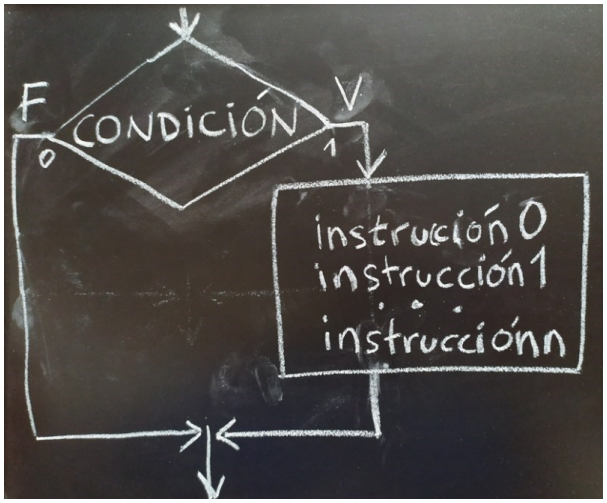
Estructuras de Selección

Cátedra de Fundamentos de Programación

Elaborado por José Colbes (jcolbes@fiuna.edu.py)

¿Qué veremos hoy?

- Elementos de la programación estructurada.
- Estructuras condicionales o de selección:
  - Necesidad de estructuras de selección.
  - Representación.
  - Operadores lógicos y relacionales.
  - Expresiones booleanas.
  - Estructuras de selección simples, alternativas y anidadas.



Estructura básica de un *script* en Python

```
‘‘‘
Comentario - Descripcion
del Programa
’’’

<linea de código 1>
<linea de código 2> #comentario
…
<linea de código n>
```

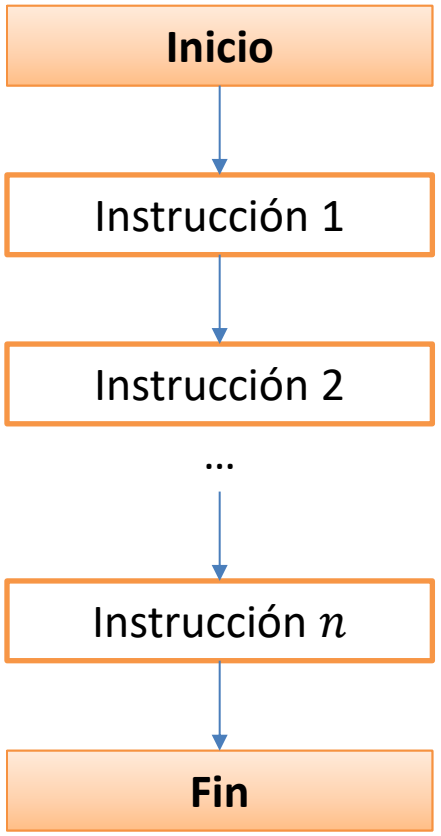
Cuerpo del programa
Instrucciones del lenguaje Python

Es bastante fácil/directo en comparación a otros lenguajes

Algoritmos lineales

Características:

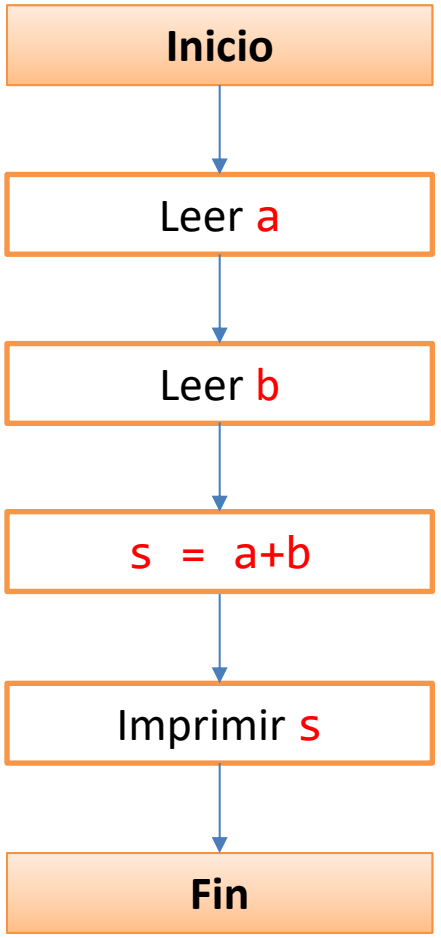
- Único flujo de ejecución.
- Se especifican las cómputos o instrucciones uno tras otro de forma secuencial.
- Por sí solos, resuelven problemas muy simples y cortos.
- Son usados en conjunción en otros algoritmos con procesos selectivos e iterativos.



Algoritmos lineales - Ejemplo

```
'''
Programa que lee dos números por
teclado y muestra su suma en
pantalla
'''

# inicio
a = int(input())
b = int(input())
s = a + b
print(s)
# fin
```

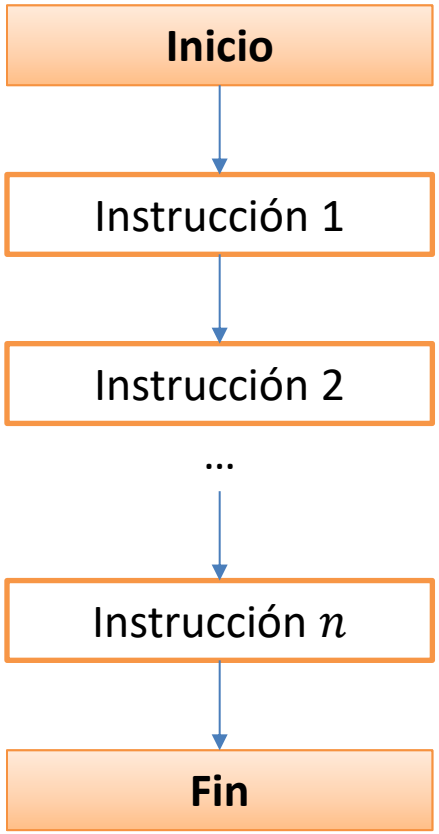


Necesidad de estructuras selectivas/repetitivas

En Python las instrucciones se ejecutan sucesivamente una tras otra. Esto define un camino que va desarrollado el programa (*flujo de control*).

Sin embargo, habrá momentos en que el programa deba ejecutar determinadas partes dependiendo del estado en el que se halle el programa o las variables.

Las estructuras de selección, repetición e invocación (*funciones*) permiten que el flujo secuencial del programa sea modificado en un modo preciso y definido con anterioridad.



Necesidad de estructuras selectivas

Problema: mostrar la calificación de un alumno a partir del puntaje obtenido en el examen, de acuerdo a la siguiente tabla:

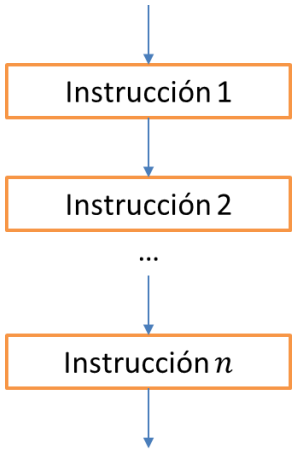
Puntaje	Calificación
Entre 90 y 100	5
Entre 80 y 89	4
Entre 70 y 79	3
Entre 60 y 69	2
Menos de 60	1

Programación estructurada

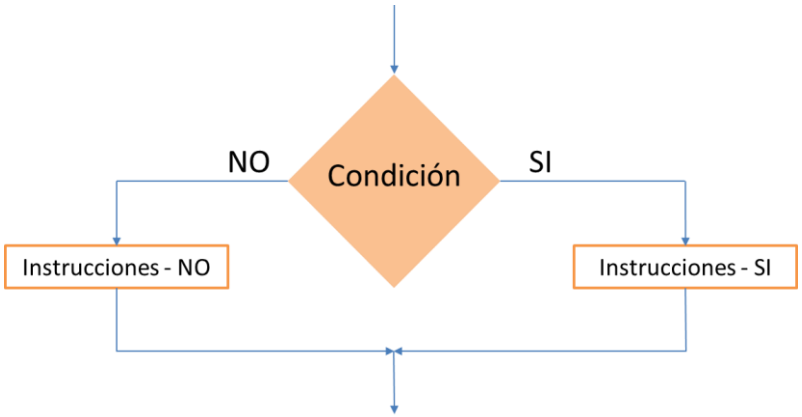
Cualquier algoritmo, no importa su complejidad puede ser construido utilizando combinaciones de tres estructuras de control de flujo estandarizadas (secuencial, selección y repetitiva o iterativa).

La **programación estructurada** es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa, utilizando únicamente subrutinas (*funciones*) y tres estructuras: secuencial, de selección (`if` y `if-else, match-case`) y de repetición (bucles `for` y `while`).

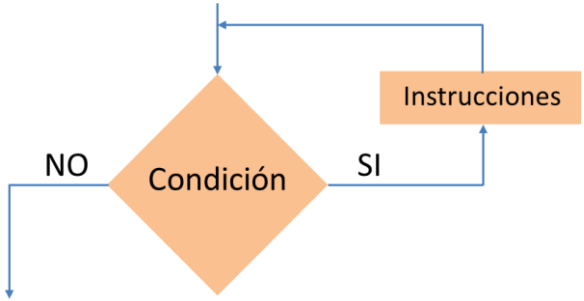
Secuencial



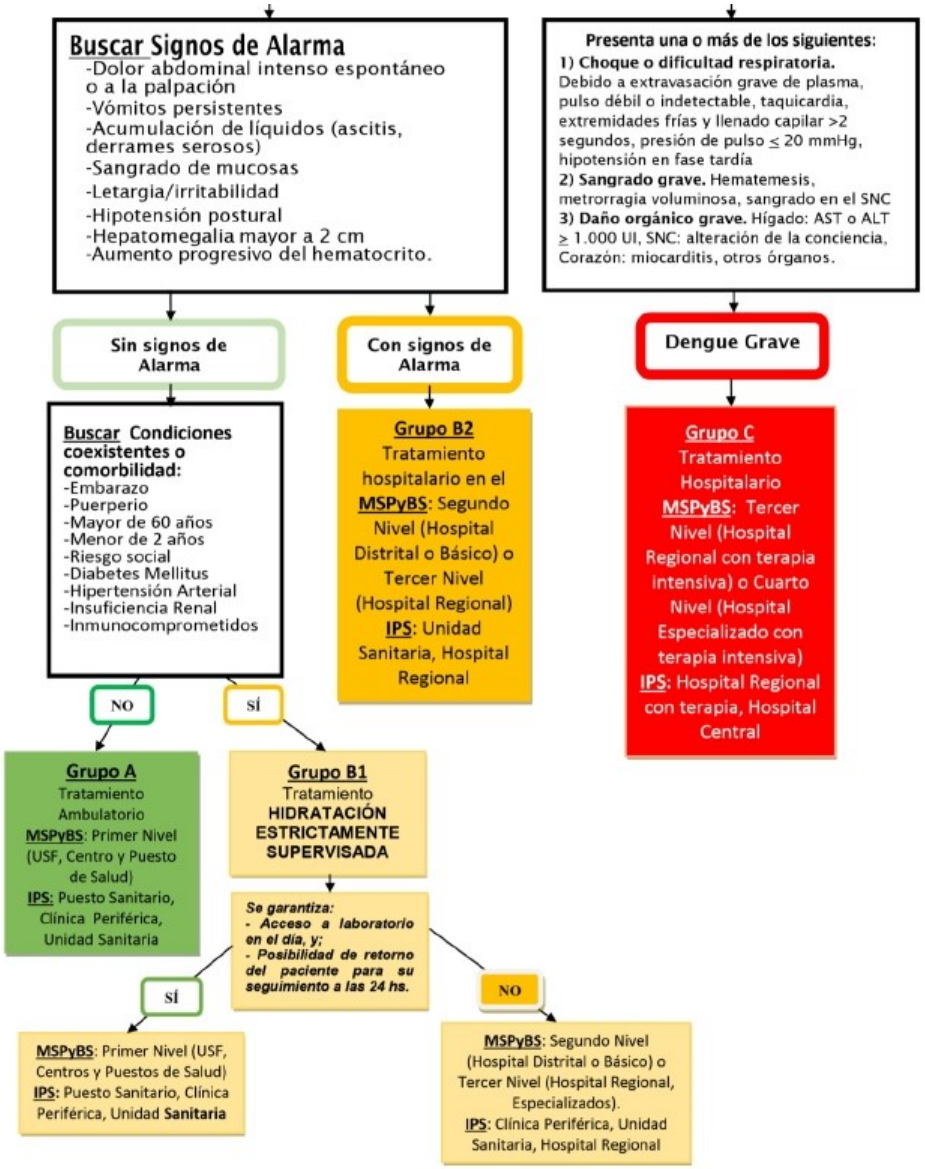
Selección



Repetición



Ejemplo - Dengue



Ejemplo - Dengue

Ministerio de SALUD PÚBLICA Y BIENESTAR SOCIAL

GOBIERNO NACIONAL

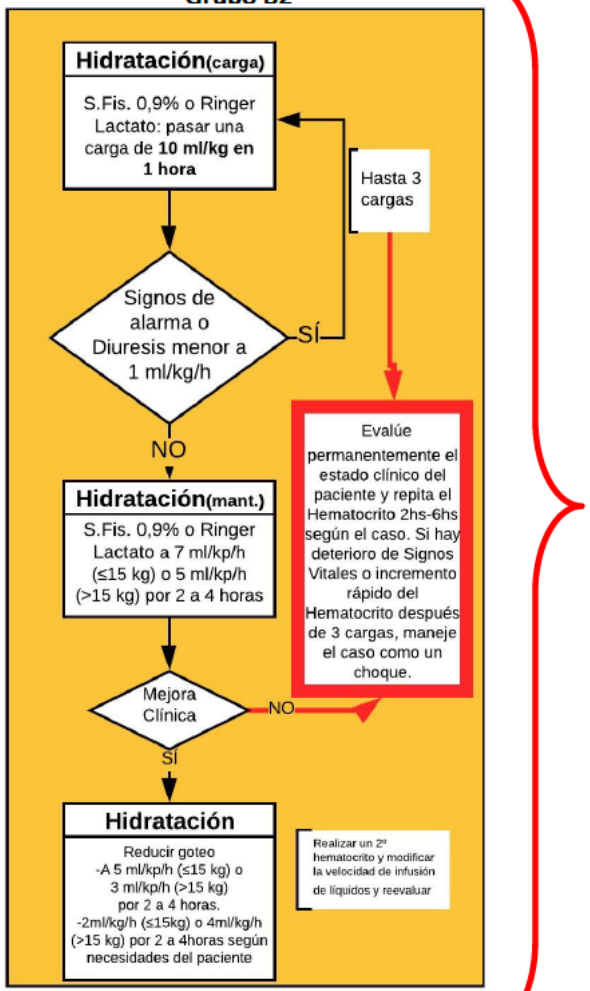
Grupo A

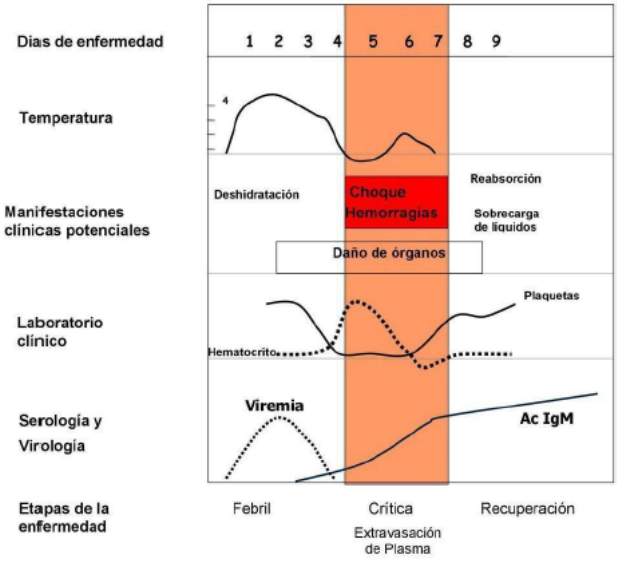
Laboratorio:
Hemograma
Tratamiento:
-Hidratación Vía Oral
-Utilización de mosquitero
-Reposo en cama
-Adecuada ingesta de líquidos
-Paracetamol
-Instruir sobre signos de alarma
-NO AINES
Entregar ficha de seguimiento ambulatorio de pacientes

Grupo B1

Laboratorio:
Hemograma
Tratamiento:
-Sintomático igual que Grupo A
-Mantener vía oral con suero oral
-Si no tolera, iniciar IV con S.Fis. 0,9% o Ringer lactato a dosis de mantenimiento
-Internación: Plantear derivación a 2º Nivel (de acuerdo a comorbilidad y signos de alarma y signos iniciales de shock compensado).

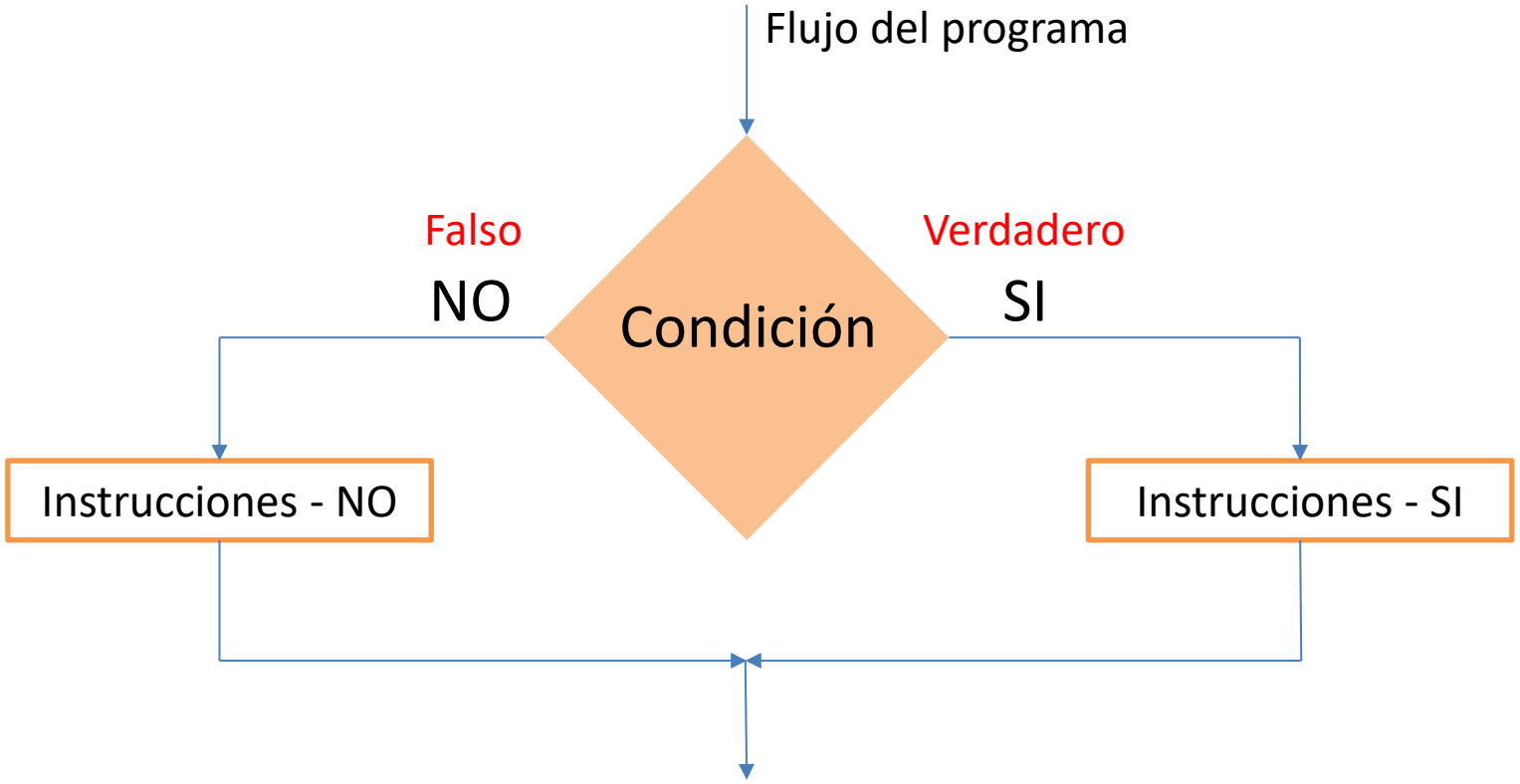
Grupo B2





Estructuras selectivas

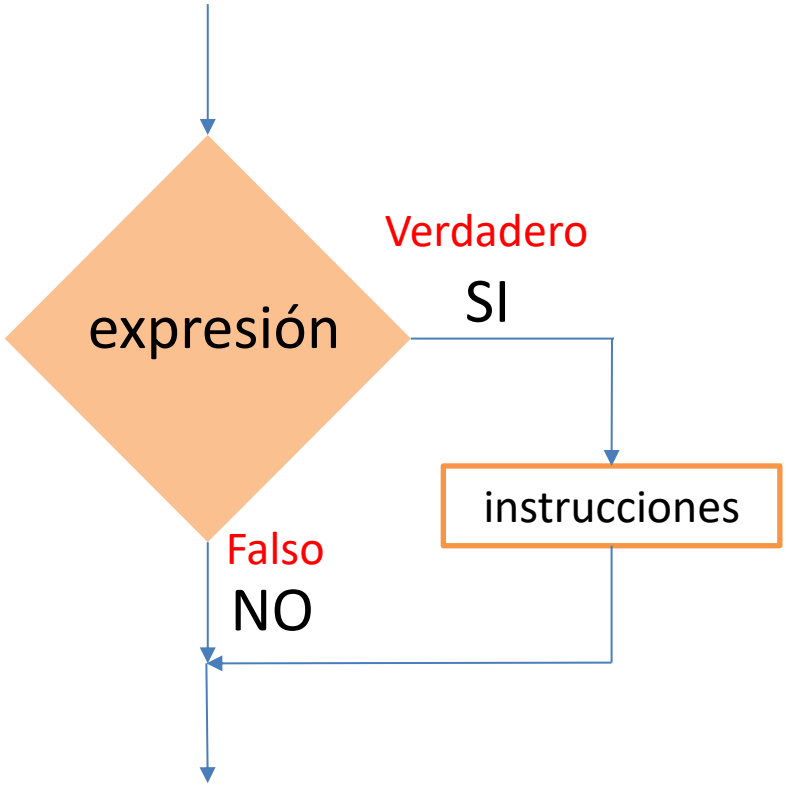
En la **estructura selectiva** se tiene una instrucción o grupo de instrucciones que se pueden ejecutar o no, en función del valor de una **condición**. También conocida como **bifurcación condicional**.



Estructura selectiva `if`

La bifurcación condicional **`if`** tiene la siguiente sintaxis:

```
if expresión:
    instrucciones
```

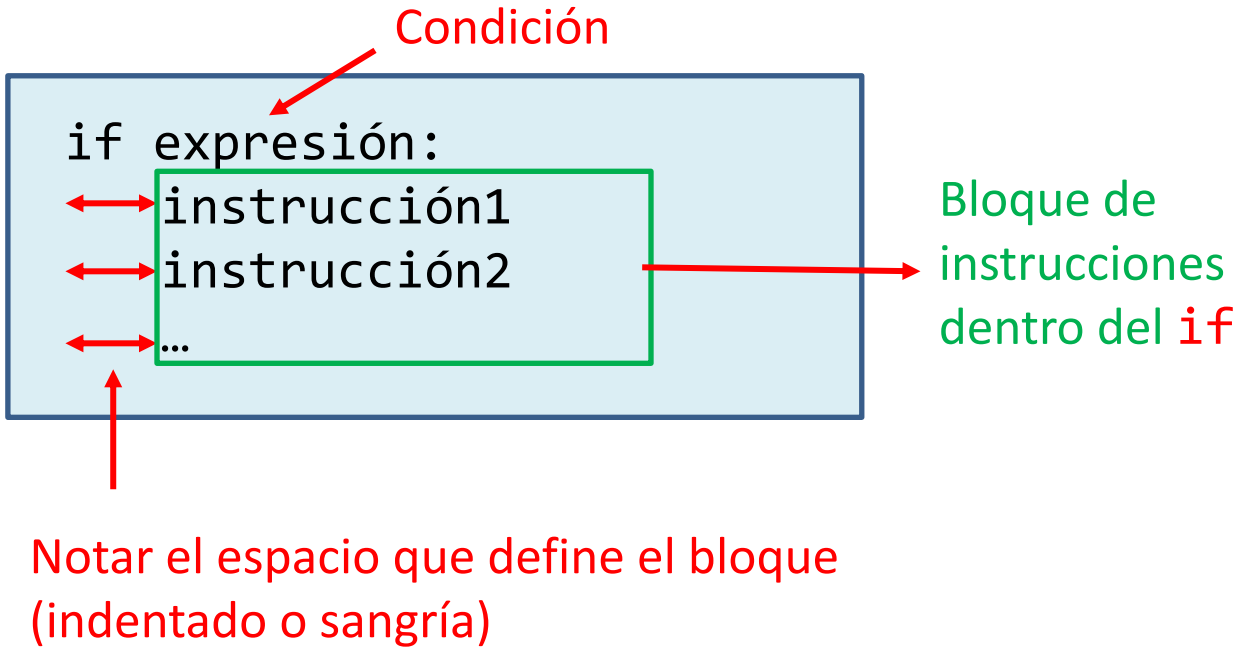


`if` permite al programa tomar decisiones. En su forma más simple, las `instrucciones` sólo se ejecutan si el resultado de evaluar la expresión es verdadero (`True`). Si es falso (`False`), nada se ejecuta.

Uso de sangría o *indentación* en Python

A diferencia de otros lenguajes de programación, Python no emplea llaves { } para delimitar *bloques* de código. En cambio, emplea la **sangría** o ***indentación***; que consiste en agregar espacios o tabulaciones para mover instrucciones a la derecha.

Un bloque de código dentro de una estructura siempre estará a su derecha.



Expresiones booleanas

- La **condición** en la estructura `if` se llama **expresión booleana**.

- Las expresiones booleanas pueden ser verdaderas (**True**) o falsas (**False**).

```
>>> 0<1
True
>>> 5%2==0
False
```

- Estas expresiones son formadas con la ayuda de operadores **relacionales** y **lógicos**. Más adelante veremos otros operadores, como los de **identidad** (is) y **membresía** (in).

Expresión dentro del `if`

Expresiones relacionales

Son expresiones que permiten determinar la relación que existe entre dos operandos a través de un operador relacional. El mismo permite comparar los valores de dos operandos de igual naturaleza. El resultado de evaluar una expresión relacional siempre es verdadero (`True`) o falso (`False`).

Operadores relacionales

- Igual que: ==
- Menor que: <
- Mayor que: >
- Menor o igual que: <=
- Mayor o igual que: >=
- Distinto que: !=

Ejemplos:

- `10 > 20` ==> `False`
- `20 == 10` ==> `False`
- `5 == 5` ==> `True`
- `'a' == 'a'` ==> `True`
- `1!=2` ==> `True`
- `3+5 <= 7` ==> `False`

Estructura selectiva `if`

Ejemplo 1: (con una expresión relacional simple)

Imprimir el mensaje “Es un numero negativo” cuando un número entero leído por teclado es negativo.

```
n = int(input("Ingrese un numero: "))
if n<0:
    print("Es un numero negativo")
```

```
========== RESTART:
Ingrese un numero: -6
Es un numero negativo
>>>
========== RESTART:
Ingrese un numero: 3
>>>
```

Expresión dentro del `if`

Obs: Tener en cuenta el orden de precedencia de los operadores!!

Expresiones lógicas

Las expresiones lógicas consisten en variables booleanas, constantes booleanas (`True` o `False` en Python), expresiones relacionales y operadores booleanos (`and`, `or`, `not`).

Operador	Descripción
and	**True** si ambas condiciones son verdaderas
or	**True** si al menos una de las condiciones es verdadera
not	**True** si la condición es falsa

and	or	not
V and V ==> V	V or V ==> V	not V ==> F
F and V ==> F	F or V ==> V	not F ==> V
V and F ==> F	V or F ==> V	
F and F ==> F	F or F ==> F	

Estructura selectiva if

Ejemplo 2: (con una expresión lógica)

Imprimir el mensaje “Es un nro. mayor a 100 y es par” cuando un número leído por teclado es mayor a 100 y es par.

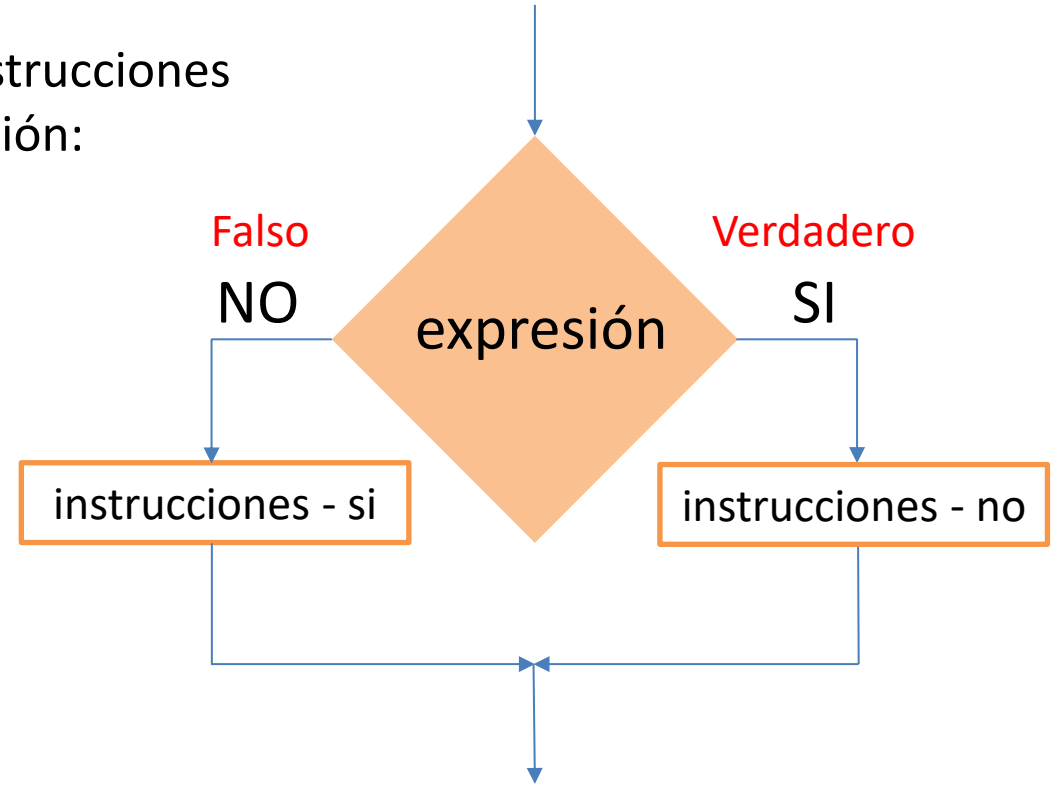
```
n = int(input("Ingrese un numero: "))
if n>100 and n%2==0:
    print("Es un nro. mayor a 100 y es par")
```

```
========== RESTART: C:
Ingrese un numero: 102
Es un nro. mayor a 100 y es par
>>>
========== RESTART: C:
Ingrese un numero: 103
>>>
========== RESTART: C:
Ingrese un numero: 96
>>>
```

Estructura selectiva `if-else`

En el caso de querer ejecutar instrucciones para cada resultado de la condición:

```
if expresión:
    instrucciones - si
else:
    instrucciones - no
```



En esta segunda forma tenemos dos posibilidades: si al evaluar la expresión el resultado es verdadero se ejecuta las `instrucciones-si`, pero si el resultado es falso se ejecuta las `instrucciones-no`. ***En cualquier caso, se ejecuta sólo uno de los dos grupos de instrucciones.***

Estructura selectiva `if-else`

Ejemplo 3:

Imprimir el mensaje “Es un nro. negativo” cuando un número entero leído por teclado es negativo, y “Es un nro. mayor o igual a cero” cuando no.

```
n = int(input("Ingrese un numero: "))
if n<0:
    print("Es un nro. negativo")
else:
    print("Es un nro. mayor o igual a cero")
```

```
========== RESTART: C:
Ingrese un numero: 5
Es un nro. mayor o igual a cero
>>>
========== RESTART: C:
Ingrese un numero: -2
Es un nro. negativo
>>>
```

Estructura selectiva anidada

Una estructura `if` o `if-else` puede estar dentro de otro `if` o `else`. En este caso, se tiene una estructura selectiva *anidada*. De esta manera, aumenta el número de caminos posibles.

Ejemplo 4-a:

Escribir un programa en Python que indique en pantalla si el número ingresado por teclado es negativo, positivo o cero.

```
n = int(input("Ingrese un numero: "))
if n<0:
    print("Es un nro. negativo")
else:
    if n>0:
        print("Es un nro. positivo")
    else:
        print("El nro. es cero")
```

Estructura selectiva anidada

Una estructura `if` o `if-else` puede estar dentro de otro `if` o `else`. En este caso, se tiene una estructura selectiva *anidada*. De esta manera, aumenta el número de caminos posibles.

Ejemplo 4-b:

Escribir un programa en Python que indique en pantalla si el número ingresado por teclado es negativo, positivo o cero.

```
n = int(input("Ingrese un numero: "))
if n>=0:
    if n>0:
        print("Es un nro. positivo")
    else:
        print("El nro. es cero")
else:
    print("Es un nro. negativo")
```

Estructura selectiva anidada

Ejemplo 5:

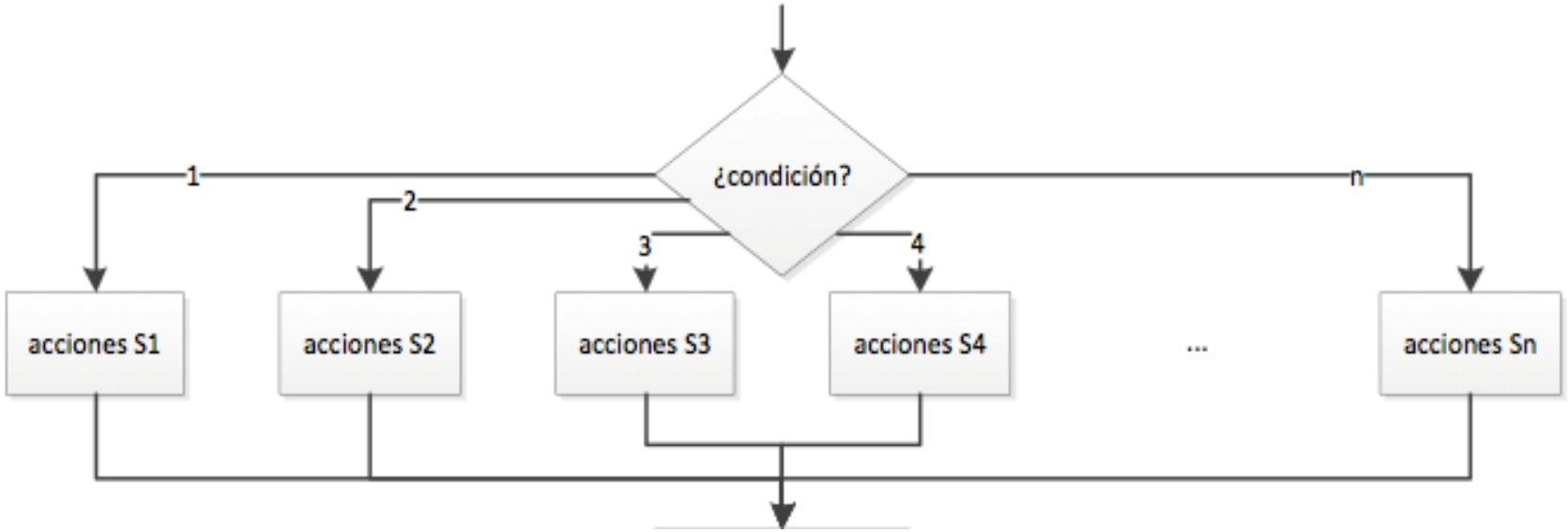
Escribir un programa en Python que lea 3 números por teclado e indique el valor central.

```
a = int(input("Primer numero: "))
b = int(input("Segundo numero: "))
c = int(input("Tercer numero: "))
if a<=b:
    if b<=c:
        central=b
    else: #b>c
        if a<=c:
            central=c
        else:
            central=a
else: #a>b
    if a<=c:
        central=a
    else: #a>c
        if b<=c:
            central=c
        else:
            central=b
print("Valor central:",central)
```

Estructura selectiva múltiple

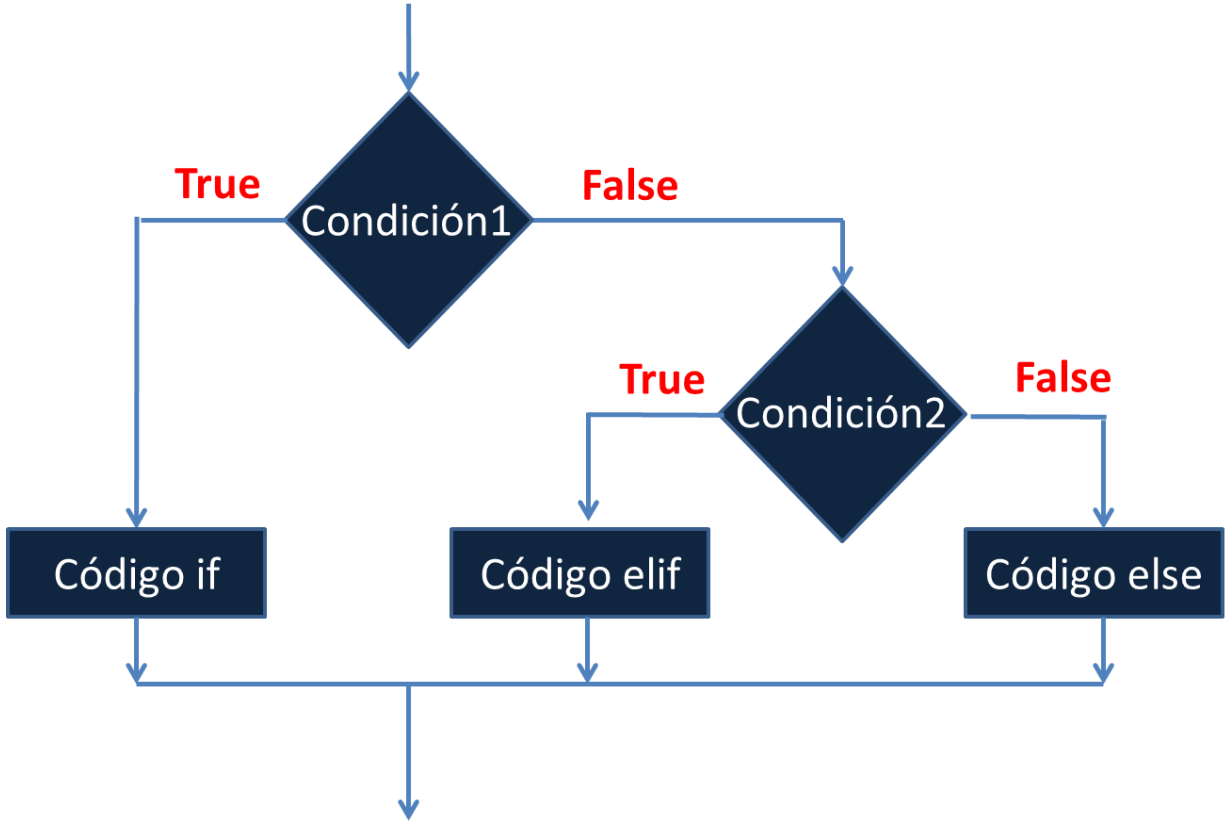
La estructura de decisión múltiple evaluará una expresión que podrá conducir a n caminos distintos, 1, 2, 3, …, n . Según se elija una de estas posibilidades en la condición, se realizará una de las n acciones.

Esta estructura también puede ser resuelta con estructuras simples o dobles anidadas. Sin embargo, si el número de alternativas es grande puede plantear serios problemas de escritura del algoritmo y naturalmente de legibilidad. Para facilitar esto, Python proporciona `elif`.



Estructura selectiva múltiple

```
if condición 1:
        realizar acción 1
elif condición 2:
        realizar acción 2
elif condición 3:
        realizar acción 3
…
elif condición n:
        realizar acción n
else: # ninguna de las anteriores condiciones se cumple
        hacer algo
```



Ejercicio 1

Mostrar la calificación de un alumno a partir del puntaje obtenido en el examen, de acuerdo a la siguiente tabla:

Puntaje	Calificación
Entre 90 y 100	5
Entre 80 y 89	4
Entre 70 y 79	3
Entre 60 y 69	2
Menos de 60	1

Ejercicio 1

Mostrar la calificación de un alumno a partir del puntaje obtenido en el examen, de acuerdo a la siguiente tabla:

```
puntaje = int(input("Ingrese el puntaje: "))
if puntaje>=90:
    calificacion=5
elif puntaje>=80:
    calificacion=4
elif puntaje>=70:
    calificacion=3
elif puntaje>=60:
    calificacion=2
else:
    calificacion=1
print("La calificacion es:",calificacion)
```

```
========== RESTART:
Ingrese el puntaje: 78
La calificacion es: 3
>>>
========== RESTART:
Ingrese el puntaje: 51
La calificacion es: 1
>>>
```

Operador ternario

Python permite el uso de operadores ternarios para expresar estructuras condicionales de forma más concisa:

```
resultado_si_verd if condicion else resultado_si_falso
```

Cabe resaltar que el operador ternario entrega un resultado en función a la condición establecida. Por ejemplo, la siguiente expresión:

```
minimo = a if a<b else b
```

Carga en `minimo` el menor valor entre `a` y `b`

Ejemplo 6:

¿Qué imprimirá el siguiente programa?

```
puntuacion=60
resultado = "Aprobado" if puntuacion>=70 else "Reprobado"
print(resultado)
```

Ejercicios propuestos

Ejercicio 1:

Dados dos valores a y b, indicar a través de un mensaje cuál de los 2 es el mayor (Ej: “El mayor es a”). En caso de que sean iguales, indicarlo a través de un mensaje.

Ejercicio 2:

Si los días LUN a DOM se ingresan de forma numérica (del 1 al 7), devolver el nombre del día correspondiente. Si el número no es válido, indicarlo con un mensaje.

Ejercicios propuestos

Ejercicio 3:

Escribir un programa que determine si un alumno tiene o no derecho a examen final en una materia. Un alumno tiene firma si la suma de los puntajes (1P: 24, 2P: 36, TPs/Lab: 10) es mayor o igual a 28. Si tiene derecho entonces imprimir el nombre y su puntaje total. Si no tiene derecho indicar que debe rendir el tercer parcial.

Se leen los puntajes de los dos parciales y el correspondiente a trabajos prácticos y/o laboratorios, y también el nombre del alumno.

Ejemplos:

- Si parc1=20, parc2=30, tps=10, y nombre="Julia"
  Se debe imprimir:
  Julia tiene firma con 60.

- Si parc1=10, parc2=5, tps=5, y nombre="Jose"
  Se debe imprimir:
  Jose no consiguió firma.

Ejercicios propuestos

Ejercicio 4:

Hacer un algoritmo que determine si tres valores ingresados pueden ser lados de un triángulo. Ninguno de sus lados puede ser superior o igual a la suma de los otros dos. Si los valores pueden ser lados de un triángulo, entonces calcular la superficie según la fórmula del semiperímetro.

Ejemplos:
- Si a=10, b=40 y c=100
Se imprime : “No pueden ser lados de un triangulo”.
- Si a=10, b=40 y c=35
Se imprime : “Pueden ser lados de un triangulo”.
Su superficie es:...

Nota: para obtener la raíz cuadrada de un número se utiliza la función `sqrt()` que está en `math`:

```
import math
…
y = math.sqrt(x) #calcula la raíz cuadrada de x y lo asigna a y
```

Ejercicios propuestos

Ejercicio 5:

Determinar si un año (ingresado por teclado) es bisiesto o no, teniendo en cuenta lo siguiente:

- Un año es bisiesto si es múltiplo de 4 pero no de 100, a no ser que lo sea también de 400.

Ejercicio 6 (Desafío):

Diseñar un programa en el que se ingresan tres variables: DIA, MES y ANHO (en forma numérica); y devuelva la fecha del día siguiente (en formato DIA/MES/ANHO). Se deben considerar los años bisiestos, cantidad de días de cada mes, etc. En caso de insertar números reales o fechas inválidas, indicar con un mensaje.

Gracias por la atención