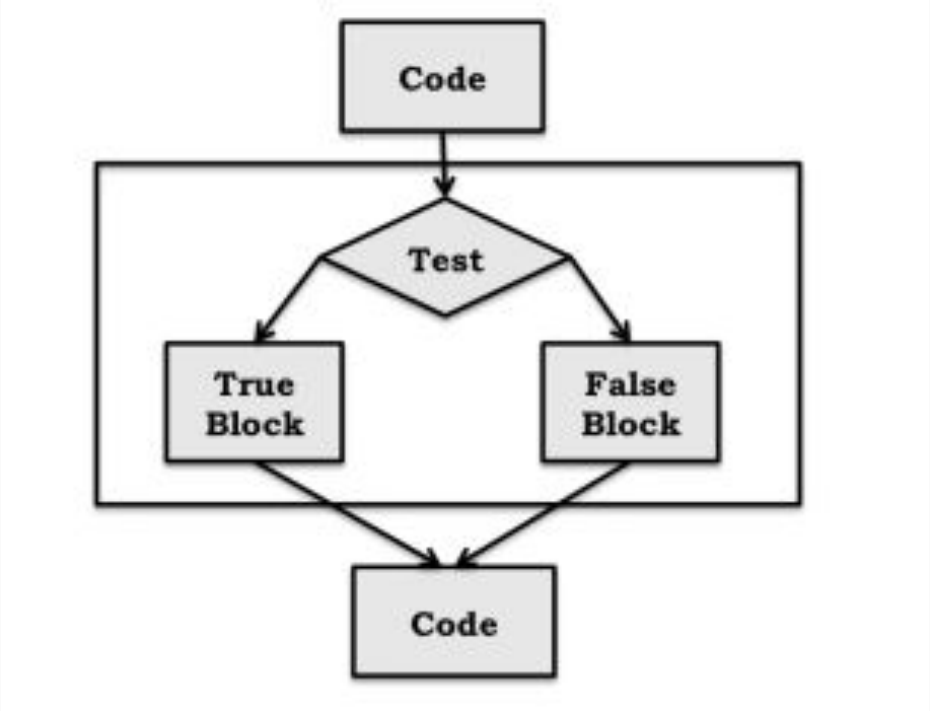
Algoritmos y Programación I
Estructuras de Control

Docente: Uriel Kelman
1er cuatrimestre de 2019

Repaso: estructuras selectivas

- Las estructuras selectivas nos permiten ejecutar un programa interviniendo sobre el flujo del programa.
- Evaluando una expresión condicional, podemos lograr que se ejecute un bloque de código u otro.
- Una expresión es una porción de código Python que produce o calcula un valor (resultado).
- El código se encuentra dividido en tres secciones:
    - La evaluación de la expresión.
    - Un bloque de código que se ejecuta si la expresión es evaluada como verdadera.
    - Un bloque opcional de código que se ejecuta si la expresión es evaluada como falsa.

Este flujo puede ser representado en un diagrama



¿Cómo se traduce esto en Python?

```
if (expresión booleana):
    //bloque de código
else:
    //bloque de código
```

O también puede ser:

```
if (expresión booleana):
    //bloque de código
elif(otra expresión):
    //bloque de código
else:
    //bloque de código
```

Por ejemplo, si quisiéramos evaluar si un número es par o no:

```
if x%2 == 0:
    print(“El numero es par”)
else:
    print(“El numero es impar”)
```

Notar que todos los bloques de código dentro de los condicionales están INDENTADOS.

La indentación es la forma que tenemos en Python de delimitar bloques de código.

Estructuras iterativas

- Como su nombre lo indica, sirven para iterar. ¿Qué significa iterar?
- Iterar significa repetir. Las estructuras iterativas nos van a permitir repetir la ejecución de cierta porción de código que queramos iterar.
- ¿Por qué querríamos repetir la ejecución de una porción de código?
- Ejemplo: Carga de datos a un sistema.

Estructuras iterativas: while

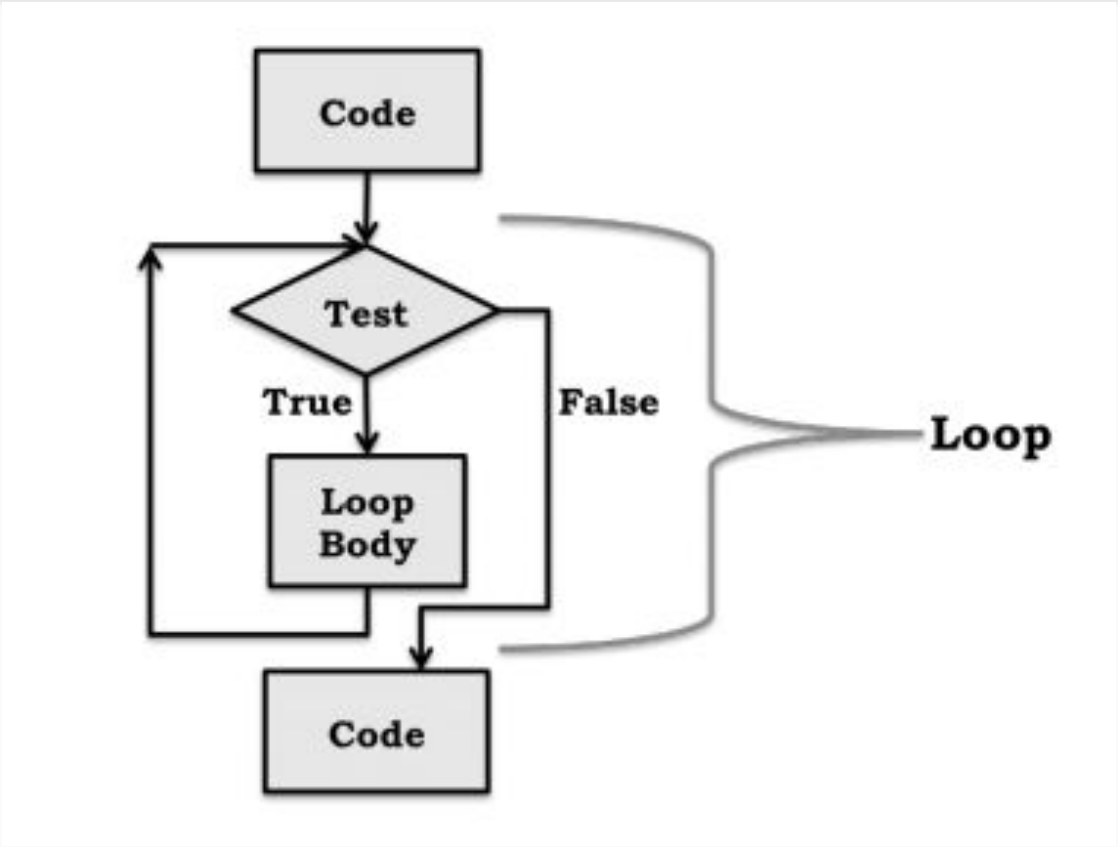
- Permite repetir una porción de código mientras se cumpla una condición.
- Al igual que para los condicionales, su ejecución comienza con la evaluación de una expresión booleana.
- ¿Cuál es la diferencia? Ahora, en vez de tomar un camino u otro, vamos a repetir una porción de código hasta que cambie el valor de la condición.
- Al cambiar el valor de la condición booleana a falso, se continúa con el flujo del programa secuencial normalmente.

while: un primer ejemplo

```
seguir_ingresando = ‘s’

while seguir_ingresando == ‘s’:
        numero = int(input(“Ingrese un numero”))

        if numero > 0:
                print(“El numero ingresado es positivo”)
        elif numero == 0:
                print(“El numero ingresado es certo”)
        else:
                print(“El numero ingresado es negativo”)

        seguir_ingresando = input(“Desea seguir ingresando numeros? s/n”)
```

- Notar que el código a ejecutar en el loop está indentado dentro del while.
- Se solicitan números hasta que el usuario del programa decide terminar con el ingreso.

Este flujo también puede ser representado en un diagrama



Ciclos definidos vs ciclos indefinidos

- Hasta ahora, trabajamos el ciclo while que nos permite repetir un ciclo indefinidamente.
- ¿Qué quiere decir esto? Que el bloque de código dentro del ciclo se puede repetir cualquier cantidad de veces mientras la condición evaluada siga siendo verdadera.
- Este tipo de ciclos, en los que no sabemos cuando va a terminar la iteración, se denominan CICLOS INDEFINIDOS.

- ¿Nos alcanza por esto? ¿No podrían darse situaciones en las que sí se sepa cuando tiene que durar la iteración?
- Lógicamente, casos así podrían darse.
- La solución es utilizar CICLOS DEFINIDOS
- Los ciclos definidos tienen un comienzo y un fin definido, ya que sabemos sobre qué vamos a iterar:
  - un rango de números.
  - una estructura de datos (más adelante).

for: primeros ejemplos de un ciclo definido en Python

```
for x in range(0, 10):
    print(x)
```

```
>> 0
>> 1
>> 2
>> 3
>> 4
>> 5
>> 6
>> 7
>> 8
>> 9
```

¿Qué significa range(0, 10)?

- La función range() recibe valores enteros y devuelve una variable de tipo rango (una secuencia de números sobre la cual nos permite iterar.
- Recibe tres parámetros: range(inicio, fin, salto)
    - inicio: es el número de inicio de la secuencia. (el valor por defecto es 0).
    - fin: es el número donde finaliza la secuencia.

ATENCIÓN: El intervalo excluye al último número.

    - salto: indica el salto a tomar (el valor por defecto es 1).

Algunas instrucciones especiales que podemos utilizar dentro de un loop

- continue: Permite saltar a la siguiente iteración, omitiendo el bloque de código que resta ejecutar de la iteración que se está ejecutando.

- break: Permite terminar con la ejecución del ciclo.

El uso de estas instrucciones debe ser excepcional y justificado. Al tener un break dentro de un ciclo, probablemente no estemos contemplando alguna condición. También, si tenemos un continue, probablemente podríamos escribir el bloque de código de forma tal que el continue no estuviera presente.