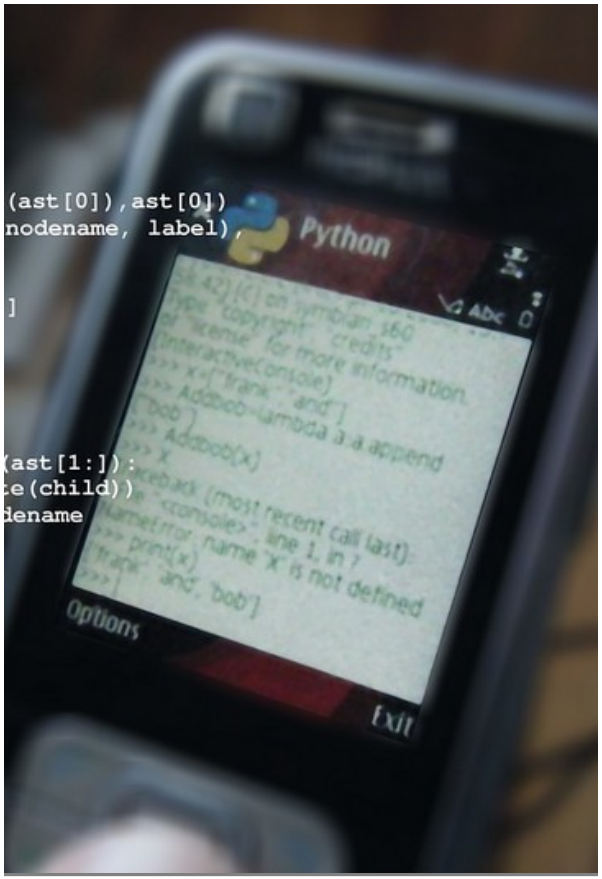
Fundamentos de Programación. Primer Curso de ASIR.

UT03 – Estructuras de Control.

UT03 – Estructuras de Control.
1.- Introducción.



<u>Estructuras de Control. Función</u>.

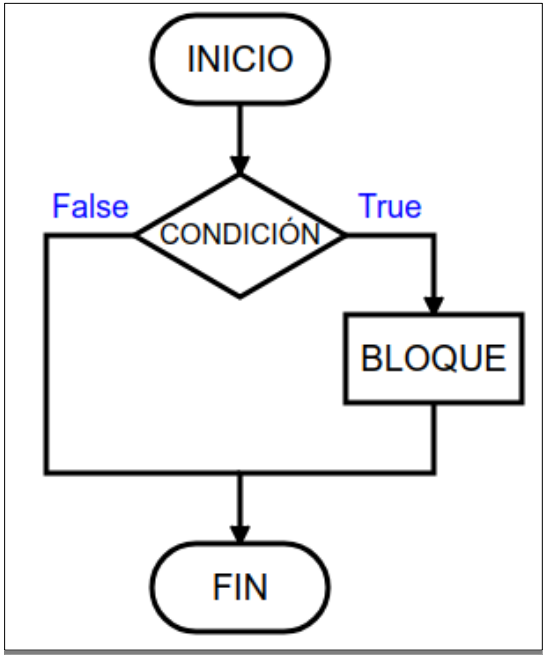
- Las estructuras de control permiten modificar el **flujo de ejecución de las instrucciones de un programa**. Es decir, dependiendo de las condiciones del conjunto de variables se realice la ejecución y una parte de código u otra.
- Con las estructuras de control se puede:
  - Bloque de sentencias a ejecuta en función de una condición.
  - Bloque de sentencias mientras o hasta que se cumpla una condición.
  - Bloque de sentencias a ejecutar un número de veces.

<u>Bloque de instrucciones en Python</u>.

- Un bloque de instrucciones en Python se define como un conjunto **no vacio** de instrucciones que van asociados a una estructura de control.
- El Bloque de sentencias se escribe tabulado en relación a su estructura de control. La sintaxis es la siguiente:

```
Estructura de Control:
    Sentencia1
    Sentencia2
    . . . . .
    SentenciaN
```

UT03 – Estructuras de Control.
2.- Bucle de bifurcación if.
2.1.- Bifurcación simple. La estructura if.



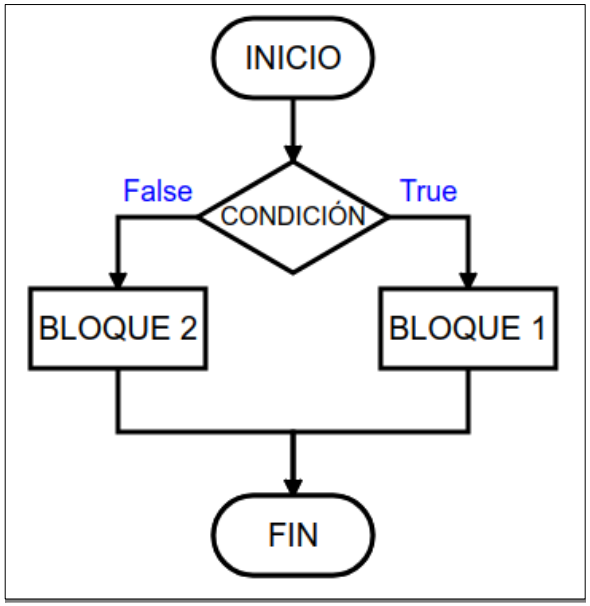
- La estructura de control if **permite que un programa ejecute unas instrucciones cuando se cumplan una condición**.
- La sintaxis es la siguiente:

  if condición:
  Bloque

- **La ejecución de esta construcción es la siguiente:**
  - **La condición se evalúa siempre. Esta línea debe terminar siempre por dos puntos.**
  - **Si el resultado es True se ejecuta el bloque de sentencias.**
  - **Si el resultado es False no se ejecuta el bloque de sentencias.**

```
numero = int(input("Escriba un número positivo: "))
if numero < 0:
    print("El número debe ser positivo.")
print("Ha escrito el número {}".format(numero)
```

UT03 – Estructuras de Control.
2.- Bucle de bifurcación if.
2.1.- Bifurcación simple. La estructura if - else.

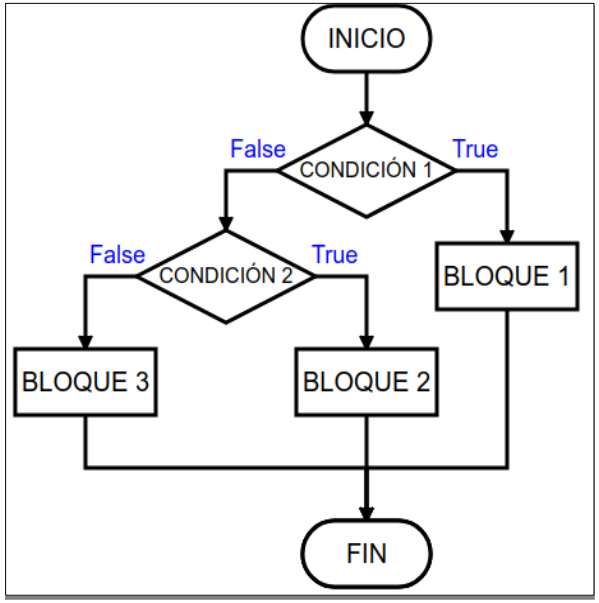


- La estructura de control if - else **permite que un programa ejecute unas instrucciones cuando se cumple una condición y otras instrucciones cuando no se cumple esa condición**.
- La sintaxis es la siguiente:

  if condición:
  Bloque 1
  else:
  Bloque 2
- **La ejecución de esta construcción es la siguiente:**
  - **La condición se evalúa siempre.**
  - **Si el resultado es True se ejecuta solamente el Bloque 1.**
  - **Si el resultado es False se ejecuta solamente el Bloque 2.**

```
edad = int(input("¿Cuántos años tienes? "))
if edad < 18:
    print(“Eres menor de edad.”)
else:
    print("Eres mayor de edad.")
```

UT03 – Estructuras de Control.
3.- Bucle de bifurcación if.
2.2.- Bifurcación múltiple. La estructura if – elif - else.



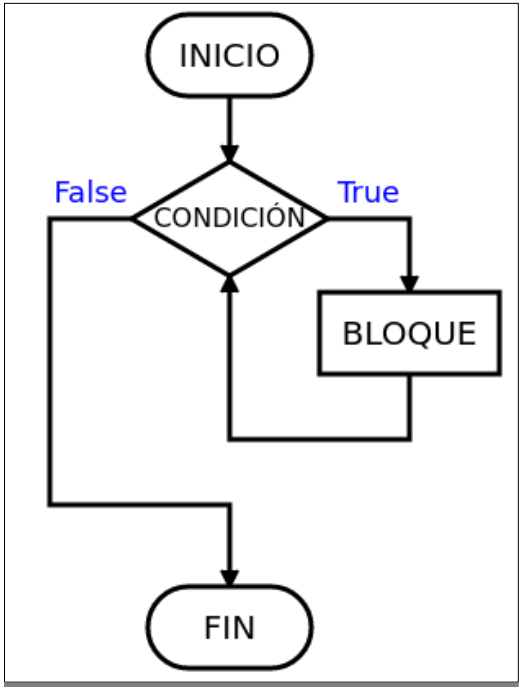
- Una sentencia condicional puede contener a su vez otra sentencia anidada. La estructura de control if - elif - else permite encadenar varias condiciones.
- La sintaxis en la siguiente:

  if (condición 1):
  Bloque 1
  elif (condición 2):
  Bloque 2
  else:
  Bloque 3

- **La ejecución de esta construcción es la siguiente:**
  - **Si se cumple la condición 1, se ejecuta el Bloque 1.**
  - **En caso contrario, si se cumple la condición 2, se ejecuta el Bloque 2.**
  - **Si no se cumplen ninguna de la condiciones, se ejecuta el bloque 3.**

```
numero = int(input("Introduce un número: "))
if (numero == 0 < 18):
    print(“Cero no es par ni impar.”)
elif (numero % 2 == 0):
    print("El numero {} es par.".format(numero))
else:
    print("El numero {} es impar.".format(numero))
```

UT03 – Estructuras de Control.
3.- Bucle While.
3.1.- Sintaxis básica.



Sintaxis.

- Un bucle while ejecuta de un grupo de instrucciones mientras se cumpla una condición.
- La sintaxis es la siguiente:

  while (condición):
  Bloque
- La ejecución de esta estructura de control while es la siguiente:
  - Se evalúa la condición. Si el resultado es **True** se ejecuta el bloque de instrucciones del bucle.
  - Se repite el proceso mientras la condición sea cierta.
  - Si el resultado es **False**, el bloque del bucle no se ejecuta, continuando con la ejecución del programa.
- **El número de iteraciones se desconoce al empezar el bucle**.

Variable de control.

- **Variables de Control**. Aparezcan en la condición se suelen llamar variables de control.
- Para ejecutar el bucle se comprueba si estas variables cumplen una condición.
- **Deben definirse antes de comenzar el bucle y modificarse en el bloque de instrucciones**.

UT03 – Estructuras de Control.
3.- Bucle While.
3.2.- Ejemplos.

Ejemplo. Desarrollar un programa que imprima los número del uno al cinco.

```
# Variable de control
Numero = 1
# Mientras que la variable de control sea menor o
# igual que cinco se ejecutara el bucle.
while (numero <= 5):
     print(numero)
     # Debemos cambiar el valor de la variable
     # de control
     numero = numero + 1
print("Programa terminado")
```

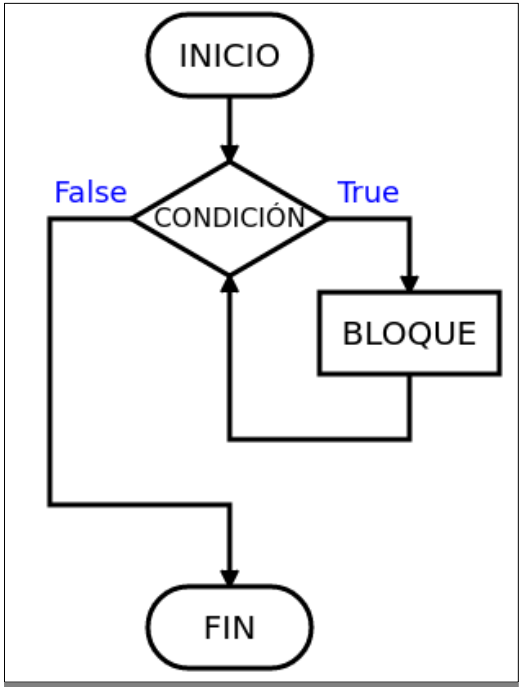
```
1
2
3
4
5
Programa terminado.
```

Ejemplo. Desarrollar un programa que solicite un número hasta que se introduzca un cero. Para cada uno de esos números, se debe indicar su es par o impar.

```
numero = int(input("Introduzca un numero: "))
while (numero != 0):
     if (numero%2 == 0):
          print("El {} es par.".format(numero))
     else:
          print("El {} es impar.".format(numero))
     numero = int(input("Introduzca un numero: "))
print("Fin de la ejecución del programa.")
```

```
Introduzca un numero: 4
El 4 es par.
Introduzca un numero: 6
El 4 es par.
Introduzca un numero: 9
El 4 es impar.
Introduzca un numero: 0
Fin de la ejecución del programa.
```

UT03 – Estructuras de Control.
3.- Bucle While.
3.3.- Bucles infinitos (I). Casos mas frecuentes.



- La condición del bucle se cumple siempre → el bucle no termina nunca.
- A veces es necesario utilizar bucles infinitos en un programa pero, normalmente se deben a errores que se deben corregir.
- Los bucles infinitos no intencionados hacen perder el control del programa.
- Motivos por lo que se crean involuntariamente bucles infinitos:
  - **No se modifica la variable de control dentro del bucle**.
  - **La condición se cumple siempre ya que no se manejado bien la condición de la variable del bucle while.**
  - **La condición que se cumple siempre ya que no se tenido en cuenta que puede que no se deje cumplir la condición.**

UT03 – Estructuras de Control.
3.- Bucle While.
3.3.- Bucles infinitos (II). Ejemplos.

La variable no se cambia dentro del bloque.

```
indice = 1
while (indice <= 10):
     print(indice)
```

```
1
1
1
1
1
. . .
```

No se definido bien la condición del bucle while.

```
indice = 1
while (indice > 0):
     print(indice)
     indice = indice + 1
```

```
1
2
3
4
. . .
```

No se definido bien la variable de control.

```
indice = 1
while (indice != 4):
     print(indice)
     indice = indice + 2
```

```
1
3
5
7
. . .
```

UT03 – Estructuras de Control.
4.- Bucle For.
4.1.- Sintaxis básica.

<u>Sintaxis</u>.

- Un bucle for es un bucle que repite el bloque de instrucciones **un número prederminado de veces**.
- La sintaxis de un bucle for es la siguiente:

  for variable in iterable:
  cuerpo del bucle

- **No es necesario definir la variable de control antes del bucle, aunque se puede utilizar como variable de control una variable ya definida en el programa**.
- El cuerpo del bucle se ejecuta tantas veces como elementos tenga el elemento iterable.

UT03 – Estructuras de Control.
4.- Bucle For.
4.2.- Elemento iterables. Listas.

Una lista es un conjunto de elementos separados por comas y contenidos entre corchetes.

```
for i in [0, 1, 2]:
     print(i)
```

```
0
1
2
```

No se definido bien la condición del bucle while.

```
for i in [2, 3, 5]:
     res = i ** 2
     print("{} al cuadrado es {}.".format(i, res))
```

```
2 al cuadrado es 4.
3 al cuadrado es 9.
5 al cuadrado es 25.
```

UT03 – Estructuras de Control.
4.- Bucle For.
4.3.- Elemento iterables. Rangos.

- **El tipo range es una lista de números enteros en sucesión aritmética**.
- Un rango se crea llamando al tipo de datos con **uno, dos o tres argumentos numéricos**.
  - **range(n)**. Crea una lista de n números enteros consecutivos que empieza en 0 y acaba en n – 1.
  - **range(m, n)**. Crea una lista inmutable de enteros consecutivos que empieza en m y acaba en n – 1.
  - **range(m, n, p)**. Crea una lista inmutable de enteros que empieza en m y acaba justo antes de superar o igualar a n sumando p al anterior. Si p es negativo, los valores se obtienen restando p al anterior.

```
for i in range(3):
     print(i)
```

```
0
1
2
```

```
for i in range(2,7):
     print(i)
```

```
2
3
4
5
6
```

```
for i in range(3,10,2):
     print(i)
```

```
3
5
7
9
```

UT03 – Estructuras de Control.
4.- Bucle For.
4.4.- Forma alternativa.

- Python soporta una forma alternativa muy semejante a otros lenguajes de programación.
- La sintaxis es la siguiente:

for (inicio; condición; salto):
cuerpo del bucle

```
for (x=0; x<5; x++):
     print (i)
```

```
0
1
2
3
4
```

UT03 – Estructuras de Control.
5.- Variables ligadas a los bucles.
5.1.- Testigos.

- Se entiende por testigo una **variable que indica simplemente si una condición se ha cumplido o no**.
- Se suele hacer con variables lógicas.

```
encontrado = False
for i in range(1, 6):
     if i % 2 == 0:
          encontrado = True
if encontrado:
     print(“Hay al menos un par en el rango.")
else:
     print(”No hay ningún par en el rango.")
```

```
Hay al menos un par en el rango.
```

```
encontrado = False
numero = 1
while (numero<6 and encontrado = False):
     if numero % 2 == 0:
          encontrado = True
     numero = numero + 1
if encontrado:
     print(“Hay al menos un par en el rango.")
else:
     print(”No hay ningún par en el rango.")
```

```
Hay al menos un par en el rango.
```

UT03 – Estructuras de Control.
5.- Variables ligadas a los bucles.
5.2.- Contadores.

- Es una variable que lleva la **cuenta del número de veces que se ha cumplido una condición**.

```
contador = 0
for i in range(2, 10, 3):
     if i % 2 == 0:
          contador = contador + 1
print(”Hay {} pares en el rango.".format(contador)
```

```
Hay 2 pares en el rango.
```

```
# Este ejemplo cuenta el producto de 3 números
# introducidos por teclado.
numMultiplosDos = 0
numMultiplosTRes = 0
for indice in range(1,10):
     if (indice % 2 == 0):
          numMultiplosDos = numMultiplosDos + 1
     if (indice % 3 == 0):
          numMultiplos = numMultiplosDos + 1
print(“Hay {} multiplos de dos.”.format(numMultiplosDos))
print(“Hay {} multiplos de tres.”.format(numMultiplosTres))
```

```
Hay 3 multiplos de dos.
Hay 3 multiplos de tres.
```

UT03 – Estructuras de Control.
5.- Variables ligadas a los bucles.
5.3.- Acumuladores.

- Un acumulador es una variable que acumula el resultado de una operación.
- Dependiendo de la operación el acumulador se debe inicializar a un valor en especial.
  - En el caso de sumas el acumulador se deberá inicializar a cero.
  - En el caso de productos el acumulador se deberá inicializar a uno.

```
# Este ejemplo cuenta el producto de 3 números
# introducidos por teclado.
resultado = 1
for indice in range(10):
     numero = int(input(“Numero: “))
     resultado = resultado * numero
print(“El producto es {}.”.format(resultado))
```

```
Numero: 2
Numero: 3
Numero: 10

El producto es 60.
```