DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

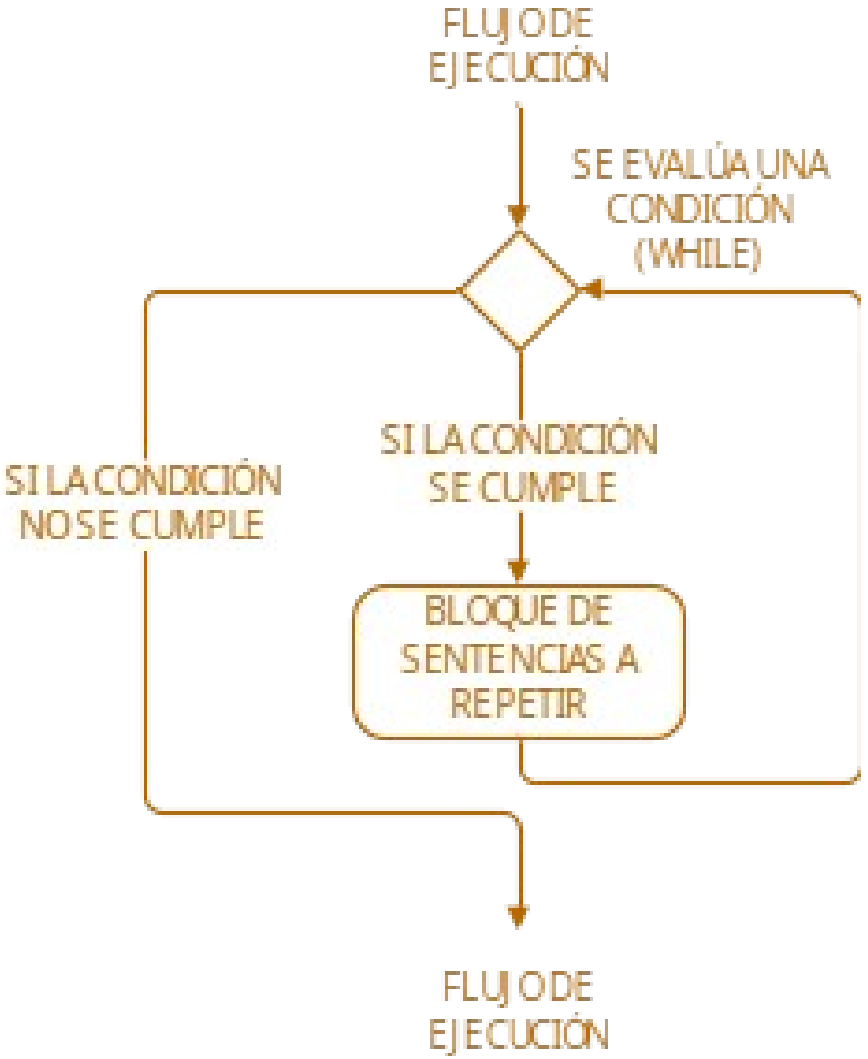
ITERACIÓN

10145 - FUNDAMENTOS DE PROGRAMACIÓN PARA INGENIERÍA
10110 – FUNDAMENTOS DE COMPUTACIÓN Y PROGRAMACIÓN

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

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

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

3. 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?