FUNDAMENTOS DE ALGORITMOS

MÓDULO 1

RECURSO DE APRENDIZAJE 5

ESTRUCTURAS DE CONTROL EN PYTHON

ITERACIÓN

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Informática

Programación Estructurada

- Recordemos que la programación estructurada es un **paradigma** orientado a mejorar la claridad, calidad y tiempo de desarrollo de un **programa** recurriendo únicamente al uso de **subrutinas (funciones)** y **3 estructuras básicas de control** :

    1. **Secuencia**
    2. **Selección**
    3. **Iteración**

- El lenguaje Python incluye estos tres tipos de **estructuras de control.**

ESTRUCTURAS ITERATIVAS

Introducción

- Los programas desarrollados hasta ahora han examinado conceptos de programación, tales como **entrada** de datos, **salida** de resultados, **asignaciones**, **expresiones** y **operaciones**, sentencias **secuenciales** y de **selección**.

- Sin embargo, muchos problemas requieren que algunos cálculos o secuencia de **instrucciones** se **repitan** una y otra vez, utilizando diferentes conjuntos de datos. Para este tipo de problemas usaremos las **estructuras iterativas**.

Ejemplos

- Algunos ejemplos de tareas **repetitivas** de procesamiento de datos incluyen:
  - **Validación de datos entrada hasta que se introduce una entrada aceptable**; por ejemplo leer una contraseña ingresada hasta que sea válida.
  - **Suma de un conjunto de datos ingresados**; por ejemplo sumar las notas finales de cada estudiante matriculado en una asignatura para calcular el promedio general del curso.
  - **Conteo de datos ingresados que cumplen o no con una determinada condición**; por ejemplo contar la cantidad de estudiantes que aprobaron y reprobaron una asignatura.

Estructuras Iterativas

- Las **estructuras iterativas** permiten **repetir** la ejecución de una o varias **instrucciones** un determinado **número de veces**.

- Esta estructuras se denominan **bucles**, **ciclos** o **loops** y se denomina **iteración** al hecho de repetir la ejecución de una secuencia de **instrucciones**.

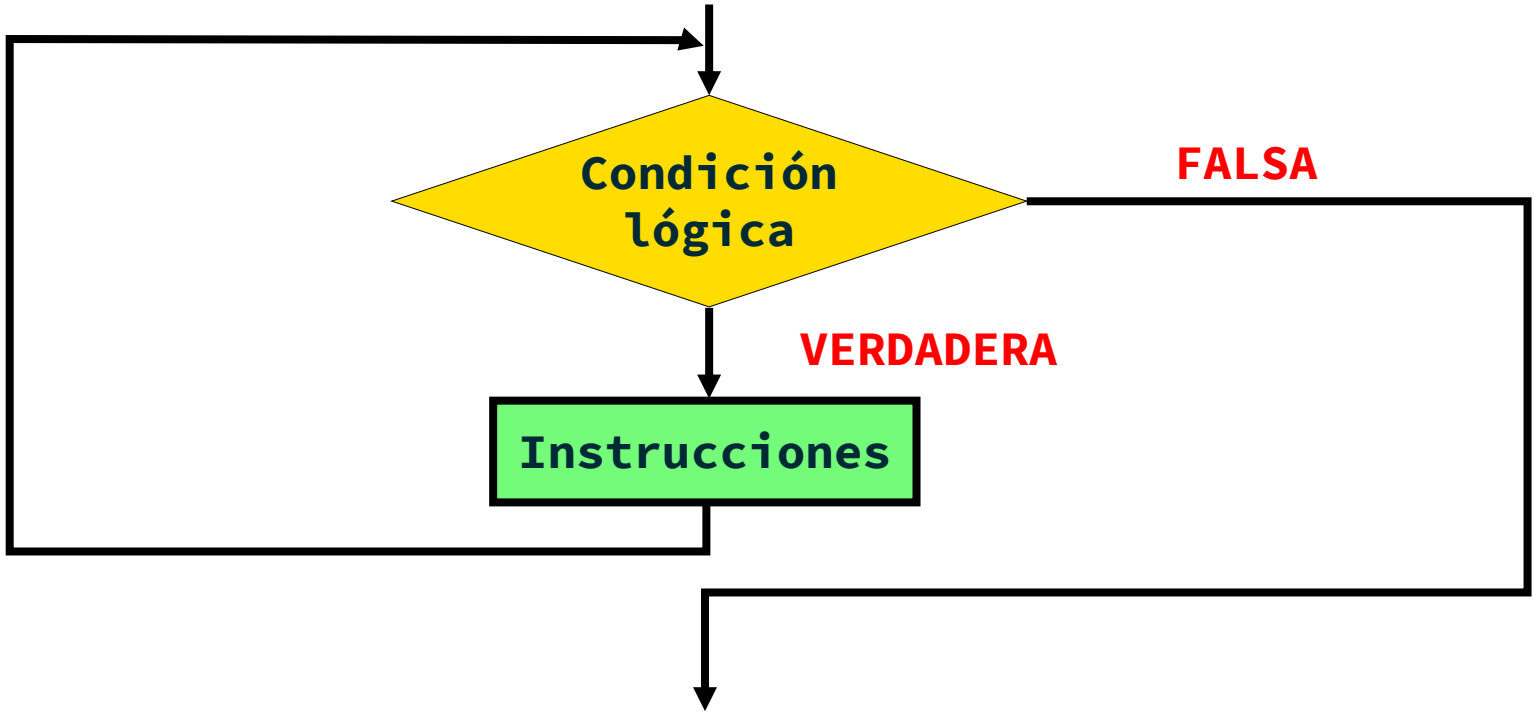
- El lenguaje de programación **Python** incluye 2 **sentencias iterativas** :

  1. **while**
  2. **for - in**

while

Estructuras Iterativa while

- La estructura iterativa **while** permite **repetir** la ejecución de una o más **instrucciones, MIENTRAS** se cumpla una determinada **condición lógica**. El siguiente **diagrama de flujo** representa esta estructura iterativa.

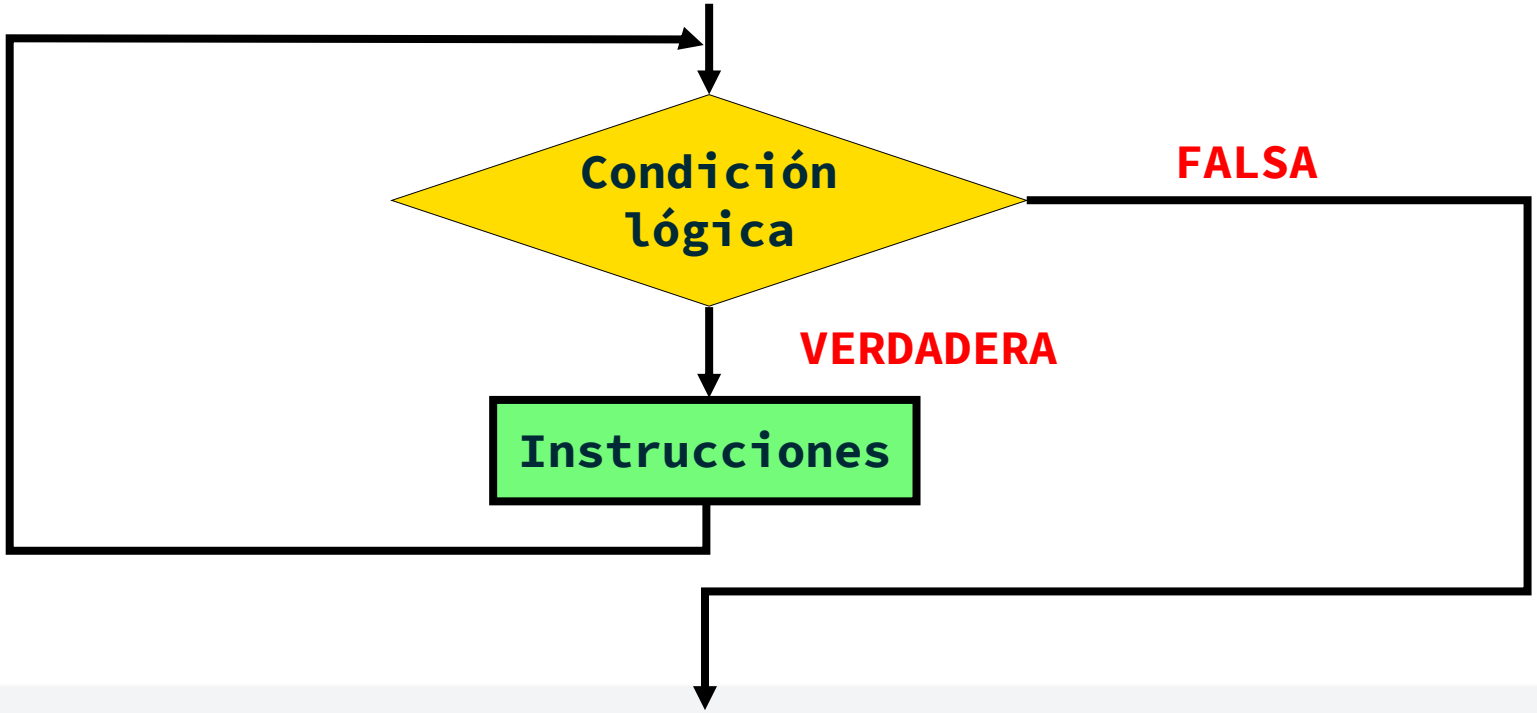


Inicio del Ciclo – Iteración – Fin del Ciclo

- En la estructura iterativa **while** se evalúa la **condición lógica** y si es **VERDADERA** se ejecuta la secuencia de **instrucciones** dentro del **ciclo** a esto se le denomina **iteración.**
- Este proceso se repite una y otra vez hasta que la **condición lógica** sea **FALSA,** en este caso el **ciclo finaliza.**

Ejecución de un ciclo 0 veces

- Observe que en una estructura **while** (**mientras**) la primera cosa que sucede es la evaluación de la **condición lógica**; si esta es evaluada como **FALSA** en ese punto, entonces las **instrucciones** dentro del **ciclo NUNCA se ejecutarán.**



Sintaxis while en Python

```
while condición lógica :
    instrucción 1
    instrucción 2
       …
    instrucción N
```

MIENTRAS condición es verdadera ejecutar instrucciones dentro del bloque

- Si la **condición lógica** al ser evaluada entrega como resultado el **valor True** se ejecutan las **instrucciones** dentro del **while**, en caso contrario **finaliza la ejecución del ciclo**.

- Las **instrucciones** dentro del **while** se **escriben** con un **tabulado mayor** que el de la línea que contiene la **condición lógica** formando un **BLOQUE**.

- En este tipo de ciclo **pre-probado**, las **instrucciones** que contiene se pueden repetir de **0 a N veces**.

Ej #1 : Imprimir Números del 1 al 10

- Escriba un **programa** que muestre por pantalla los números enteros desde el 1 al 10. La salida de este **programa** será la que se muestra a continuación:

```
1 2 3 4 5 6 7 8 9 10
```

Solución en Python

```
1   # valor inicial de x es 1
2   x = 1
3
4   # se itera - repite mientras x sea menor o igual a 10
5   while x <= 10 :
6
7       # se escribe el valor de x
8       # y nos quedamos en la misma línea
9       print( x , end = " ")
10
11      # el valor de x se aumenta en 1
12      # a continuación el control del programa vuelve a la línea 4
13      x = x + 1
```

Salida del programa

```
1 2 3 4 5 6 7 8 9 10
```

Variable de Control de Ciclo

- Este **tipo de variables** nos permiten **controlar** el **número de veces** que se ejecutarán las **instrucciones** dentro de un **ciclo**.

```
1  x = 1
2  while x <= 10 :
3      print(x,end=" ")
4      x = x + 1
```

En el ejemplo previo la **variable de control de ciclo** es **x**.

- A la **variable x** se le asigna antes del **ciclo** el **valor inicial 1** (**ver línea 1**).

- **MIENTRAS** esta variable **x** sea **menor o igual a 10 (ver línea 2)** se imprimirá su valor por pantalla (**ver línea 3**) y luego se aumentará su **valor** previo en **una unidad (ver línea 4).**

- El **número de veces** que se ejecutarán las instrucciones dentro de este **ciclo** es igual a **10.**

Ej#1 Mostrar números naturales 1..N

ESCRIBA UN PROGRAMA EN PYTHON QUE:

- Muestre por pantalla los primeros N números naturales.

ENTRADA:

- Un número entero correspondiente a N. **Se asegura N >= 1**

SALIDA:

- Los primeros N números naturales, tal como se muestra en los ejemplos.

Entrada	Resultado
5	1 2 3 4 5
10	1 2 3 4 5 6 7 8 9 10

Ej#2 Mostrar pares entre 1 y N

ESCRIBA UN PROGRAMA EN PYTHON QUE:

- Muestre por pantalla los números pares hasta N.

ENTRADA:

- Un número entero correspondiente a N. **Se asegura N > 1.**

SALIDA:

- Los números pares hasta N, tal como se muestra en los ejemplos.

Entrada	Resultado
11	2 4 6 8 10
2	2

Ej#3 Mostrar primeros 10 múltiplos de N

ESCRIBA UN PROGRAMA EN PYTHON QUE: Muestre por pantalla los primeros 10 múltiplos de un numero N leído.

ENTRADA: Un número entero correspondiente a N.

SALIDA: Los 10 primeros múltiplos de N, tal como se muestra en los ejemplos.

Entrada	Resultado
5	Primeros 10 múltiplos de 5 : 5 10 15 20 25 30 35 40 45 50

for - in

Estructuras Iterativa for - in

- La estructura iterativa **for - in** permite **repetir** una o más **instrucciones PARA CADA UNO DE LOS ELEMENTOS** que contiene el **objeto** sobre el cual la aplicamos.

- Ese **objeto** puede ser un **rango (secuencia ordenada de valores)** o una **estructura de datos compleja** como las **cadenas**, **tuplas**, **conjuntos, listas o diccionarios** (que estudiaremos más adelante).

Sintaxis for – in en Python

Sintaxis :

```
for elemento in objeto :
    instrucción 1
    instrucción 2
        …
    instrucción N
```

PARA cada elemento en objeto ejecutar **instrucciones** dentro del bloque

- En esta estructura iterativa **para** cada **elemento** en **objeto se ejecutarán** una o más **instrucciones**.
- Las **instrucciones** dentro del **for** se **escriben** con un **tabulado mayor** formando un **bloque.**

Función range

- La función **range** en **Python** nos permite establecer un rango o secuencia ordenada de valores.

- Esta **función** puede ser **llamada** de las siguientes 3 formas :

  - **range**(a) → Establece un rango de valores **desde 0 hasta a - 1**

  - **range**(a,b) → Establece un rango de valores **desde a hasta b - 1**

  - **range**(a,b,x) → Establece un rango de valores **entre a y b-1, con un** incremento/decremento **x** positivo/negativo en **cada iteración**.

Ej : Imprimir Números Naturales 1..N

Escriba un **programa** que muestre por pantalla los primeros **N números naturales**. El valor de N será ingresado por el usuario, se asegura que **N >= 1**.

Entrada	Resultado
10	1 2 3 4 5 6 7 8 9 10
1	1

Solución en Python

```python
# se lee el valor de n
n = int(input())

# se itera n veces con la variable i
# i toma valores desde 1 a n
for i in range( 1 , n+1 ) :

    # se escribe el valor actual de i
    # y nos quedamos en la misma línea
    print(i , end = " ")
```

Salida del programa para **n** ingresado igual a 10

```
1 2 3 4 5 6 7 8 9 10
```

Ej #4 : Mostrar enteros entre A y B

ESCRIBA UN PROGRAMA EN PYTHON QUE:

- Muestre por pantalla los números enteros que existen en el rango [a..b].

ENTRADA:

- Valores a y b, se asegura que $a \leq b$.

SALIDA:

- Los números enteros entre a y b.

Entrada	Resultado
10 20	Números enteros en el rango de 10 a 20 : 10 11 12 13 14 15 16 17 18 19 20

Ej#5 : Mostrar Potencias de 2 desde 0 a N

ESCRIBA UN PROGRAMA EN PYTHON QUE: Muestre por pantalla las potencias de 2 desde la 0-ésima hasta la n-ésima.

ENTRADA: Un número entero n . Se asegura que N≥0.

SALIDA: Las potencias de 2 desde 0 a N.

Entrada	Resultado
5	Potencias de 2 de 0 a 5 : 1 2 4 8 16 32