CEPSUNI

Dirección de Responsabilidad Social y Desarrollo Sostenible

PYTHON BÁSICO - NIVEL I

TANIA ALDORADIN QUINTANILLA

Clase 2: Estructuras de control

Objetivos

Al finalizar con éxito este curso, podrá diseñar, crear, modificar programas usando el lenguaje de programación Python vas a:

- Comenzar con la instalación de Python.
- Conocer los diferentes IDE de Python.
- Conocer las sentencias de control.
- Conocer las sentencias de repetición.
- Conocer los arreglos lineales (Listas, Tuplas y Diccionarios).
- Conocer las operaciones con conjuntos (Unión e intercepción).

Contenido

- Estructuras de control selectivas simples
- Estructuras de control selectivas dobles
- Estructuras de control selectivas múltiples
- Manejo de excepciones
- Ejercicios de aplicación.

Operadores de comparación

SÍMBOLO	DESCRIPCIÓN	EJEMPLO	BOOLEANO
==	IGUAL QUE	5 == 7	FALSE
!=	DISTINTO QUE	ROJO != VERDE	TRUE
<	MENOR QUE	8 < 12	TRUE
>	MAYOR QUE	12 > 7	TRUE
<=	MENOR O IGUAL QUE	16 <= 17	TRUE
>=	MAYOR O IGUAL QUE	67 >= 72	FALSE

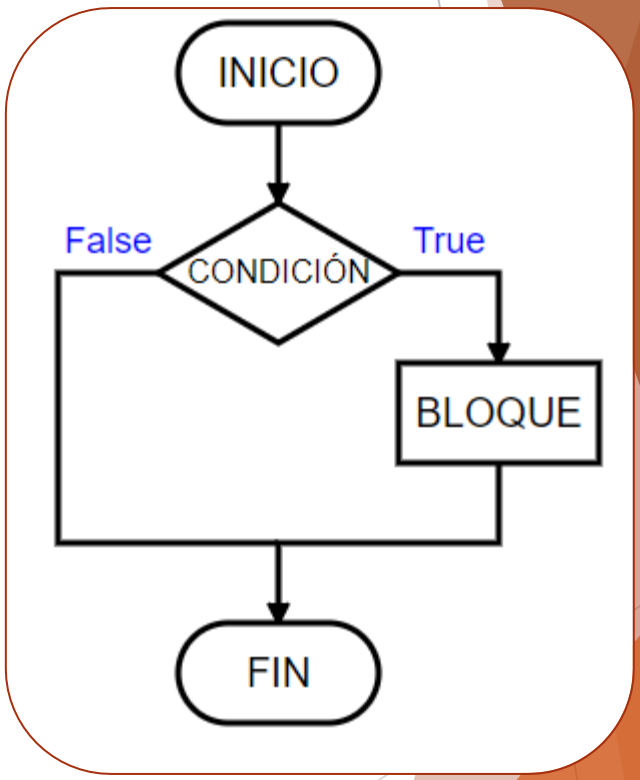
Estructuras de control selectivas simples

- Python es uno de los lenguajes de programación más utilizados en el mundo.
- Fue creado y lanzado por Guido van Rossum en 1991 y ha evolucionado enormemente a lo largo de los años gracias a sus colaboradores. Se utiliza principalmente para desarrollo web, desarrollo de software, inteligencia artificial, scripting, matemáticas y más.
- Si necesita desarrollar códigos para un proyecto grande, utilizando un software dedicado y de alta calidad. Se recomienda Python IDE.

Estructuras de control selectivas simples

```
a = 50
b = 120
if b > a:
  print("b es mayor que a")
```

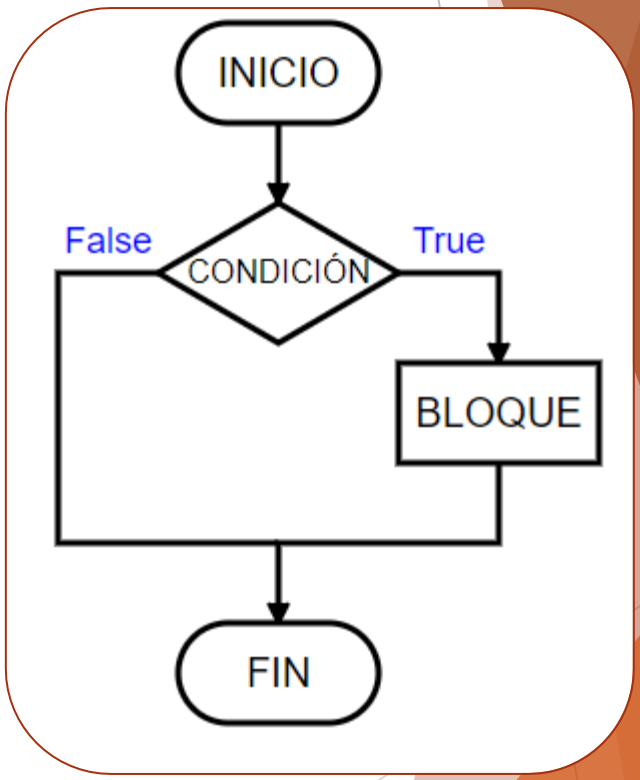
```
1    a = 50
2    b = 120
3    if b > a:
4        print("b es mayor que a")
```



Indentación

```
a = 50
b = 120
if b > a:
print("b es mayor que a")
```

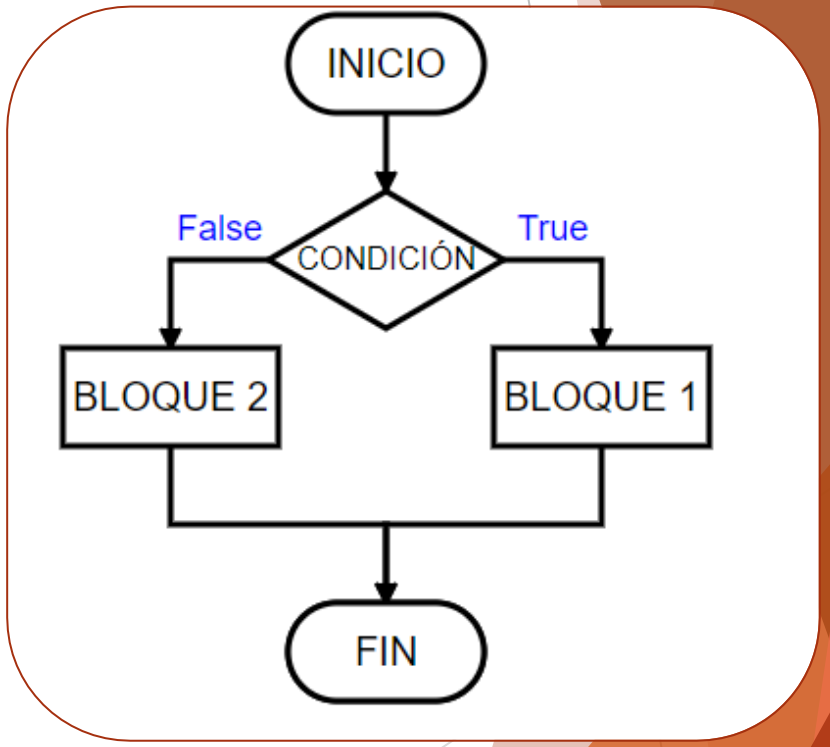
```
1    a = 50
2    b = 120
3    if b > a:
4    print("b es mayor que a")
```



Estructuras de control selectivas dobles

```
a = 120
b = 50
if b > a:
  print("b es mayor que a")
else:
  print("b no es mayor que a")
```

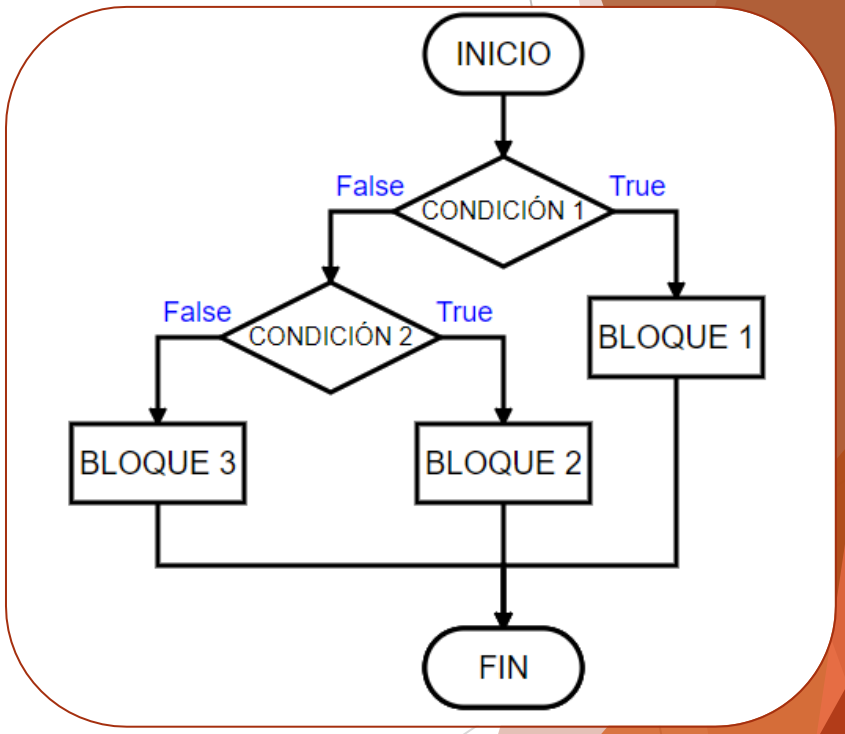
```
1   a = 120
2   b = 50
3   if b > a:
4   |  print("b es mayor que a")
5   else:
6   |  print("b no es mayor que a")
```



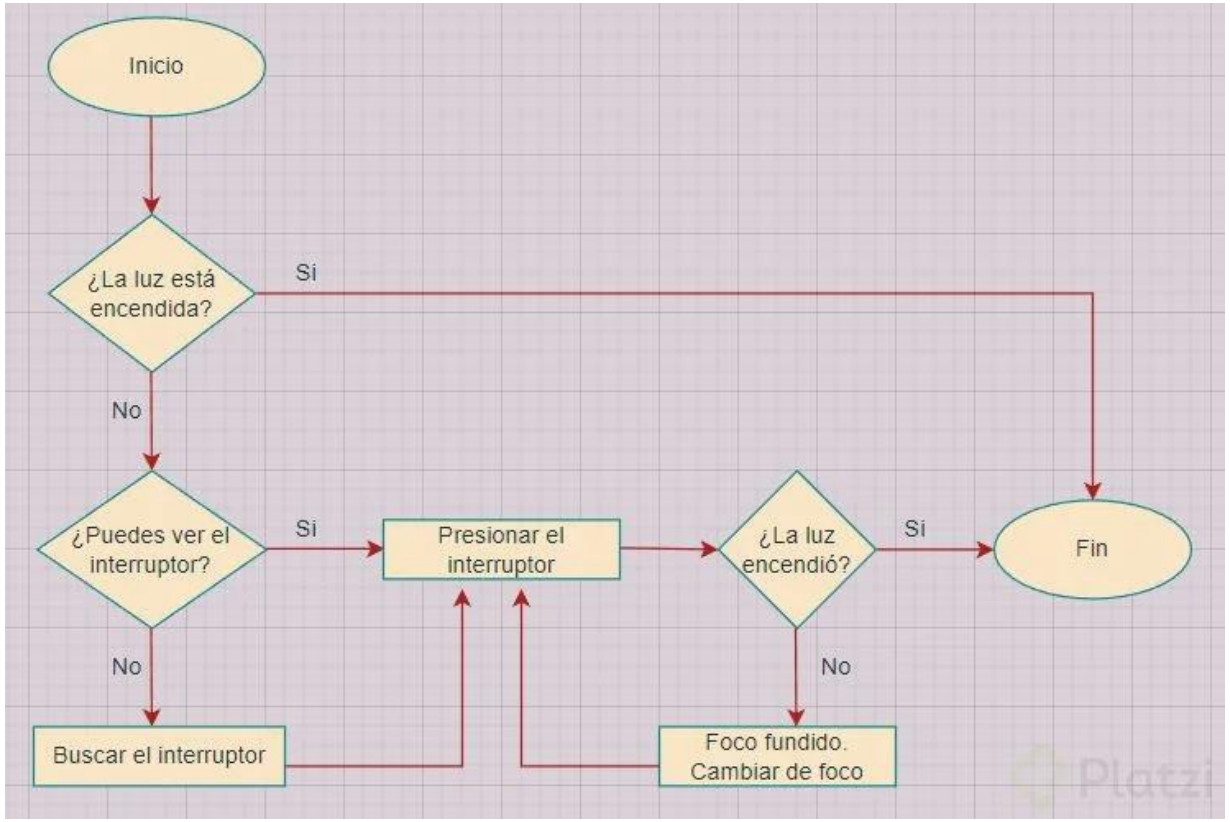
Estructuras de control selectivas múltiples

```
a = 120
b = 50
if b > a:
  print("b es mayor que a")
elif a == b:
  print("a y b son iguales")
else:
  print("a es mayor que b")
```

```
1   a = 120
2   b = 50
3   if b > a:
4   |  print("b es mayor que a")
5   elif a == b:
6   |  print("a y b son iguales")
7   else:
8   |  print("a es mayor que b")
```



Ejercicio


Inicio
¿La luz está encendida?
Si
No
¿Puedes ver el interruptor?
Si
Presionar el interruptor
¿La luz encendió?
Si
Fin
No
No
Buscar el interruptor
Foco fundido. Cambiar de foco

Short

```
1    x = 120
2    y = 50
3    if x > y: print("x es mayor que y")
```

Operadores lógicos

Y	and
O	or
No	not

And

p	q	p^q
V	V	V
V	F	F
F	V	F
F	F	F

```python
p = 120
q = 50
r = 280
if p > q and r > p:
    print("Ambas condiciones son verdaderas")
```

Lógica correcta

```
3 > 2 > 1
(3 > 2) and (2 > 1)
True and True
True
```

Lógica incorrecta

```
3 > 2 > 1
(3 > 2) > 1
True > 1
False
```

Or

p	q	pvq
V	V	V
V	F	V
F	V	V
F	F	F

```
1   p = 120
2   q = 50
3   r = 280
4   if p > q or p > r:
5       print("Al menos una de las condiciones es verdadera")
```

Operator	Meaning	Example	Result
and	Logical and	(5<2) and (5>3)	False
or	Logical or	(5<2) or (5>3)	True

Not

p	not p
V	F
F	V

```
1    a = 50
2    b = 120
3    if not a > b:
4        print("a no es mayor que b")
```

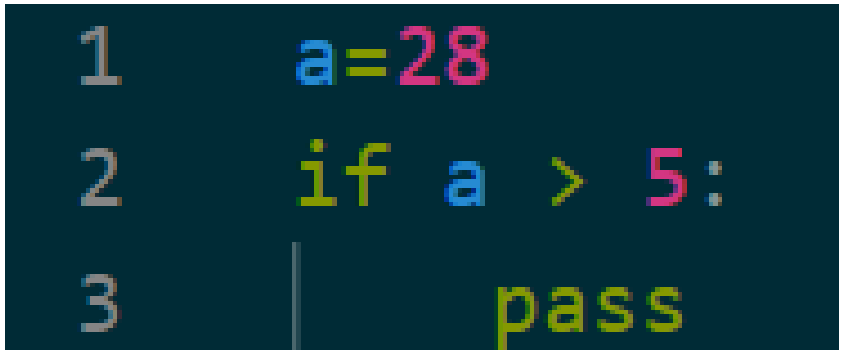
If anidadas

```
1   m = 73
2   if m > 10:
3       print("m es mayor a 10")
4       if m > 20:
5           print("m tambien es mayor a 20")
6       else:
7           print("pero m no es mas que 20")
```

```
1   m = 16
2   if m > 10:
3       print("m es mayor a 10")
4       if m > 20:
5           print("m tambien es mayor a 20")
6       else:
7           print("pero m no es mas que 20")
```

Pass

Si tenemos un if sin contenido, tal vez porque sea una tarea pendiente que se deja para implementar en un futuro. Es necesario hacer uso de pass para evitar el error. Realmente pass no hace nada, simplemente es para tener contento al interprete de código.



Manejo de excepciones

Las excepciones en Python son una herramienta muy potente que la gran mayoría de lenguajes de programación modernos tienen. Se trata de una forma de controlar el comportamiento de un programa cuando se produce un error.

```
1    a = 8
2    b = 5
3    c =a/b
4    print(c)
```

```
6    a = 9; b = 0
7    print(a/b)
ZeroDivisionError: division by zero
```

Se trata de la excepción ZeroDivisionError.

Manejo de excepciones

¿Que pasaría si intentásemos sumar un número con un texto?

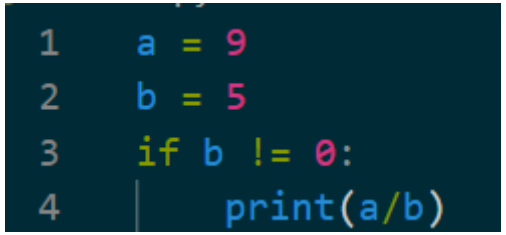
```
1    print(7 + "7")
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Por ello es muy importante controlar las excepciones, porque por muchas comprobaciones que se realiza es posible que en algún momento ocurra una, y si no se hace nada el programa se parará.

¿Imagina que las ventas de una tienda, un avión o un cajero automático tiene un error, es decir una excepción y se detiene por completo?

Manejo de excepciones

En una primer momento para el control de excepciones podría ser el siguiente ejemplo. Podemos realizar una comprobación manual de que no estamos dividiendo por cero, para así evitar tener un error tipo ZeroDivisionError.



Sin embargo todo esto es complicado, ya que se tendría que escribir código que prevenga todo tipo de excepciones. Por ello existe el uso de except.

Manejo de excepciones

Lo bueno es que existe try y except, el cual pueden capturar y manejar adecuadamente las excepciones, sin que el programa se detenga.

```
1   a = 9
2   b = 0
3   try:
4       c = a/b
5   except ZeroDivisionError:
6       print("No se ha podido realizar la división")
```

Ahora ya el programa no se detiene.

Entonces, lo que hay dentro del try es la sección del código que podría lanzar la excepción que se está capturando en el except. Por lo tanto cuando ocurra una excepción, se entra en el except pero el programa no se detiene.

Manejo de excepciones

```
1   c=8
2   d=3
3   e=0
4
5   try:
6       c = d/e          # Comentar esta linea para ver TypeError
7       #f = 2 + "Hola" # Comentar esta linea para ver ZeroDivisionError
8   except ZeroDivisionError:
9       print("No se puede dividir entre cero!")
10  except TypeError:
11      print("Problema de tipos!")
```

Manejo de excepciones

```
1   c=8
2   d=3
3   e=0
4
5   try:
6       #c = d/e          # Comentar esta linea para ver TypeError
7       f = 2 + "Hola" # Comentar esta linea para ver ZeroDivisionError
8   except (ZeroDivisionError, TypeError):
9       print("No se puede dividir entre cero o hay un error de tipo!!!!!!!")
```

Try-Except Exception

```
1   c=8
2   d=3
3   e=0
4
5   try:
6       #c = d/e          # Comentar esta linea para ver TypeError
7       f = 2 + "Hola" # Comentar esta linea para ver ZeroDivisionError
8   except  Exception:
9       print("Se ha encontrado una excepción")
```

Try-Except-Else

- Este bloque se ejecuta si no ha ocurrido ninguna excepción.

```
1   x=6
2   y=0
3
4   try:
5       div = x/y
6   except  Exception:
7       print("Se ha encontrado una excepción")
8   else:
9       print("No ha ocurrido ninguna excepción")
```

```
1   x=6
2   y=3
3
4   try:
5       div = x/y
6   except  Exception:
7       print("Se ha encontrado una excepción")
8   else:
9       print("No ha ocurrido ninguna excepción")
```

Try-Except-Else-Finally

Este bloque **se ejecuta siempre**, haya o no haya habido excepción.

```
1   x=6
2   y=0
3
4   try:
5       div = x/y
6   except:
7       print("Haz entrado en el bloque de except, ha ocurrido una excepción")
8   else:
9       print("No ha ocurrido una excepción en tu programa")
10  finally:
11      print("Haz entrado en el bloque de finally, se ejecuta el bloque finally")
```

```
1   x=6
2   y=3
3
4   try:
5       div = x/y
6   except:
7       print("Haz entrado en el bloque de except, ha ocurrido una excepción")
8   else:
9       print("No ha ocurrido una excepción en tu programa")
10  finally:
11      print("Haz entrado en el bloque de finally, se ejecuta el bloque finally")
```