.5) + 1, 2):
    if numero % i == 0:
      return False
  return True

# Imprimir los números primos entre 0 y 30 utilizando la función optimizada
print("Números primos entre 0 y 30 (optimizado):")
for numero in range(31):
  if es_primo_optimizado(numero):
    print(numero)
```

```
Números primos entre 0 y 30 (optimizado):
2
3
5
7
11
13
17
19
23
29
```

11) En los puntos 9 y 10, se diseño un código que encuentra números primos y además se lo optimizó. ¿Es posible saber en qué medida se optimizó?

[11]:
```python
import time

# Medir el tiempo de ejecución de la función original
start_time = time.time()
print("Ejecutando función original...")
for numero in range(101): # Aumentamos el rango para una mejor comparación
    es_primo(numero)
end_time = time.time()
tiempo_original = end_time - start_time
print(f"Tiempo de ejecución (original): {tiempo_original:.6f} segundos")

# Medir el tiempo de ejecución de la función optimizada
start_time = time.time()
print("Ejecutando función optimizada...")
```

```
for numero in range(101): # Mismo rango para comparación
    es_primo_optimizado(numero)
end_time = time.time()
tiempo_optimizado = end_time - start_time
print(f"Tiempo de ejecución (optimizado): {tiempo_optimizado:.6f} segundos")

# Comparar los tiempos
if tiempo_original > 0:
  mejora_porcentaje = ((tiempo_original - tiempo_optimizado) / tiempo_original)␣
 ↪* 100
  print(f"La optimización mejoró el rendimiento en aproximadamente un␣
 ↪{mejora_porcentaje:.2f}%")
else:
  print("No se pudo calcular la mejora porcentual (tiempo original es cero)")
```

```
Ejecutando función original…
Tiempo de ejecución (original): 0.000409 segundos
Ejecutando función optimizada…
Tiempo de ejecución (optimizado): 0.000232 segundos
La optimización mejoró el rendimiento en aproximadamente un 43.36%
```

12) Aplicando continue, armar un ciclo while que solo imprima los valores divisibles por 12, dentro del rango de números de 100 a 300

[12]:
```
# Ciclo while para imprimir números divisibles por 12 entre 100 y 300
numero = 100
while numero <= 300:
  # Si el número no es divisible por 12, saltar a la siguiente iteración
  if numero % 12 != 0:
    numero += 1
    continue
  # Si es divisible por 12, imprimirlo
  print(numero)
  numero += 1
```

```
108
120
132
144
156
168
180
192
204
216
228
240
252
264
```

276
288
300

13) Utilizar la función **input()** que permite hacer ingresos por teclado, para encontrar números primos y dar la opción al usario de buscar el siguiente

[ ]:
```
# Función para verificar si un número es primo
def es_primo(numero):
  if numero < 2:
    return False
  for i in range(2, int(numero**0.5) + 1):
    if numero % i == 0:
      return False
  return True

# Utilizar input para encontrar números primos
numero_actual = 0
while True:
  if es_primo(numero_actual):
    print(f"Número primo encontrado: {numero_actual}")
    respuesta = input("¿Buscar el siguiente número primo? (s/n): ")
    if respuesta.lower() != 's':
      break
  numero_actual += 1
```

```
Número primo encontrado: 2
Número primo encontrado: 3
Número primo encontrado: 5
Número primo encontrado: 7
Número primo encontrado: 11
Número primo encontrado: 13
Número primo encontrado: 17
Número primo encontrado: 19
Número primo encontrado: 23
Número primo encontrado: 29
```

14) Crear un ciclo while que encuentre dentro del rango de 100 a 300 el primer número divisible por 3 y además múltiplo de 6

[14]:
```
# Ciclo while para encontrar el primer número entre 100 y 300 divisible por 3 y␣
 ↪múltiplo de 6
numero = 100
while numero <= 300:
  # Verificar si es divisible por 3 y múltiplo de 6
  if numero % 3 == 0 and numero % 6 == 0:
    print(f"El primer número divisible por 3 y múltiplo de 6 entre 100 y 300 es:
 ↪ {numero}")
    break # Salir del ciclo una vez encontrado el número
```

```
  numero += 1
```

```
El primer número divisible por 3 y múltiplo de 6 entre 100 y 300 es: 102
```