Flujo de control

CI-0202 Principios de informática

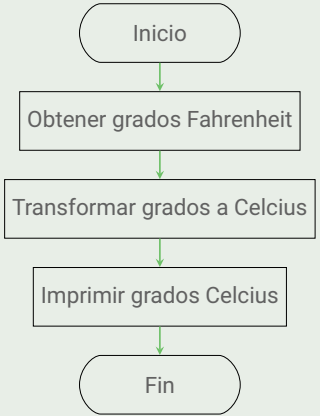
Sivana Hamer - sivana.hamer@ucr.ac.cr
Escuela de Ciencias de la Computación e Informática, Universidad de Costa Rica

UNIVERSIDAD DE COSTA RICA

Flujo de control

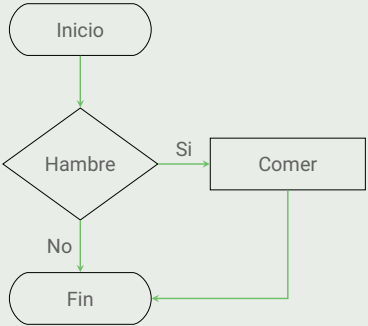
Progrmas anteriores

Secuencia **lineal**.



Flujo control

Cambiar la secuencia.



Estructuras de secuenciación (1)

Recordatorio: Un **bloque de instrucciones** son todas las instrucciones dentro del mismo nivel de indentación.

Ejemplo en Python

```
bloque general
try:
    bloque del try, linea 1
    linea 2
except:
    bloque del except, linea 1
    linea 2
bloque general
```

Estructuras de secuenciación (2)

En otros lenguajes de programación, se utilizan **estructuras de secuenciación** para demarcar los bloques. Usualmente, es con llaves ({ }). Python utiliza el **nivel de indentación** para demarcar los bloques.

Ejemplo de estructuras de secuenciación

```
bloque general
try
{
    bloque del try, linea 1
    linea 2
}
except
{
    bloque del except, linea 1
    linea 2
}
bloque general
```

Estructuras de secuenciación (3)

Se vuelve más visualmente distinctivo cuando hay más **niveles**.

Ejemplo de estructuras de secuenciación

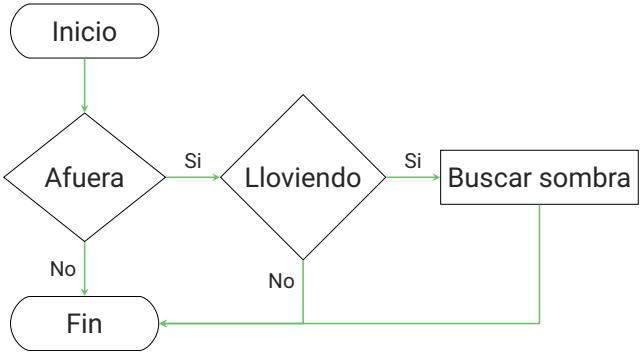
```
bloque general
try{
    bloque del try, linea 1
    linea 2
    try{
        bloque del try try, linea 1
        linea 2
    }
    except{
        bloque del try except, linea 1
    }
}
except{
    bloque del except, linea 1
    linea 2
}
```

Ejemplo en Python

```
bloque general
try:
    bloque del try, linea 1
    linea 2
    try:
        bloque del try try, linea 1
        linea 2
    except:
        bloque del try except, linea 1
        linea 2
except:
    bloque del except, linea 1
    linea 2
bloque general
```

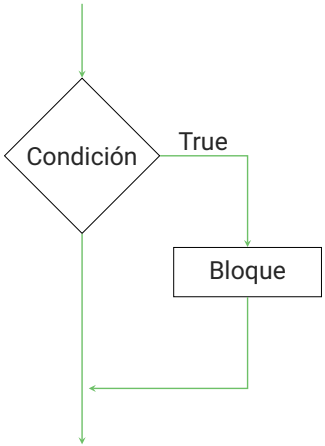
Estructuras de selección

Nos permite definir bloques de código podrían o no ser ejecutados dependiendo una **condición**. Son llamadas también estructuras de **bifurcación**.



If (1)

Estructura de control que ejecuta un **bloque si cumple cierta condición**.



If (2)

Estructura

```
if condicion:
    bloque, linea 1
    bloque, linea 2
linea luego del if 1
linea luego del if 2
```

Nota

Los **operadores relacionales** son tipicamente usados para las **condiciones** del if.

If (3)

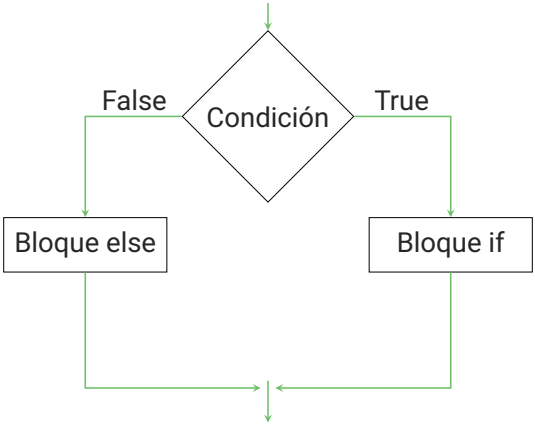
Ejemplo en Python

```
from math import sqrt

numero = float(input("Ingrese un numero:"))

if(numero >= 0):
    print(f"La raiz cuadrada de {numero:.4f} es {sqrt(numero):.4f} ")
```

```
Ingrese un numero: 4
La raiz cuadrada de 4.0000 es 2.0000
```

Else (1)

Estructura de control que ejecuta un **bloque si cumple cierta condición (if)** y **otro bloque si la condición es falsa (else).**



Else (2)

Estructura

```
if condicion:
    bloque if, linea 1
    bloque if, linea 2
else:
    bloque else, linea 1
    bloque else, linea 2
linea luego del if 1
linea luego del if 2
```

Else (3)

Ejemplo en Python

```
from math import sqrt

numero = float(input("Ingrese un numero:"))

if(numero >= 0):
    print(f"La raiz cuadrada de {numero:.4f} es {sqrt(numero):.4f} ")
else:
    print(f"El numero {numero:.4f} no es valido ya que es negativo")
```

```
Ingrese un numero: -100
El numero -100.0000 no es valido ya que es negativo
```

Elif

Combina un else y un if.

Ejemplo sin elif

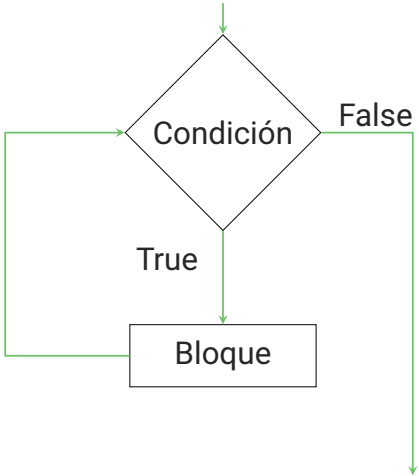
```
#Imprime la palabra de 1, 2, o 3.
from math import log

num = int(input("Ingrese un numero:"))

if num <= 3 and num >=1:
    if num == 1:
        print("Uno")
    else
        if num == 2:
            print("Dos")
        else:
            print("Tres")
```

Ejemplo con elif

```
#Imprime la palabra de 1, 2, o 3.
from math import log

num = int(input("Ingrese un numero:"))

if num <= 3 and num >=1:
    if num == 1:
        print("Uno")
    elif num == 2:
        print("Dos")
    else:
        print("Tres")
```

Estructuras de repetición

Nos permite **repetir** instrucciones.



While (1)

Repite un bloque de instrucciones **mientras una condición es verdadera**.

Estructura

```
while condicion:
    bloque while, linea 1
    bloque while, linea 2
bloque general, linea 1
bloque general, linea 2
```

While (2)

Partes de un ciclo:

- **Inicialización:** Antes del ciclo se crean o asignan las variables.
- **Condición:** Condicion de parada de las iteraciones cuando es falso.
- **Incremento:** Cambio en las variables (usualemente) al final del ciclo.

Además, usualmente una variable está involucrada en todos los pasos. Por lo que se **itera** sobre esta variable.

While (3)

Ejemplo en Python: Contar

```
#Variable de iteracion: numero
numero = 1 # Inicializacion

while(numero <= 10): #Condicion
    print(numero)
    numero += 1 #Incremento
print("Terminamos")
```

Ejemplo en Python: Tablas de multiplicar

```
#Variable de iteracion: i
numero = int(input("Introduzca un numero: "))
i = 0 # Inicializacion
while(i <= 10): #Condicion
    print(f"{numero}x{i} = {numero*i}")
    i = i + 1 #Incremento
```

While (4)

Ejemplo en Python: Entrada

```
#Variable de iteracion: entrada
entrada = "" # Inicializacion

while(entrada != "q" and entrada != "b" and entrada != "p"): #Condicion
    entrada = input("Presione q o b o p para terminar el programa:") #Incremento
```

Nota

La variable de iteración no siempre es un número. Además, el incremento no siempre es una suma.

For (1)

Otra estrucutra que hace **repeticiones**, que hace los 3 pasos implicitos.

Estructura

```
for variable in range(inicio, fin, incremento):
    bloque for, linea 1
    linea 2
    linea 3
bloque general, linea 1
linea 2
```

For (2)

Ejemplo en Python: Contar

```
for numero in range(1, 11, 1): #Aqui se hacen los pasos
    print(numero)
print("Terminamos")
```

Nota

La instrucción for tiene que **conocer la cantidad de iteraciones que va a realizar**, mientras que el while no tiene este requerimiento.

Break

Termina un for o un while **aún cuando la condición de parada no es falso**.

Ejemplo en Python

```
numero = 0
while True:
    if(numero == 5):
        break
    print(numero)
    numero += 1
```

Referencias I

“Flujo de control,” Material del curso CI-0202 Universidad de Costa Rica de Leonardo Villalobos, 2019.

C. Swaroop, *A Byte of Python*. Independent, 2020.

J. Elkner, A. B. Downey, and C. Meyers, “How to think like a computer scientist: Learning with python,” 2012.