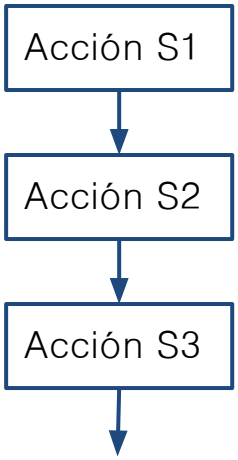
Programación
Licenciatura en Agroinformática
Licenciatura en Bioinformática

Tema 04: Estructuras Iterativas

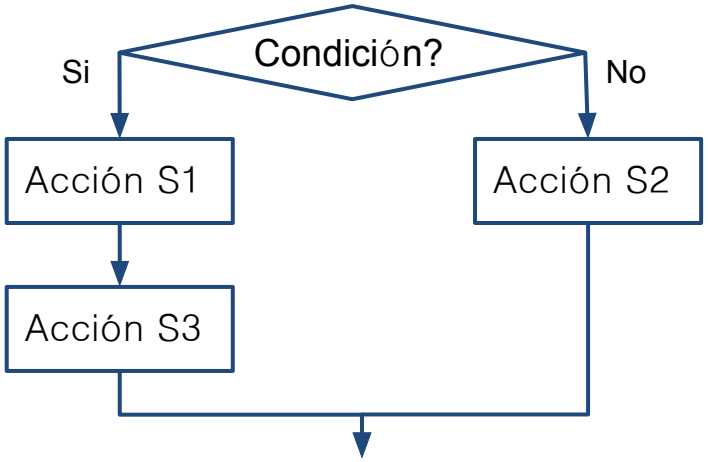
UNRaf UNIVERSIDAD NACIONAL DE RAFAELA

Estructuras de control

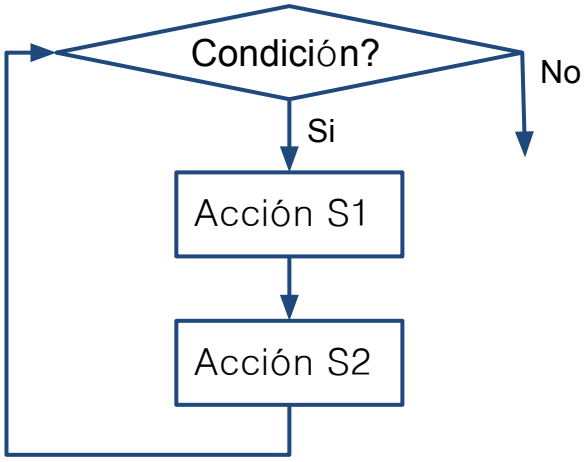
- Cualquier algoritmo se puede construir utilizando combinaciones de tres estructuras de control:



Secuencial



Selectiva



Iterativa

Estructuras Iterativas

- Se denominan **bucles** (o ***loop***, en inglés) a una secuencia de instrucciones que se repite un número determinado de veces.
- Una **iteración** a cada ejecución (repetición) de esa secuencia de instrucciones.
- Todos los bucles se repiten hasta que se cumpla una **condición** determinada.

Estructuras Iterativas

- Existen 2 tipos de estructuras repetitivas en Python:
  - ***while***
  - ***for***
- En los bucles tipo ***while*** se conoce la condición de corte, pero no es necesario saber a priori el número de iteraciones.
- Al momento de declarar un bucle ***for*** es necesario definir el número de iteraciones que se van a necesitar.
- Al igual que en las estructuras selectivas, es necesario **indentar** los bloques de código.

Bucle *while*

```
print("Calculo aproximado del log2")
cant = 0
while x > 0:
    x //= 2
    cant += 1
print("El log2 aproximado de", x, "es", cant)
```

- Se realiza una acción ***mientras*** una condición sea verdadera.
- Dentro del bucle debe modificarse la variable incluida en la condición para poder alcanzar el final del bucle.

Bucles infinitos

- Es imprescindible identificar correctamente la condición de corte de un ***while***.
- Si la condición nunca toma el valor *False*, el bucle y el programa continuarán infinitamente.

```
print("Ejemplo de bucle infinito")
cant = 1
while cant > 0:
    cant += 1
```

Bucle *for*

```
print ("Imprimiendo palabra, letra por letra")
for letra in "programacion":
    print(letra)
```

- Realiza una acción ***por*** cada valor a ser evaluado.
- Todo lo que siga a la palabra reservada ***in*** se considera la lista de valores que se va a evaluar.
- Este grupo de valores no puede modificarse.

Listas de valores

- Un bucle ***for*** solo funciona a partir de una lista prefijada de valores.
- Esos valores pueden ser:
  - Caracteres dentro de un *string*:

```
for letra in "programacion":
```

  - Números en un rango:

```
for numero in range(0,10):
```

  - Elementos dentro de una lista <- siguiente clase!!

Sentencias de salto

- Es posible detener o saltar valores en un bucle.
- La sentencia ***continue*** hace que la iteración no termine y se continúe con la siguiente
- La función ***break*** hace que el bucle se termine
- Se aconseja evitar estas sentencias y definir mejor las condiciones de corte o los rangos de valores a utilizar.

Ejemplo de *continue*

```
for mes in range(1,13):
    for dia in range(1,32):
        if mes == 2 and dia > 28:
            continue
        elif dia == 31 and (mes == 4 or mes == 6 or mes == 9 or mes ==
11):
            continue
        print (dia, "/", mes)
```

Estructuras anidadas

- Al igual que con las estructuras selectivas, es posible anidar bucles entre sí, junto con otras estructuras.
- La indentación va a definir qué bloque de código pertenece a cada estructura.

Ejemplo 1

- Calcular la media de una lista de números ingresados por el usuario. La lista termina cuando el usuario ingresa un número negativo.

- Entrada: lista de números (negativo termina).
- Salida: promedio

```
suma = 0
cantidad = 0
numero_nuevo = 1
mientras numero_nuevo > 0:
    Pedir numero_nuevo
    suma += numero_nuevo
    cantidad += 1
Devolver suma/cantidad
```

Ejemplo 1: código y ejecución

```
suma = 0
cantidad = 0
numero_nuevo = float(input("Ingrese un nuevo numero: " ))
while numero_nuevo > 0:
    suma += numero_nuevo
    cantidad += 1
    numero_nuevo = float(input("Ingrese un nuevo numero: " ))
print("El promedio de los numeros es:" , suma/cantidad)
```

```
(base) matias@rfgenom002:~$ python suma.py
Ingrese un nuevo numero: 2
Ingrese un nuevo numero: 4
Ingrese un nuevo numero: 6
Ingrese un nuevo numero: 8
Ingrese un nuevo numero: 1
Ingrese un nuevo numero: 3
Ingrese un nuevo numero: -5
El promedio de los numeros es: 2.7142857142857144
```

Ejemplo 2

- Obtenga la raíz cuadrada de todos los números entre 0 y un número ingresado por el usuario.

- Entrada: num
- Salida: raiz cuadrada de los números entre 0 y num

```
Pedir num
for i entre 0 y num:
    raiz = i ** 1/2
    Devolver raiz
```

Ejemplo 2: código y ejecución

```
num = float(input("Ingrese un numero: "))
for i in range (0, num+1):
    raiz = i ** (1/2)
    print("La raiz de", i, "es: ", raiz)
```

```
(base) matias@rfgenom002:~$ python raices.py
Ingrese un numero: 6
La raiz de  0 es:  0.0
La raiz de  1 es:  1.0
La raiz de  2 es:  1.4142135623730951
La raiz de  3 es:  1.7320508075688772
La raiz de  4 es:  2.0
La raiz de  5 es:  2.23606797749979
La raiz de  6 es:  2.449489742783178
```

Ejemplo 3

- Contar cuántas vocales tiene una palabra ingresada por un usuario.

- Entrada: palabra
- Salida: número de ‘a’, ‘e’, ‘i’, ‘o’ y ‘u’

```
Pedir palabra
contador = 0
for letra en palabra:
    if letra == ‘a’ o letra == ‘e’ o letra == ‘i’ o letra == ‘o’ o letra == ‘u’
        contador += 1
Devolver contador
```

Ejemplo 3: código y ejecución

```
palabra = input("Ingrese una palabra: ")
contador = 0
for l in palabra:
    if l == "a" or l == "e" or l == "i" or l == "o" or l == "u":
        contador += 1
print("El numero de vocales es: " ,contador)
```

```
(base) matias@rfgenom002:~$ python vocales.py
Ingrese una palabra: programacion
El numero de vocales es:  5
```