Status	Finished
Started	Friday, 22 November 2024, 11:36 AM
Completed	Friday, 22 November 2024, 11:38 AM
Duration	1 min 38 secs
Marks	5.00/5.00
Grade	**10.00** out of 10.00 (**100**%)

Information

Repetición

A veces, un algoritmo requiere repetir una acción o conjunto de acciones varias veces seguidas. Por ejemplo, supongamos que queremos que un programa muestre diez líneas con la palabra "Hola". Es fácil:

```
print("Hola")
print("Hola")
print("Hola")
print("Hola")
print("Hola")
print("Hola")
print("Hola")
print("Hola")
print("Hola")
print("Hola")
```

La repetición, o **iteración**, es bastante frecuente, por lo que los lenguajes ofrecen estructuras para expresarlas de un modo más conciso y flexible:

```
líneas_escritas = 0
líneas_a_escribir = 10
while líneas_escritas < líneas_a_escribir:
    print("Hola")
    líneas_escritas = líneas_escritas + 1
```

La sentencia ***while*** expresa la repetición de una secuencia de instrucciones mientras se cumpla una condición dada. Más precisamente, primero comprueba la condición a la derecha de la palabra clave *while*: si el resultado fuera False, salta las instrucciones del cuerpo o bloque de instrucciones (que en Python se identifican porque tienen un sangrado extra respecto de la palabra *while*), dando por terminada la ejecución de la sentencia *while*, y por tanto reanudando la ejecución a continuación; pero si el resultado fuera True, entra a ejecutar secuencialmente las instrucciones que estén contenidas en su cuerpo, y una vez que no quedaran más instrucciones por ejecutar, volvería al principio y repetiría el proceso: se evaluaría de nuevo la condición (cuyo resultado puede haber variado entretanto si lo han hecho sus operandos), y si fuera False terminaría y saltaría el bloque de instrucciones y si fuera True entraría a ejecutarlo de nuevo y a continuación de nuevo volvería al principio, etc. etc., repitiéndose el mismo mecanismo cuantas veces fuera necesario en tanto la condición no resulte False. Obsérvese que el número de iteraciones puede ser cualquiera, de 0 en adelante.

Por lo dicho anteriormente, la condición y las instrucciones relacionadas con una sentencia *while* deben estar diseñadas de tal manera que busquen que la condición deje de cumplirse oportunamente en algún intento de iteración. De lo contrario, se entraría en un **bucle infinito**.

En el ejemplo esto lo conseguimos incrementando dentro del cuerpo (y por tanto en cada iteración) la variable *líneas_escritas*, que actúa como contador de las líneas escritas; de esta forma, en algún momento su valor llegará a igualar el de la variable *líneas_a_escribir*, que define el objetivo del problema; la condición dejará entonces de cumplirse y cesará la repetición. Si, por ejemplo, hubiera código posterior al mismo nivel de la palabra *while*, este pasaría a ejecutarse a continuación.

Information

Repetición (2)

Una sentencia de repetición, como while, permite expresar la repetición de una secuencia de instrucciones de forma concisa.

```
líneas_escritas = 0
líneas_a_escribir = 10
while líneas_escritas < líneas_a_escribir:
    print("Hola")
    líneas_escritas = líneas_escritas + 1
```

Sin embargo, la mayor virtud de una sentencia de repetición no es la concisión, sino la flexibilidad que proporciona. El siguiente ejemplo es una variante del anterior en la que el objetivo de líneas a escribir no depende de un valor fijo, sino que puede variar en cada ejecución, produciendo resultados diferentes en cada caso, sin tener que modificar el algoritmo:

```
líneas_escritas = 0
líneas_a_escribir = int(input("¿Líneas a escribir? "))
while líneas_escritas < líneas_a_escribir:
    print("Hola")
    líneas_escritas = líneas_escritas + 1
```

En el caso de que el usuario introduzca un valor igual o menor que cero, no se escribirá ninguna línea, ya que la condición no se cumple desde el principio; en cualquier otro caso, se mostrarán tantas líneas como haya indicado el usuario.

Question 1

Complete

Mark 1.00 out of 1.00

¿Cuál es el número más pequeño que muestra el siguiente código?

```
num = 10

while num > 0:
    print(num)
    num = num - 3
```

Answer: 1

Question 2

Complete

Mark 1.00 out of 1.00

¿Cuál es el valor más pequeño que muestra el siguiente código?

```
num = 9

while num > 0:
    print(num)
    num = num - 3
```

Answer: 3

Information

Esquema básico de iteración

La iteración expresada por la sentencia while responde a un esquema básico bastante simple:

```
inicializaciones
```

```
while condición:
    Hacer algo (Tratamiento)
    avanzar
```

```
líneas_escritas = 0  # Inicializaciones
líneas_a_escribir = int(input("¿Líneas a escribir? "))

while líneas_escritas < líneas_a_escribir:  # Condición
    print("Hola")                           # Tratamiento
    líneas_escritas = líneas_escritas + 1   # Avance
```

Las inicializaciones, la condición, lo que se hace en cada iteración y cómo se avanza para preparar la siguiente pueden tener muchas variantes, pero siempre es posible reconocer el esquema.

Ejemplo 1:

```
current = 2
last = 15
sum_evens = 0
```

```
while current < last:
    sum_evens += current
    current += 2
```

Ejemplo 2:

```
penúltimo = 1
último = 1
actual = último
objetivo = 15
iteración = 1
```

```
while iteración <= objetivo:
    actual = último + penúltimo
    penúltimo = último
    último = actual
    iteración += 1
```

Ejemplo 3:

```
num1 = 128
num2 = 96
```

```
while num2 > 0:
    mod = num1 % num2
    num1 = num2
    num2 = mod
```

Question 3

Complete

Mark 1.00 out of 1.00

¿Qué valor muestra el siguiente código?

```
penúltimo = 0
último = 1
actual = último
objetivo = 5
iteración = 1

while iteración <= objetivo:
    actual = último + penúltimo
    penúltimo = último
    último = actual
    iteración += 1

print(actual)
```

Answer: 8

Question 4

Complete

Mark 1.00 out of 1.00

¿Qué valor muestra el siguiente código?

```
num1 = 90
num2 = 25
while num2 > 0:
    mod = num1 % num2
    num1 = num2
    num2 = mod

print(num1)
```

Answer: 5

Question 5

Complete

Mark 1.00 out of 1.00

¿Qué valor muestra el siguiente código?

```
num1 = 128
num2 = 96
while num2 > 0:
    mod = num1 % num2
    num1 = num2
    num2 = mod

print(num1)
```

Answer: 32