DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

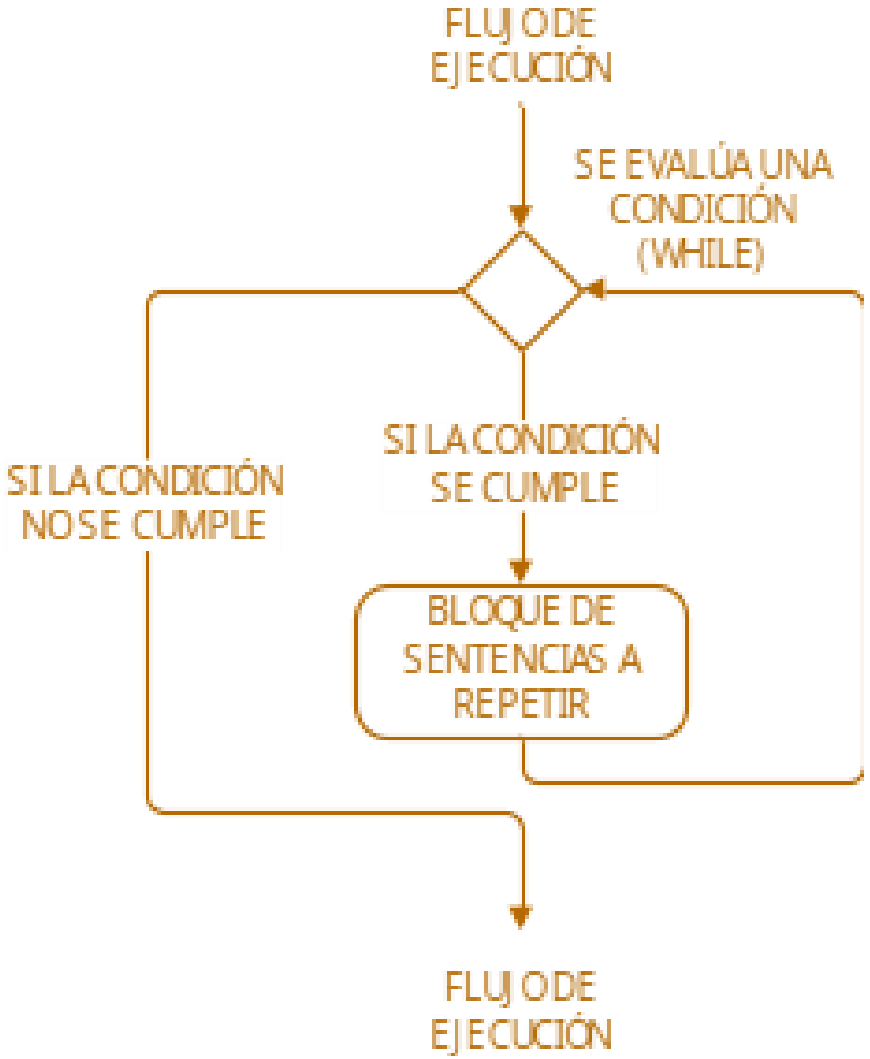
ITERACIÓN

10145 - FUNDAMENTOS DE PROGRAMACIÓN PARA INGENIERÍA

RESUMEN DE CONTENIDOS

ITERACIÓN

- Flujo de una sentencia iterativa (`while`)
  - A diferencia de un `if`, cuando un bloque de sentencias condicionado por un `while` alcanza su fin **la condición se vuelve a evaluar**
  - Esto implica que el código puede **repetirse**
  - Pero en algún momento se debe **encontrar una forma de salir** del `while`



SENTENCIA `while`

- Estructura de la sentencia `while`

```
<Sentencias previas>
while <condición>:
    # Se ejecuta si la condición se cumple
    <Bloque de sentencias a repetir>
<Sentencias después del ciclo>
```

TRAZAS

- Analicemos el programa con **numero = 5**

```
numero = 5
i = 0
suma = 0
while i <= numero :
    suma = suma + i
    i = i + 1
# SALIDA
print ("La suma de los primeros", numero)
print ("números, es: ", suma)
```

Variable\Valor	Iteración						
	Inicio	1	2	3	4	5	6
numero	5	5	5	5	5	5	5
suma	0	0	1	3	6	10	15
i	0	1	2	3	4	5	6

TRAZAS

- Analicemos el programa con **numero = 5**

```
numero = 5
i = 0
suma = 0
while i <= numero :
    suma = suma +
```

Variable\Valor	Iteración						
	Inicio	1	2	3	4	5	6
numero	5	5	5	5	5	5	5
suma	0	0	1	3	6	10	15

Al proceso de revisar manualmente el comportamiento del código en base a las variables involucradas, se le conoce como realizar una **traza del programa**

Y es una herramienta vital para entender qué es lo que hace un código

TAUTOLOGÍAS

- ¿Qué pasa si colocamos una tautología como condición?

```
i  = 0
while i < 10 :
    print('Debo practicar programación')
```

- El mensaje `'Debo practicar programación'` se repetirá por siempre
- El programa seguirá iterando hasta que **forcemos una detención**
  - En estos casos, es posible detener la ejecución de un programa con el comando **ctrl + c**
- Si tenemos una sentencia `while` de la cual es imposible salir durante una ejecución normal del programa significa que hemos creado **ciclo infinito**

SENTENCIA `for-in`

- La estructura **`for-in`** nos permite realizar iteración donde una variable irá tomando distintos valores para cada ciclo

- La **sintaxis** es la siguiente:

```
<sentencias previas>
for <identificador> in <elemento iterable> :
      <operaciones a realizar en el ciclo>

<Sentencias siguientes>
```

COMPARANDO `while` CON `for-in`

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

```
texto = input("Ingrese un texto: ")
i = 0
while i < len(texto):
   print(texto[i])
   i= i +1
```

```
texto = input("Ingrese un texto: ")
for caracter in texto:
      print(caracter)
```

FUNCIÓN `range()`

- La función `range()` tiene tres parámetros, uno mandatorio (obligatorio) y dos opcionales:
  - **Inicio:** Corresponde al primer valor del elemento iterable
    - Este parámetro es opcional y si se omite, se asume que el valor de inicio será `0`
  - **Fin:** Corresponde al valor hasta el que esperamos llegar
    - Este parámetro **no puede omitirse** y `range()` creará elementos iterables hasta `fin - 1`
  - **Salto:** Indica, de cuánto es el salto entre cada elemento del iterable
    - Este parámetro es opcional y si se omite, se asume que el valor de inicio será 1.

EJERCICIOS

Ejercicio 1

- Construya en Python un programa para calcular la siguiente serie finita, para los primeros n números naturales

$$\sum_{k=1}^{n} \frac{k^2 + 2}{k^3 + 6k}$$

Ejercicio 1

- DATOS ENTRADA:
  - Número natural
- DATOS SALIDA:
  - Suma de los primeros n términos (variable acumuladora)
- ALGORITMO:
  1. Solicitar un número natural.
  2. Inicializar iterador (=1) y acumulador (=0).
  3. Mientras el iterador sea menor o igual al número ingresado, realizar 4 y 5. En caso contrario ir a 6.
  4. Sumar a la variable acumuladora el valor de la expresión evaluada con el valor actual del iterador.
  5. Incrementar el valor del iterador e ir a 3.
  6. Mostrar por consola el valor de la suma (acumulador).

Ejercicio 1

```
# DATOS DE ENTRADA
# Solicita un número
numero = int(input("Ingrese el número de términos de la serie: "))

# PROCESAMIENTO
# Inicializa la variable iteradora k = 1
# Inicializa la variable acumuladora suma = 0
while k <= numero:
        # Calcula los resultados parciales
        suma = suma + ((k ** 2 + 2) / (k ** 3 + 6 * k))
        # Incremeta el iterador
        k = k + 1



# SALIDA
print("La suma de los primeros", numero, "términos de la serie es: ", suma)
```

Ejercicio 2

- Escriba en Python un programa que solicite por teclado dos números enteros positivos y los multiplique mediante sumas sucesivas

  - Ejemplo:

    $3 * 4 = 3 + 3 + 3 + 3$

Ejercicio 2

- DATOS ENTRADA:
  - Dos números enteros positivos
- DATOS SALIDA:
  - El producto de los números ingresados.
- ALGORITMO:
  1. Solicitar un multiplicando y un multiplicador.
  2. Inicializar iterador (=0) y acumulador (=0).
  3. Mientras el iterador sea menor al multiplicador ingresado, realizar 4 y 5. En caso contrario ir a 6.
  4. Sumar a la variable acumuladora el multiplicando.
  5. Incrementar el valor del iterador e ir a 3.
  6. Mostrar por consola el producto (acumulador).

Ejercicio 2


```
# DATOS DE ENTRADA
# Solicita un número
multiplicando = int(input("Ingrese un multiplicando: "))
# Solicita otro número
multiplicador = int(input("Ingrese un multiplicador: "))
# PROCESAMIENTO
# Inicializa una variable acumuladora
producto = 0
# Inicializa una variable iteradora
 i = 0
while i < multiplicador:
        # Calcula los resultados parciales
        producto = producto + multiplicando
        # Incremeta el iterador
        i = i + 1
# SALIDA
print("El resultado de la multiplicación es: " , producto)
```

Ejercicio 3

- Calcule el cuadrado de un número entero mediante sumas sucesivas (considere números enteros).

Ejemplos:

$$3^2 = 3 + 3 + 3 = 9$$

$$(-7)^2 = 7 + 7 + 7 + 7 + 7 + 7 + 7 = 49$$

Ejercicio 3

- DATOS ENTRADA:
  - Un número entero
- DATOS SALIDA:
  - Cuadrado del número ingresado.
- ALGORITMO:
  1. Solicitar un número entero.
  2. Si el número es negativo, entonces cambiarle el signo.
  3. Inicializar iterador (=1) y acumulador (=0).
  4. Mientras el iterador sea menor o igual al número ingresado (positivo), realizar 5 y 6. En caso contrario ir a 7.
  5. Sumar a la variable acumuladora el número (positivo).
  6. Incrementar el valor del iterador e ir a 4.
  7. Mostrar por consola el valor del cuadrado (acumulador).

Ejercicio 3


```
# DATOS DE ENTRADA
numero = int(input("Ingrese un número: "))

#PROCESAMIENTO
# Calcula el valor absoluto
if numero < 0:
    numero = -numero

# Calcula el cuadrado mediante sumas
resultado = 0
i = 0
while (i < numero):
    # Acumula los resultados parciales
    resultado = resultado + numero
    i = i + 1

#SALIDA
print("El cuadrado del número es: ",resultado)
```

Ejercicio 4

- Construya un programa en Python que calcule y muestre por pantalla el factorial de un número entero mayor o igual que cero.

Ejercicio 4

- DATOS ENTRADA:
  - Un número entero
- DATOS SALIDA:
  - El factorial del número ingresado.
- ALGORITMO:
  1. Solicitar un número entero no negativo.
  2. Inicializar iterador (=número ingresado) y acumulador (=1).
  3. Mientras el iterador sea mayor que uno, realizar 4 y 5. En caso contrario ir a 6.
  4. Multiplicar la variable acumuladora por el iterador.
  5. Decrementar en uno el valor del iterador e ir a 3.
  6. Mostrar por consola el valor del factorial (acumulador).

Ejercicio 4

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

```
# DATOS DE ENTRADA
numero = int(input("Ingrese un número natural: "))

# PROCESAMIENTO
factorial = 1
while numero > 1:
    factorial = factorial * numero
    numero = numero - 1

# SALIDA
print("El factorial del número ingresado es:", factorial)
```

Ejercicio 5

- Construya un programa en Python que calcule el máximo común divisor entre dos números, mediante el algoritmo de Euclides (considere que se ingresa primero el mayor de los números).

Ejercicio 5

- DATOS ENTRADA:
  - Dos números enteros
- DATOS SALIDA:
  - Máximo común divisor.
- ALGORITMO:
  1. Solicitar dos números ordenados (dividendo y divisor).
  2. Mientras el resto de la división sea distinto de cero, realizar 3 . En caso contrario ir a 4.
  3. Asignar divisor a dividendo y asignar resto a divisor.
  4. Mostrar por consola el M.C.D. (divisor).

Ejercicio 5


```
# DATOS DE ENTRADA
dividendo = int(input("ingrese el mayor número: "))
divisor = int(input("ingrese otro número: "))

# PROCESAMIENTO
while (dividendo % divisor) != 0:
        # Almacena el divisor
        auxiliar = divisor
        # Asigna el resto al divisor
        divisor = dividendo % divisor
        # Asigna el divisor al dividendo
        dividendo = auxiliar

# SALIDA
print("El M.C.D. es: ", divisor)
```

EJERCICIOS PROPUESTOS

- Construya un programa en Python que calcule la multiplicación de dos números enteros positivos usando solamente operadores de suma y resta. Por ejemplo:
  - 5 * 4 = 5 + 5 + 5 + 5 = 4 + 4 + 4 + 4 + 4 = 20

EJERCICIOS PROPUESTOS

1. Modifique el código anterior para que calcule la división entera y el resto de números enteros positivos

1. Modifique el código anterior para que calcule la multiplicación utilizando cualquier par de números enteros (positivos o negativos)

1. Modifique el código anterior para que, en caso de que alguno de los números sea flotante, el programa lo informe y pida nuevamente el número

¿CONSULTAS?