Expresiones relacionales.

Operandos
- Variables
- Constantes
- Expresiones aritméticas

Operadores
- `>` Mayor que
- `>=` Mayor o igual que
- `<` Menor que
- `<=` Menor o igual que
- `==` Igual
- `!=` Diferente

Expresiones lógicas

Operandos
- Expresiones relacionales
- Variables lógicas

Operadores
- not Negación
- and Conjunción
- or Disyunción

Tablas de verdad

P	Q
Verdadero	Verdadero
Verdadero	Falso
Falso	Verdadero
Falso	Falso

P and Q	P or Q
Verdadero	Verdadero
Falso	Verdadero
Falso	Verdadero
Falso	Falso

```python
# -*- coding: utf-8 -*-
"""
Created on Tue May 25 11:28:01 2021

@author: diego Iván Oliveros Acosta
"área de cualquier tipo de triangulo"
"""
import math
a=float(input('Ingrese la longitud del primer lado: '))
b=float(input('Ingrese la longitud del segundo lado: '))
c=float(input('Ingrese la longitud del tercer lado: '))
p=(a+b+b)/2
print('el área del triangulo es:', math.sqrt(p*(p-a)*(p-b)*(p-c)))
```

```
runfile('C:/Users/diego/Desktop/PythonExample/untitled0.py', wdir='C:/Users/diego/Des
ktop/PythonExample')

Ingrese la longitud del primer lado: 10

Ingrese la longitud del segundo lado: 12

Ingrese la longitud del tercer lado: 10
el área del triangulo es: 64.53681120105021
```

CICLOS

FOR, WHILE, DO WHILE

FACULTAD DE INGENIERIA UdeA
Una facultad para la sociedad del aprendizaje

Objetivo

Entender la sintaxis y semántica de los ciclos en Java.

Ejecutar una o varias líneas de código de acuerdo a sus necesidades.

Realizar ejercicios de forma iterativa, incremental o repetitiva.

- Tener control y conocimiento sobre las iteraciones.
- Entender que son las estructuras de control cíclica.
- Entender la sintaxis y semántica de los ciclos en Java.
- Ejecutar una o varias líneas de código de acuerdo a sus necesidades.
- Realizar ejercicios de forma iterativa, incremental o repetitiva.
- Tener control y conocimiento sobre las iteraciones.

El futuro digital es de todos

MinTIC

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Mision TIC 2022

Agenda

- Definición
- Sintaxis
- Patrones
- Ejercicios
  1. **Números pares**
  2. **Múltiples definiciones**
  3. **Cuenta regresiva**
  4. **Contador**
  5. **Conversión grados Fahrenheit a Celsius**
  6. **Promedio de una lista enteros**
- **El juego alto bajo**

Diego Iván Oliveros Acosta

¡Si lo logramos no necesitamos más!

El teorema del programa estructurado, de ***Böhm-Jacopini,*** demuestra que todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:

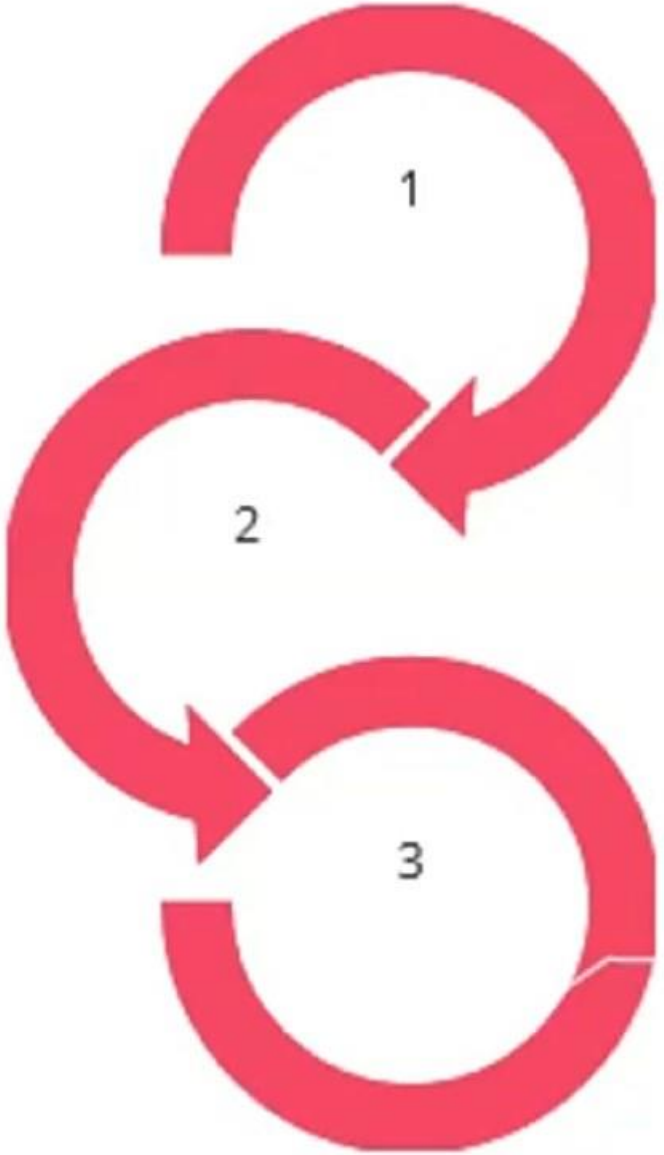
- ***Secuencia***
- ***Instrucción condicional.***
- ***Iteración (bucle de instrucciones) con condición al principio.***

Solamente con estas tres estructuras o "patrones lógicos" se pueden escribir todos los programas y aplicaciones posibles.

Si bien los lenguajes de programación tienen un mayor repertorio de estructuras de control, éstas pueden ser construidas mediante las tres básicas.

Estructuras de control

- Selectivas
- Repetitivas
  - While
  - Do while
  - For



Las estructuras de control repetitivas

Son aquellas en las que una sentencia o grupo de sentencias se repiten muchas veces. Este conjunto de sentencias se llama Bucle, también conocido como lazo o ciclo (Joyanes, 2003).

Un bucle es cualquier construcción de programa que repite una sentencia o secuencia de sentencias un número de veces (Borjas, 2012).

Los ciclos o bucles

- Son una estructura de control
- Permite repetir una o varias instrucciones cuantas veces se necesite.
- Son una estructuras de control cíclica, nos permiten ejecutar una o varias líneas de código de forma iterativa (o repetitiva), pero teniendo cierto control y conocimiento sobre las iteraciones.

- For
- Do-while
- While

@scalapp.co

Después del cuerpo del ciclo

Estructura de control: Do while

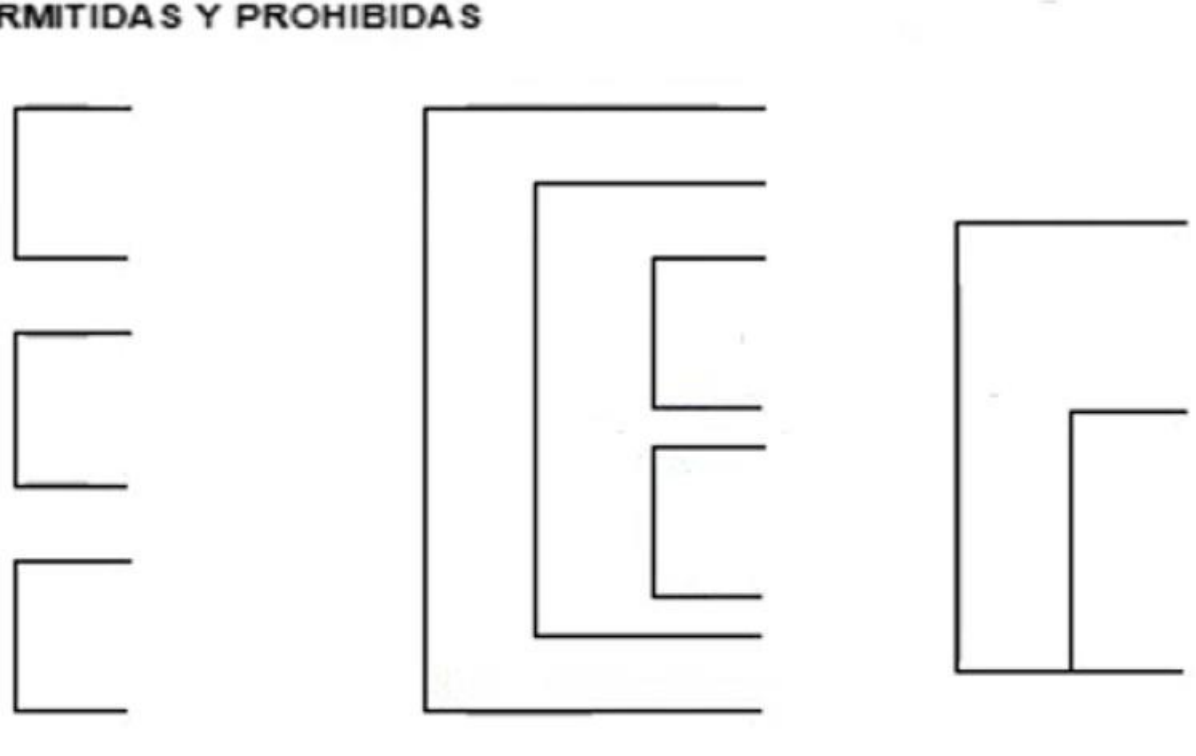
Antes del cuerpo del ciclo

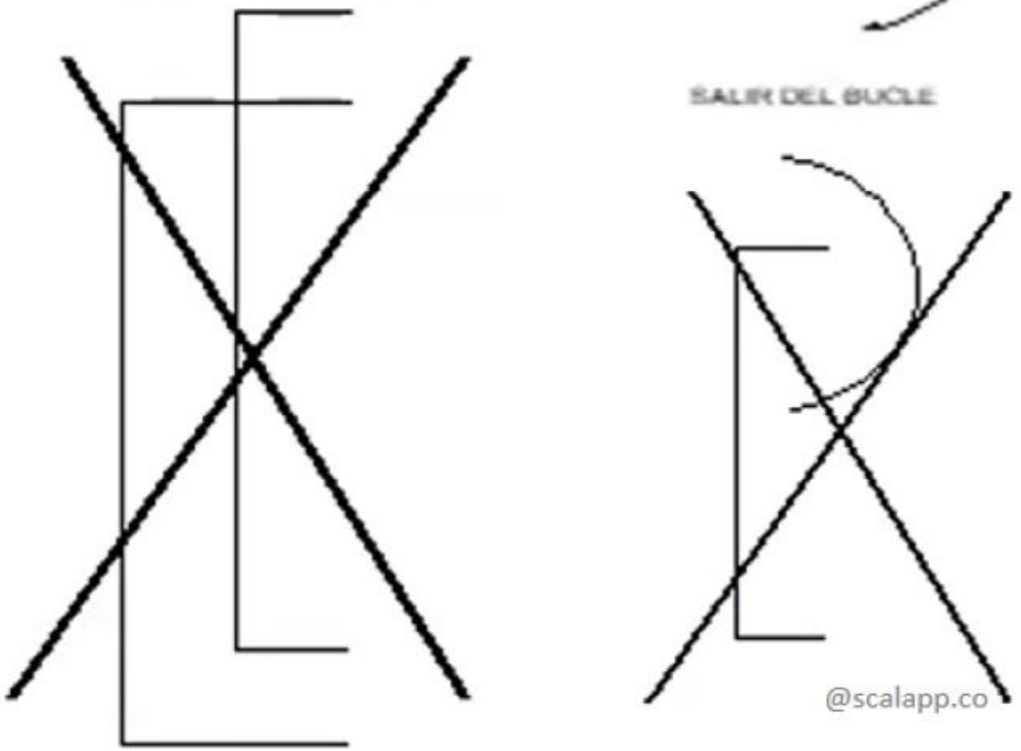
- **Estructura de control: while**
- **Estructura de control: for**

La diferencia radica en que la estructura que controla antes del ciclo no se ejecuta mientras no se cumpla la condición, sin embargo la del control después del ciclo, se ejecuta por lo menos una vez y evalúa, sólo se repite si se cumple la condición de lo contrario, termina el ciclo.

Diseño de bucles. *Bucles Anidados*

PERMITIDAS Y PROHIBIDAS



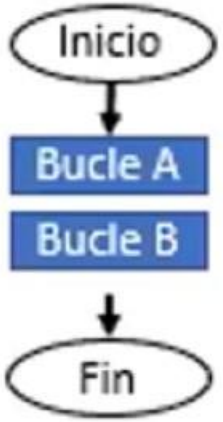


Diseño de estructuras en flujograma:

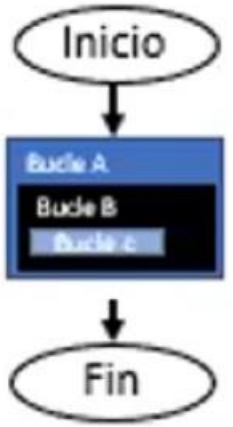
a) Independientes
b) Anidados
c) Cruzados

x) Estructura repetitiva simple
y) Estructura repetitiva anidada
z) Estructura repetitiva compuesta múltiple.

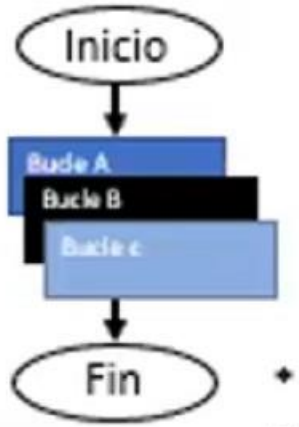
a) Independientes



b) Anidados

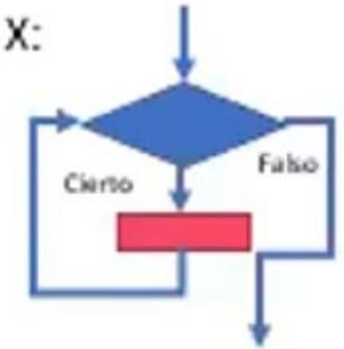


b) Cruzados

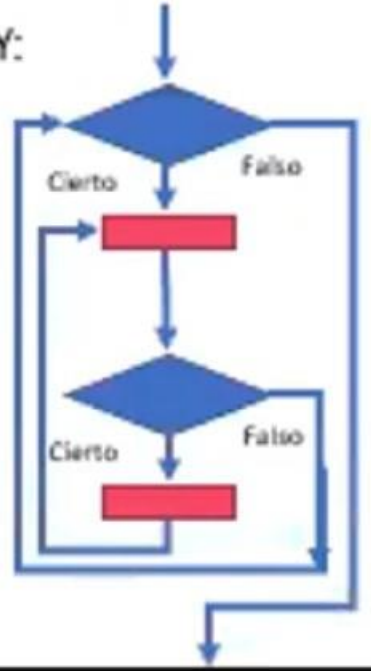


* NO es correcto su diseño

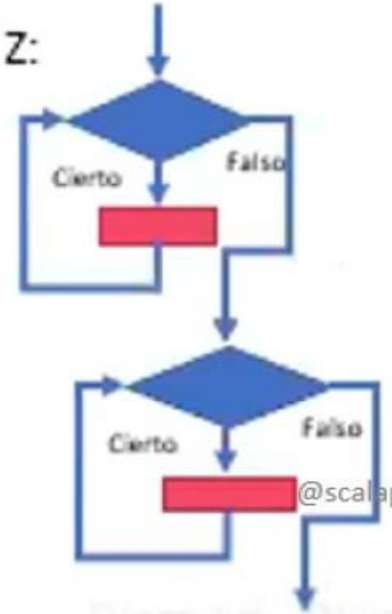
X:



Y:

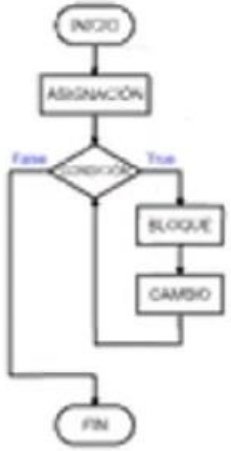


Z:



Ciclo for o ciclos *para*

Estructura, sintaxis y



- Un ciclo for es una **estructura iterativa** para ejecutar un mismo segmento de código una cantidad de veces deseada; **conociendo previamente** un valor de inicio, un tamaño de paso y un valor final para el ciclo.

- **Ejercicio**:
- Queremos mostrar los **números pares** entre el 500 y el 1000.
- Tenemos:
  - ¿un valor inicial?
  - ¿un valor final?
  - ¿ un tamaño de paso?

Sintaxis del Ciclo For en pyhton:

- La sintaxis de **un ciclo for** es simple en python,
- En realidad en la mayoría de los lenguajes de alto nivel es incluso muy similar, de hecho,
- Con tan solo tener bien claros los 3 componentes del ciclo tenemos prácticamente todo hecho.

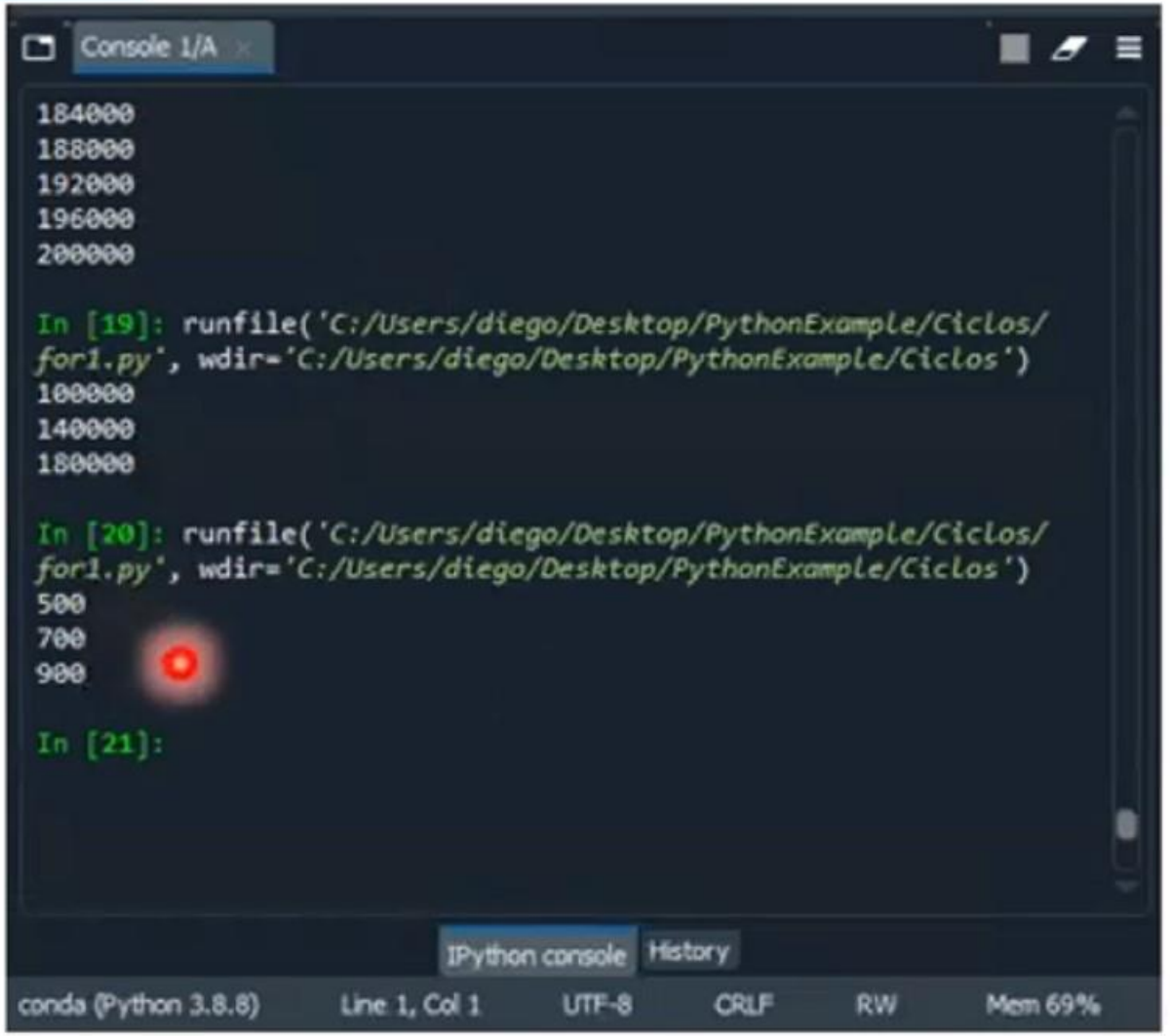
```
for variable in elemento iterable (lista, cadena, range, etc.):
    cuerpo del bucle o Bloque de Instrucciones
```

Ejemplo del Ciclo For en Python:

Ejemplo 1: Mostrar en pantalla los números pares

```
# -*- coding: utf-8 -*-
"""
Created on Tue May 25 11:52:05 2021
@author: diego Iván Oliveros Acosta www.scalapp.co
Vamos a retomar el ejemplo anterior, donde deseábamos sacar los números pares
entre el numero 500 y el 1000, es un ejemplo sencillo con el que nos
aseguraremos de haber comprendido bien lo anterior:
"""

for i in range(500, 1001,200):
    print(i)
```


Console 1/A
184000
188000
192000
196000
200000
In [19]: runfile('C:/Users/diego/Desktop/PythonExample/Ciclos/for1.py', wdir='C:/Users/diego/Desktop/PythonExample/Ciclos')
100000
140000
180000
In [20]: runfile('C:/Users/diego/Desktop/PythonExample/Ciclos/for1.py', wdir='C:/Users/diego/Desktop/PythonExample/Ciclos')
500
700
900
In [21]:
IPython console
History
conda (Python 3.8.8)
Line 1, Col 1
UTF-8
CRLF
RW
Mem 69%

Ejemplo 3: Cuenta regresiva en un ciclo for

- Ahora veremos otro ejemplo sencillo en cual haremos que el ciclo for también haga sus iteraciones en sentido inverso.
- Es decir disminuyendo el valor de la variable de control vamos a imprimir por pantalla una cuenta regresiva desde el número 100 hasta el 0, veamos:

```
for i in range(1000, 500,-100):
    print(i)
```

Ejemplo 4: Contador con un ciclo for

Para este ejemplo haremos algo un poco más complejo. El ejemplo consiste en contar al interior de un ciclo for, cuántos números entre el 0 y el 10.000 son múltiplos del 20. Para ello haremos uso del operador % (módulo) que obtiene el residuo de una división y también usaremos un pequeño condicional para verificar que el módulo sea cero al dividir por 20.

Nota: El operador de módulo (%) obtiene el residuo de una división, por tanto cuando el residuo es cero implica que la division es exacta y el dividendo es un múltiplo del divisor. Por ejemplo 10%3 nos dará el residuto de dividir 10 entre 3, el cual es 1, si calculamos 120%20 nos dará cero, pues 120 es múltiplo de 20 (20 * 6 = 120)

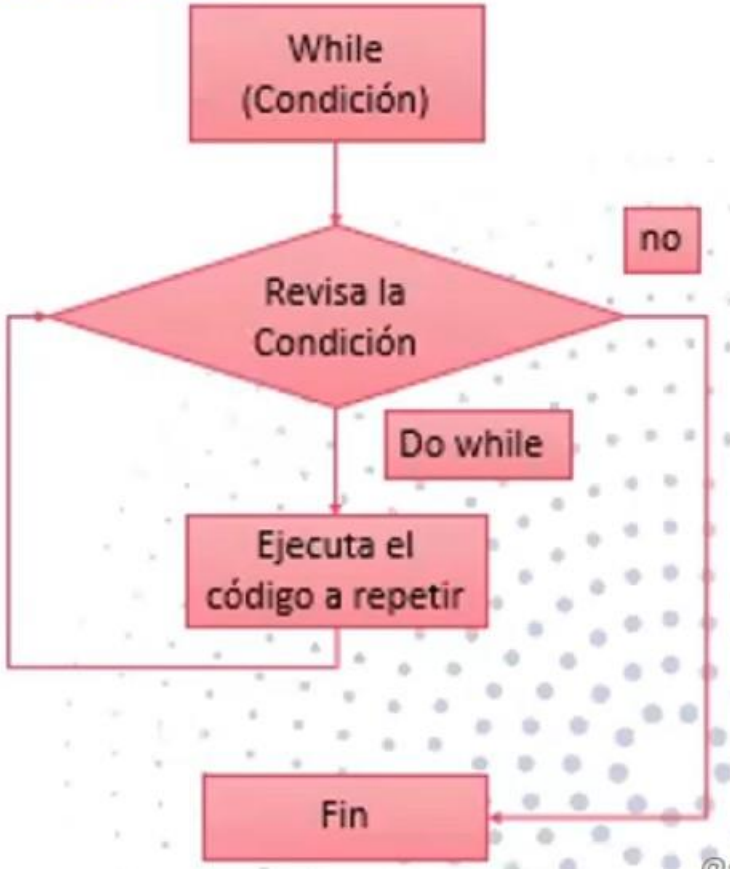
```
contador=0
for i in range(0, 10001):
    if i % 20 ==0:
        contador+=1
print(i)
```

Este ciclo for nos permitirá saber que existen 501 múltiplos del número 20 en los números del 0 al 10000.

El bucle while y el bucle do while

- presenta ciertas similitudes y ciertas diferencias con el bucle for. La repetición en este caso se produce no un número predeterminado de veces, sino mientras se cumpla una condición.
- Conceptualmente el esquema más habitual es el presentado:

- El bucle **do while** es prácticamente igual al **while**, pero con la diferencia de que el código del bucle se ejecutara al menos una vez ya que la comprobación se hace después de cada iteración y no antes como en el caso del while.



El futuro digital es de todos | MinTIC

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Misión TIC 2022

Ciclo while

Console 1/A

```
IndentationError: expected an indented block


In [12]: runfile('C:/Users/diego/Desktop/PythonExample/Ciclos/while.py', wdir='C:/
Users/diego/Desktop/PythonExample/Ciclos')
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

```
a=0
while True:
    print (a)
    if a==2:
        pass
    if (a==15):
        break
    a+=1
```

En este código hemos hecho algo un poco extraño;
Como condición a evaluar hemos puesto "true".

- Esto significa que la condición es siempre verdadera, lo que en teoría daría lugar a un bucle infinito y a un bloqueo del ordenador.
- Sin embargo, utilizamos un contador auxiliar que inicializamos en cero y en cada repetición del bucle aumentamos en una unidad.
- A su vez, introducimos una condición dentro del bucle según la cual cuando el contador alcanza el valor 9 se ejecuta la instrucción break.

```
In [13]:
```


IPython console | History

```
número =int(input("Ingrese el número: "))
cont=0
print(número)
while número>0.0:
    número = número / 10
    cont += 1
    print('el número es: ',  número)
    print('el contador es: ', cont)
    if número <0:
        break
```

- Realicemos un programa que cuente la cantidad de dígitos que posee un número.
- Para ello tendremos que dividir por diez el número que nos han dado, hasta que el resultado se vuelva cero.
- Entonces recurrimos al while para realice los ciclos necesarios.

```
package javaciclo;
public class Javaciclo {
    public static void main(String[] args) {

        int número = 546823;
        int digitos = 0;
        do {
            número /=10;
            digitos++;
        }
        while ( número > 0 );
        System.out.println(digitos);
```

@scalapp.co

es de todos

Facultad de Ingeniería

TIC2022

Grados

```
Console 1/A

      1e+02         37.78

In [64]: runfile('C:/Users/diego/Desktop/PythonExample/Ciclos/
untitled2.py', wdir='C:/Users/diego/Desktop/PythonExample/Ciclos')
    Fahrenheit      Celsius

          0.0         -17.78

         20.0         -6.667

         40.0          4.444

         60.0          15.56

         80.0          26.67

         1e+02         37.78

In [65]:

conda (Python 3.8.8)   Line 18, Col 5   UTF-8   CRLF   RW   Mem 69%
```

```
rangoInicial=0;
rangoFinal=100;
paso=20;
fahrenheit=float(rangoInicial);
print('\t Fahrenheit \t Celsius\n')
celsius=float(0);
while fahrenheit <= rangoFinal:
    celsius=5.*(fahrenheit-32)/9
    print('\t \t {0:.3} \t \t  {1:.4} \n'.format(fahrenheit,celsius))
    fahrenheit +=paso
```


Diego Iván Oliveros Acosta

Calcula promedio de una lista enteros

```
n, cont = 1,0
print('son {} y el otro{}'.format(n,cont))
x,promedio,suma=0.0,0.0,0.0
print('Promedio de números enteros')
n=int(input('¿Cuántos números?'))
#do
while cont<n:
    x=float(input('ingrese el número: '))
    suma+=x
    cont+=1
promedio=suma/n
print('el promedio es {0:3}'.format(promedio))
```

El juego alto bajo

- *Escriba una aplicación que permita jugar Alto-Bajo con el usuario.*
- *Se generarán números secretos aleatorios entre 1 y 100.*
- *Cuando el usuario propone un número, el programa responde Alto o Bajo dependiendo si es mayor o menor que el número secreto.*
- *El número máximo de intentos*

El juego alto bajo