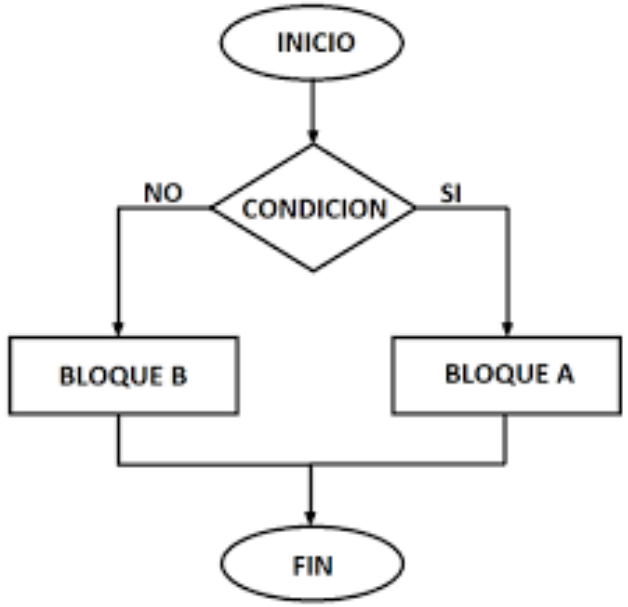
Control de flujo

El control de flujo es la forma en que un programa ejecuta una serie de instrucciones en un orden específico. El control de flujo permite a un programa ejecutar una tarea o un conjunto de tareas dependiendo de la lógica que se incorpora en el programa.



Sentencia **If**

- La sentencia **if** toma una decisión sobre la base de una condición o expresión dada.
- El bloque de código dentro de la instrucción **if** es ejecutado cuando dicha condición sea True.
- La sentencia **elif** es ejecutada solamente si la expresión if precedente y cualquiera de las expresiones **elif** precedentes son evaluadas como false. Puede haber cero o más bloques **elif**.
- La sentencia **else** es ejecutada solamente si la expresión if precedente y todas las expresiones **elif** precedentes son evaluadas también como false. E l bloque es opcional.

Sintaxis:

```
if condición:
    código
elif condición_dos:
    código
else:
    código
```

Nota: Las setencias elif y else son opcioales

Ing. Antonio Torres Sarmiento

Sentencia **for**

- La sentencia **for** de Python itera sobre los ítems de cualquier secuencia (una lista o una cadena de texto), en el orden que aparecen en la secuencia.

Sintaxis:

```
for elemento in lista_de_elementos:
    código
```

Sentencia **while**

- La sentencia **while** de Python se usa para la ejecución repetida siempre que una expresión sea verdadera.

Sintaxis:

while condición:
código

Ing. Antonio Torres Sarmiento

Función **range()**

- Para iterar sobre una secuencia de números, es apropiado utilizar la función integrada **range()**, la cual genera progresiones aritméticas.
- El valor final dado nunca es parte de la secuencia.
- Es posible hacer que el rango empiece con otro número, o especificar un incremento diferente (incluso negativo; algunas veces se lo llama “paso”).

Sintaxis:

```
for  elemento in range( valor ):
    código
```

Nota: El valor ingresado en la función range debe ser de tipo entero.

Ing. Antonio Torres Sarmiento

Sentencias **brack, continue y else**

- La sentencia **break**, termina el bucle for o while.
- Las sentencias de bucle pueden tener una cláusula **else** que es ejecutada cuando el bucle termina, después de agotar el iterable (con for) o cuando la condición se hace falsa (con while), pero no cuando el bucle se termina con la sentencia break.
- En una sentencia try la cláusula **else** se ejecuta cuando no se genera ninguna excepción.
- La declaración **continue**, continua con la siguiente iteración del ciclo.

ejemplo:

```
for  condición:
    código
else:
    código
```

```
while condición:
    código
else:
    código
```

```
try:
    código
except:
    código
else:
    código
```

Ing. Antonio Torres Sarmiento

Sentencia **pass**

- La sentencia **pass** no hace nada. Se puede usar cuando una sentencia es requerida por la sintaxis pero el programa no requiere ninguna acción.

ejemplo:

```
class NombreClase:
    pass

def nombre_función():
    pass
```

Sentencia **match**

- Recibe una expresión y compara su valor con patrones sucesivos dados en uno o más bloques case.
- Sólo se ejecuta el primer patrón que coincide.
- Es capaz de extraer componentes (elementos de una secuencia o atributos de un objeto) de un valor y ponerlos en variables.

Sintaxis:

```
match valor_a_comparar:
    case  valor:
        código
    case  valor_dos:
        código
    case _:
        código:
```

Nota: _ funciona como un *comodín* y nunca fracasa la coincidencia. Si ninguno de los casos **case** coincide, ninguna de las ramas es ejecutada.