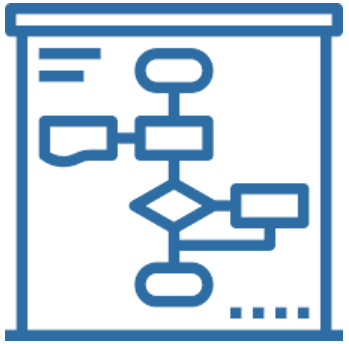
INTRODUCCIÓN A LA PROGRAMACIÓN



OTOÑO, 2022

UNIVERSIDAD TECNOLÓGICA DE CHILE
INSTITUTO PROFESIONAL
CENTRO DE FORMACIÓN TÉCNICA
nacap

UNIDAD II

Lenguajes Altamente Dinámicos/ Débilmente Tipados

Ciclos while, for

RECORDANDO

Conceptos Generales Ciclos de Control

Bucle o ciclo

En informática, la mayoría de las veces las **tareas** que realiza el computador son **repetitivas**, lo único que varía son los valores de los datos con los que se está operando. Se llama bucle o ciclo a todo proceso que **se repite un número de veces** dentro de un programa.

Contador

Un contador es una variable cuyo valor se **incrementa** o **decrementa** en una **cantidad constante** cada vez que se produce un determinado suceso o acción.
Los contadores normalmente se utilizan en las estructuras de repetición con la finalidad de **contar sucesos** o acciones internas de bucle o ciclo, por lo que se inicializan antes y fuera del ciclo.

Ejemplos

cont=cont + 1
a=a –3
final=final-1

Acumulador

Son variables cuyo valor se incrementa o decrementa en una **cantidad variable**. Al igual que los contadores también necesitan inicializarse fuera del ciclo.

Ejemplos

saldo=saldo+depósito
saldo=saldo-retito
*prompar=prompar*num*

Interruptor

Un interruptor o bandera (switch) es una variable que puede tomar los valores 1(verdadero) ó 0 (falso) a lo largo de la ejecución de un programa, dando así información de una parte a otra del mismo. Puede ser utilizado para **control de ciclo**, Ejemplo: sw=1.

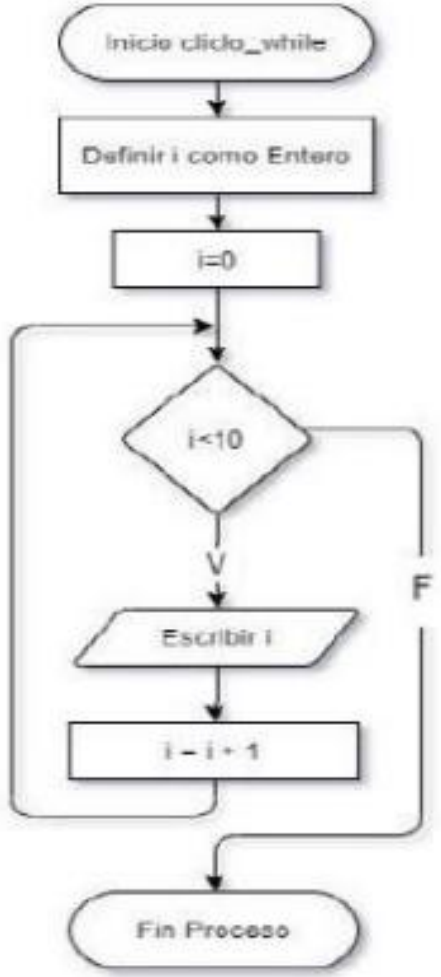
```
while True:
    print("Hola \n")
```

Ciclos

Existen dos tipos de ciclos en Python, los ciclos while y los ciclos for

While:

```
while (condicion):
    #accion/es que se repite/n
```



Aquí al igual que con las condicionales, es fundamental que se respete la indentación, que indicará el bloque de instrucciones dentro del ciclo.

Ciclos: while

Ejemplo, el siguiente Código:

```
c=0;
while (c<5):
        print("Hola" + str(c))
        c+=1
```

Da como resultado:

hola 0
hola 1
hola 2
hola 3
hola 4

```
i = 10
while i !=0:
    print (i)
    i -= 1
```

¿Cuál es la salida por pantalla?

Casos del while

- Con la sentencia **break** podemos detener el ciclo incluso si la condición while es verdadera:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
1
2
3
```

- Con la declaración **continue** podemos detener la iteración actual y continuar con la siguiente:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
1
2
4
5
6
```

Nota: el número 3 falta en el resultado.

- Con la instrucción **else** podemos ejecutar un bloque de código una vez cuando la condición ya no es verdadera:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i no es menor que  6")
```

```
1
2
3
4
5
i no es menor que 6
```

Nota: imprime el mensaje una vez que la condición sea falsa.

Escenario

Escriba un programa en Python con el ciclo While que valide la entrada de un usuario por teclado, consultando: **¿Desea continuar (s/n)?.** Si el usuario ingresa una letra distinta el programa debe imprimir "Ingrese respuesta correcta" y vuelve a solicitar una entrada, hasta que ingresa una "s" o una "n" y sale del programa. Puedes utilizar operadores lógicos de comparación.

Salida propuesta por pantalla

```
¿Desea continuar (s/n)? j
Ingresar respuesta correcta: m
Ingresar respuesta correcta: f
Ingresar respuesta correcta: s
```

```
#modifique el siguiente código
sw = False
while not sw:
    salir = input ("¿Desea continuar? (s/n) ")
    rpta=salir.lower()
    if rpta == "s" :
        sw = True
```

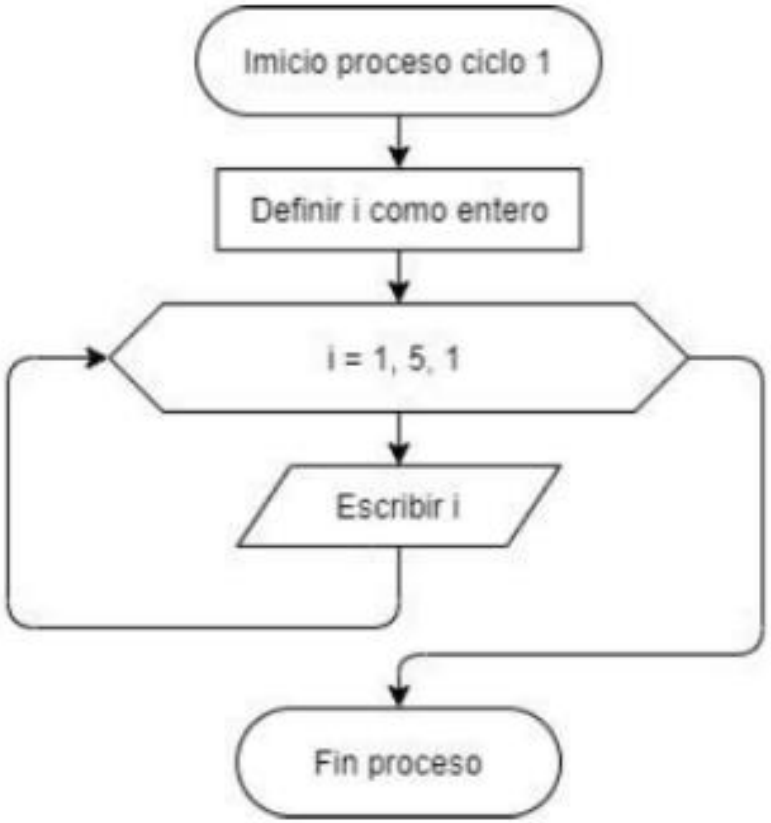
Ciclo for

Esta estructura de ciclo se utiliza para que un trozo de algoritmo se repita un **número específico** de veces que se le indique, cuando se conoce de antemano los límites en que varía una variable (desde hasta).

```
for i in range(a,b):
        comando 1
```

Donde:

- `i` es la variable que incrementa
- `a` es el valor inicial de `i`
- `i` aumenta hasta `b-1`



Ejemplos:

```
for i in range(10, 0, -2):
    print(i)
```

```
for x in range(3):
  print("\n"+"a",end="")
  for j in range(3):
    print("b",end="")
print("\n"+"Hecho"+"\n")
```

Casos del for

- La función range ()

  Para recorrer un conjunto de código un número específico de veces, podemos usar la función range (), devuelve una secuencia de números, comenzando desde 0 de forma predeterminada, se incrementa en 1 (de forma predeterminada) y termina en un número especificado.

```
for x in range(6):
  print(x)
```

```
0
1
2
3
4
5
```

- La función range () tiene como valor predeterminado incrementar la secuencia en 1, sin embargo, es posible especificar el valor de incremento agregando un tercer parámetro: range (2, 30, 3):

```
for x in range(2, 30, 3):
  print(x)
```

```
2
5
8
11
14
17
20
23
26
29
```

- Incluso las cadenas son objetos iterables, contienen una secuencia de caracteres:

```
for x in "banana":
  print(x)
```

```
b
a
n
a
n
a
```

Escenarios

Escriba un programa en Python utilizando el ciclo For, que imprima la siguiente salida por pantalla:

Salida propuesta por pantalla

Escriba un programa en Python utilizando el ciclo For, que dado un número imprima su tabla de multiplicar, de 1 a 10.

Salida propuesta por pantalla

```
Ingrese un número para la tabla de multiplar: 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

inacap.cl