Status	Finished
Started	Wednesday, 6 November 2024, 8:52 AM
Completed	Friday, 22 November 2024, 11:17 AM
Duration	16 days 2 hours
Marks	9.00/9.00
Grade	**10.00** out of 10.00 (**100**%)

Information

Sentencias de control

Prácticamente todos los programas deben contemplar distintas situaciones o variabilidad en los datos, que requerirán realizar acciones distintas según el caso. Pero esto no sería posible si los programas únicamente pudieran ejecutar sus instrucciones una tras otra, de la primera a la última.

Por tanto, los lenguajes de programación, además de las instrucciones básicas, disponen de estructuras y **sentencias de control** que permiten reorientar *el flujo de ejecución* hacia unas u otras instrucciones para así adaptarse a las distintas situaciones previstas.

En particular, existen sentencias de control que permiten elegir, en función de una (o más) condiciones, entre dos (o más) posibles "vías" por los que continuar la ejecución; y otras que permiten ejecutar repetidamente un mismo grupo de instrucciones, ya sea un cierto número de veces o mientras se cumpla una condición. En esta lección nos centraremos en las primeras.

Una **condición** es cualquier expresión que al ser evaluada dé un resultado booleano: en Python un valor **True** o **False** que nos indicará, respectivamente, si se ha cumplido o no la condición.

Information

Condiciones

Las sentencias de control suelen decidir las acciones a ejecutar en función de condiciones, que son expresiones que al ser evaluadas devuelven un resultado booleano.

Para formar condiciones se usan con frecuencia los **operadores relacionales**, que son operadores que comparan los valores de sus dos operandos y devuelven un valor booleano de acuerdo con la relación entre ellos.

Supongamos, en los siguientes ejemplos, que la variable *a* tiene asociado el valor 10 y la variable *b* tiene asociado el valor 20:

operador	descripción	ejemplo
==	Compara si los operandos **son iguales** entre sí.	a == b *da falso*
!=	Compara si los operandos **no son iguales** entre sí.	a != b *da verdadero*
>	Compara si el operando izquierdo es **mayor** que el derecho.	a > b *da falso*
<	Compara si el operando izquierdo es **menor** que el derecho.	a < b *da verdadero*
>=	Compara si el operando izquierdo es **mayor o igual** que el derecho.	a >= b *da falso*
<=	Compara si el operando izquierdo es **menor o igual** que el derecho.	a <= b *da verdadero*

(En los símbolos de operador formados por dos caracteres, estos deben escribirse siempre juntos y en el orden indicado).

Se pueden formar condiciones más complejas combinando el resultado de condiciones simples usando **operadores booleanos**. Suponiendo que x e y sean valores booleanos:

operación	resultado
x **or** y	da False si ambos son False, si no, da True
x **and** y	da True si ambos son True, si no, da False
not x	da False si x es True, si no, da True

Ejemplos de condiciones compuestas:

```
x > y and y < z
```

```
(x > y or x > z) and y > z
```

```
x > 2 and 0 == x % 2
```

Téngase en cuenta que, aparte paréntesis, las operaciones aritméticas tienen precedencia sobre las comparaciones y estas sobre las operaciones booleanas. Por ejemplo, supongamos que x vale 3 al momento de evaluar la última expresión: primero se evalúa el operando izquierdo del and, para ello se compara si x es mayor que 2, resultando True, por lo que a continuación se evalúa el operando derecho, para lo cual primero se calcula el módulo (resto de dividir x entre 2), que da 1 como resultado, valor que se compara en igualdad con el 0, lo que da False; por lo que el resultado del and y, por tanto, de la expresión completa es el valor False.

Question 1

Complete

Mark 1.00 out of 1.00

Sea:

```
a = True
b = False
c = True
d = False
e = False
f = True
```

¿El resultado de evaluar la siguiente condición es Verdadero o Falso?

```
b and not d and (a or not a)
```

Select one:

- True
- False

Question 2

Complete

Mark 1.00 out of 1.00

Sea:

```
a = True
b = False
c = True
d = False
e = False
f = True
```

¿El resultado de evaluar la siguiente condición es Verdadero o Falso?

```
(e and not a) or (d and a) or (e and b)
```

Select one:

- True
- False

Question 3

Complete

Mark 1.00 out of 1.00

Sea:

```
a = True
b = False
c = True
d = False
e = False
f = True
```

¿El resultado de evaluar la siguiente condición es Verdadero o Falso?

```
a and not a or a
```

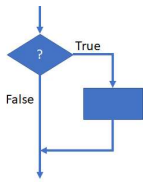
Select one:

- True
- False

Information

Sentencia if

La sentencia **if** permite decidir entre ejecutar, o no, una acción (o una secuencia de acciones) dependiendo de una condición.




```
num = int(input("Dame un número entero: "))
```

```
if num < 0:
    num = -num

print("El valor absoluto del número es ", num)
```

En Python la sentencia if empieza con la palabra `if` seguida de una condición y del caracter de dos puntos, controlando la ejecución de una o más instrucciones en las siguientes líneas; en el ejemplo, únicamente de la instrucción `num = -num`, que se ejecutará, o no, si la condición `num < 0` resulta verdadera o falsa, respectivamnnte.

En general, la sentencia de control **if** se usa para decidir en cada ocasión si se ejecuta, o no, un bloque de instrucciones. Si la decisión fuese no ejecutar dichas instrucciones, estas se "saltan", reanudando la ejecución del programa en la instrucción siguiente a ellas; si por el contrario se ejecutaran, posteriormente se continuaría la ejecución igualmente en la instrucción siguiente a ellas.

En todas las sentencias de control de Python, las instrucciones que controla se empiezan a escribir a partir de la siguiente línea y aumentando en uno el nivel de **sangrado** respecto a la sentencia de control (con un salto de tabulación o, preferiblemente, **cuatro espacios**).

Nótese, que la instrucción *print* del final del ejemplo, no forma parte de las instrucciones controladas por el if, ya que está al mismo nivel de sangrado que éste, por lo que forma parte de la secuencia primaria del programa.

Question 4

Complete

Mark 1.00 out of 1.00

Observe el siguiente código:

```
a = 54
b = 15
c = 0

if a > b:
    c = a
```

¿Con qué valor queda la variable c tras ejecutarlo?

Answer: 54

Question 5

Complete

Mark 1.00 out of 1.00

Observe el siguiente código:

```
a = 8
b = 15
c = 0
```

```
if a > b:
    c = b
```

¿Cuál es el valor de la variable c al ejecutarlo?

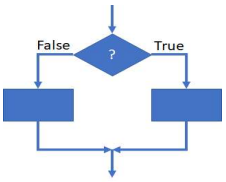
Answer: 0

Information

Cláusula else

La sentencia if se usa para decidir si se ejecuta, o no, un bloque de instrucciones. Si se decide no ejecutarlas, se saltan y la ejecución continúa en la instrucción siguiente a ellas; si se ejecutan, se alcanza ese mismo punto y se continúa.

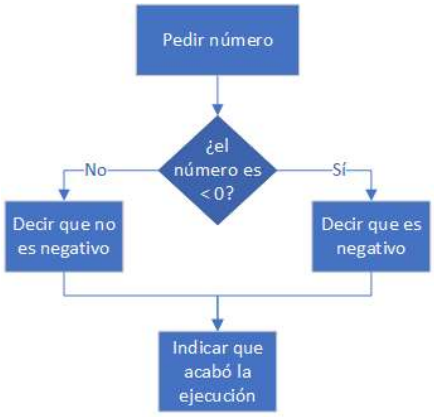
Se puede añadir una cláusula **else** a una sentencia if. Una sentencia **if-else** se usa para decidir cuál bloque de instrucciones ejecutar de entre dos posibles alternativas, reanudándose la ejecución en cualquiera de los dos casos a partir de la instrucción que sigue al segundo bloque.




```
num = int(input("Dame un número entero: "))

if num < 0:
    print(num, "es un número negativo")
else:
    print(num, "es un número no negativo")

print("Fin de la ejecución.")
```



En el ejemplo anterior se muestra un mensaje diferente dependiendo de si el valor numérico en la variable num recibido de teclado es o no negativo.

Question 6

Complete

Mark 1.00 out of 1.00

Observe el siguiente código:

```
a = 180
b = 90

if a > b and b % 5 != 0:
    c = a
else:
    c = b
```

¿Cuál es el valor asociado a la variable c cuando termina de ejecutarse?

Answer: 90

Question 7

Complete

Mark 1.00 out of 1.00

Observe el siguiente código:

```
a = 180
b = 90

if a < b:
    c = a
else:
    c = b
```

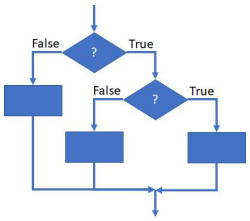
¿Cuál es el valor asociado a la variable c cuando termina de ejecutarse?

Answer: 90

Information

"Anidamiento"

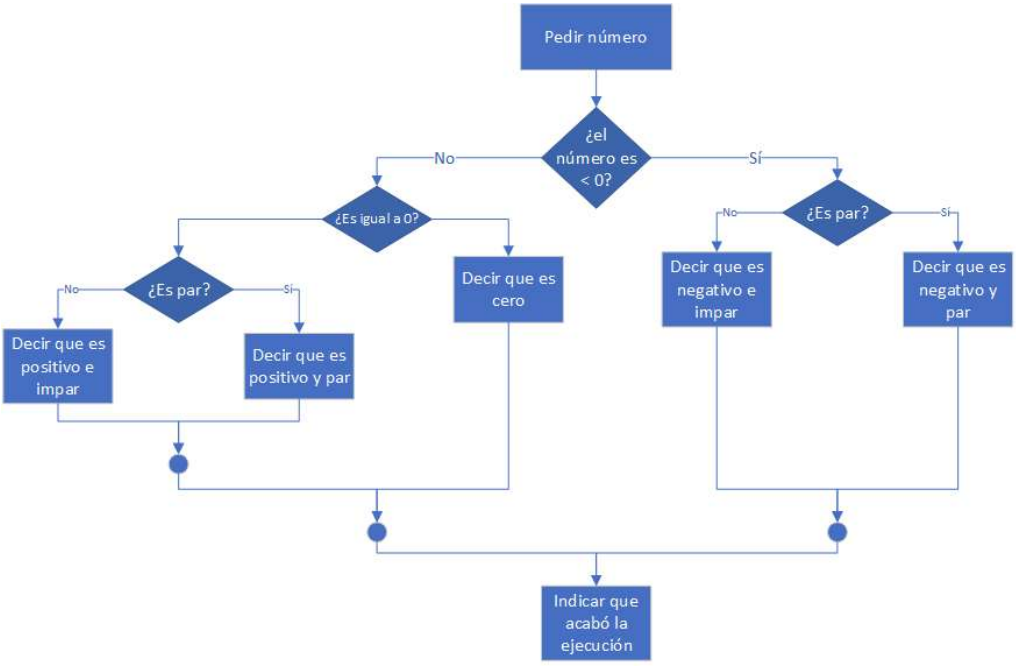
Las sentencias de control se pueden "anidar", esto significa que una (o más) de las instrucciones controladas por una sentencia de control puede ser a su vez una sentencia de control que controle una subsecuencia de instrucciones, y así sucesivamente.




```
num = int(input("Dame un número entero: "))

if num < 0:
    if num % 2 == 0:
        print(num, "es un número negativo par")
    else:
        print(num, "es un número negativo impar")
else:
    if num == 0:
        print(num, "es cero")
    else:
        if num % 2 == 0:
            print(num, "es un número positivo par")
        else:
            print(num, "es un número positivo impar")
```

```
print("Fin de la ejecución")
```



En el ejemplo, se empieza comprobando si num es menor que cero; si lo es, se evalúa si es divisible por 2 (el resto de la división es cero), mostrando un mensaje adecuado en función de la respuesta.

Si, por el contrario, el número resultase no ser menor que cero, se comprobaría a continuación si es igual a cero.

Y si al final el número resultara ser mayor que cero, se vuelve a comprobar si es par o impar. Nótese que solo se alcanzará el último de los if (que, por cierto, está en un segundo nivel de anidamiento y por tanto de sangrado) si se ha comprobado previamente que el número no es negativo ni cero, por lo que en ese punto sabemos que el número no puede sino ser positivo. Y en el último de los else sabemos además que el número no es par, por lo que se puede realizar la acción correspondiente sin más comprobaciones.

Information

Contracción elif

Cuando se encadenan una o más cláusulas `else` conteniendo solo una sentencia `if`, se pueden contraer en cláusulas **`elif`** facilitando la sintaxis, seleccionando entre múltiples alternativas excluyentes entre sí, que colocaremos en el mismo nivel de sangrado que el `if` inicial:

```
num = int(input("Dame un número entero: "))
```

```
if num < 0:
    if num % 2 == 0:
        print(num, "es un número negativo par")
    else:
        print(num, "es un número negativo impar")
elif num == 0:
    print(num, "es cero")
elif num % 2 == 0:
    print(num, "es un número positivo par")
else:
    print(num, "es un número positivo impar")

print("Fin de la ejecución")
```

Nótese que un if anidado dentro de la cláusula if no se puede abreviar de esta manera; ni tampoco si en una cláusula else a un primer if anidado siguen otras instrucciones en el mismo bloque.

Obsérvese que de la estructura if-elif··elif-else solo se podrá ejecutar uno de sus bloques de instrucciones (o incluso posiblemente ninguno si no hay un else al final). En particular, se ejecutará únicamente el primer bloque cuya condición se cumpla, o el del else si este existe y ninguna condición se cumple, reanudándose (sin comprobar las siguientes condiciones) la ejecución a continuación del último bloque.

A la hora de escribir las condiciones en los elif, hemos de ser conscientes de que para que se pueda alcanzar a comprobar una de ellas, todas las de if y elif anteriores de esa misma estructura han de haber resultado falsas, por lo que sería innecesario volver a comprobarlo. En el ejemplo, al llegar al segundo elif ya se ha comprobado anteriormente que el número no es ni negativo ni cero, de donde podemos concluir que alcanzado ese punto solo puede ser positivo, y por tanto no hace falta comprobarlo.

Question 8

Complete

Mark 1.00 out of 1.00

Observe el siguiente código:

```
a = 180

if a < 100:
    c = a
elif a < 150:
    c = 12
elif a < 200:
    c = a + 1
else:
    c = 0
```

¿Qué valor tiene la variable c al terminar la ejecución?

Answer: 181

Question 9

Complete

Mark 1.00 out of 1.00

Observe el siguiente código:

```
a = 100

if a < 100:
    c = a
elif a < 150:
    c = 12
elif a < 200:
    c = a + 1
else:
    c = 0
```

¿Qué valor tiene la variable c al terminar la ejecución?

Answer: 12