

Comenzado el	sábado, 20 de mayo de 2023, 14:41
Estado	Finalizado
Finalizado en	sábado, 20 de mayo de 2023, 15:33
Tiempo empleado	52 minutos 8 segundos
Puntos	15/19
Calificación	8 de 10 (79%)

Pregunta **1**

Incorrecta

Se puntúa 0 sobre 1

Cuál de las siguientes cabeceras de ciclo *for* en Python, producirá un esquema de exactamente n repeticiones, suponiendo que la variable n tiene un valor entero mayor a cero previamente asignado?

Seleccione una:

- ☐ a. `for i in range(n+1):`
- ☒ b. `for i in range(1, n):` **Incorrecto...** ¡aunque pegó en el palo! Este ciclo hace $n-1$ repeticiones: comienza la cuenta desde $i = 1$ y hace la última vuelta cuando $i = n-1$...
- ☐ c. `for i in range(n, 0, -1):`
- ☐ d. `for i in range(0, n//2):`

Incorrecto... revise la Ficha 7, página 139 y siguientes.

Pregunta **2**

Correcta

Se puntúa 1 sobre 1

¿Cuál de los siguientes scripts **NO** creará una tupla conteniendo exclusivamente los caracteres de la palabra 'Python', en ese mismo orden? (Repetimos: la pregunta es cuál de todos **NO** creará la tupla pedida...)

Seleccione una:

- ☐ a. `t1 = tuple('Python')`
- ☒ b. `Python = 'Java'` **¡Ok!**
`t5 = tuple(Python)`
- ☐ c. `t3 = ('P', 'y', 't', 'h', 'o', 'n')`
- ☐ d. `t2 = 'P', 'y', 't', 'h', 'o', 'n'`
- ☐ e. `t4 = ()`
`for c in 'Python':`
`t4 += c,`

¡Correcto!

Pregunta 3

Correcta

Se puntúa 1 sobre 1

¿Cuáles de los siguientes ciclos for mostrarán en pantalla **solamente** números negativos? (Aclaración: más de una respuesta puede ser correcta... marque TODAS las que considere válidas...)

Seleccione una o más de una:

- ☒ a. `for i in range(-10, -1):` ✓ ¡Ok!
 `print('i:', i)`
- ☒ b. `for i in range(-1, -10, -1):` ✓ ¡Ok!
 `print('i:', i)`
- ☐ c. `for i in range(10, 1, -2):`
 `print('i:', i)`
- ☐ d. `for i in range(5, -15, -5):`
 `print('i:', i)`

¡Correcto!

Pregunta 4

Correcta

Se puntúa 1 sobre 1

Analice el siguiente script en el cual se recorre una secuencia llamada `sec` con un ciclo `for`:

```
# suponga que sec es una secuencia (cadena, tupla, rango, etc.)  
# ya inicializada  
for x in sec:  
    print('x:', x)
```

¿Qué hace este script si la secuencia `sec` estuviese inicialmente vacía?

Seleccione una:

- ☒ a. El ciclo `for` no hace ninguna repetición y ✓ el script termina normalmente sin mostrar nada en pantalla. ¡Ok! Efectivamente... el ciclo `for` es de tipo `[0, N]` por lo que si la secuencia a recorrer está vacía, el `for` no llega a hacer ninguna repetición y sigue de largo a buscar la siguiente instrucción.
- ☐ b. El ciclo `for` hace una y sólo una repetición y muestra el mensaje `x: None`.
- ☐ c. El ciclo `for` no hace ninguna repetición pero el script se interrumpe lanzando un mensaje de error.
- ☐ d. El ciclo `for` entra en repetición infinita.

¡Correcto!

Pregunta **5**

Correcta

Se puntúa 1 sobre 1

¿Cuál es el significado de la sigla *IGU* (en español) en el ámbito del diseño de pantallas y sistema de carga de datos de un programa?

Seleccione una:

- ☐ a. Independencia Gráfica Universal
- ☐ b. Implementación Guiada por el Usuario
- ☒ c. Interfaz Gráfica de Usuario ✓ ¡Ok!
- ☐ d. Interfaz Genuina de Usuario

¡Correcto!

Pregunta **6**

Correcta

Se puntúa 2 sobre 2

El proceso de búsqueda del mayor en el siguiente programa, contiene una ligera modificación a las variantes analizadas en la Ficha 7: la variable *may* se inicializa en 0 antes del ciclo, y luego simplemente se aplica la idea de comparar el número cargado *num* contra *may* en cada vuelta, sin preocuparse por el valor de *may* frente al primer dato. ¿Hay algún problema en el planteo de esta estrategia?

```
__author__ = 'Catedra de AED'

print('Determinacion del mayor de una sucesion (variante: may inicializada en 0)...')
n = int(input('Ingrese n: '))

may = 0
for i in range(1, n+1):
    num = int(input('Numero: '))
    if num > may:
        may = num

print('El mayor es:', may)
```

Seleccione una:

- ☐ a. Sí, hay un problema: si todos los números del conjunto de entrada fuesen cero, se mostraría un *None* como resultado, en lugar de lo que correspondería retornar que es un 0.
- ☐ b. Sí, hay un problema: si todos los números del conjunto de entrada fuesen positivos o cero, se mostraría un 0 como resultado, en lugar del mayor del conjunto cargado.
- ☒ c. Sí, hay un problema: si todos los números del conjunto de entrada fuesen negativos, se mostraría un 0 como resultado, en lugar del mayor de los negativos cargados. ✓ ¡Correcto!
- ☐ d. No. No hay ningún problema. Funcionará correctamente en todos los casos.

¡Correcto!

Pregunta 7

Correcta

Se puntúa 3 sobre 3

El proceso de búsqueda del mayor en el siguiente programa está ligeramente cambiado con relación a su versión original (la tercera variante del problema de la búsqueda del mayor (Ficha 7): En la versión original, se comenzaba definiendo la variable *may* con el valor *None*, y ahora en esta versión que proponemos, hemos eliminado esa inicialización **marcándola como un comentario** (y sin borrarla, pero sólo por razones de claridad). ¿Hay algún problema en el planteo de esta pequeña variante? (Recuerde: si la instrucción está marcada como comentario, es lo mismo que si no estuviese).

```
__author__ = 'Catedra de AED'

print('Determinacion del mayor de una sucesion (variante 3)...')

# may = None
b = False
num = int(input('Ingrese un numero (con 0 finaliza): '))
while num != 0:
    if b == False:
        may = num
        b = True
    elif num > may:
        may = num
    num = int(input('Ingrese otro (con 0 finaliza): '))

print('El mayor es:', may)
```

Seleccione una:

- ☐ a. Sí, hay un problema: la variable *may* estará sin definir incluso dentro del ciclo. Cuando se intente asignar en ella el valor de *num* que corresponda a cada vuelta, el programa se interrumpirá y lanzará un error de variable no definida.
- ☒ b. Sí, hay un problema: si en la primera carga antes del ciclo se ingresa un 0, el ciclo no ejecutará su bloque de acciones y la variable *may* quedará sin definir, provocando en ese caso que el programa se interrumpa y lance un error al intentar mostrar el valor de *may* en el programa. ✔ ¡Correcto!
- ☐ c. Sí, hay un problema: si todos los números que cargan fuesen negativos, en ese caso la variable *may* por defecto quedaría valiendo 0, y el programa mostraría un cero al final.
- ☐ d. No. No hay ningún problema. Funcionará correctamente en todos los casos, y la instrucción *may = None* era completamente innecesaria en la versión original.

¡Correcto!

Pregunta 8

Correcta

Se puntúa 3 sobre 3

El proceso de búsqueda del mayor en el siguiente programa, contiene una modificación de la primera variante analizada en la Ficha 7: En ambas se usa un ciclo for que ejecute n repeticiones para cargar los n números. Pero en la versión original de la Ficha, el ciclo se ajusta como `for i in range(1, n+1)` mientras que ahora proponemos `for i in range(n)` para intentar abreviar, basándonos en la idea de que en ambos casos el ciclo hará efectivamente n repeticiones. ¿Hay algún problema con este cambio propuesto?

```
__author__ = 'Catedra de AED'

print('Determinacion del mayor de una sucesion (for cambiado)...')

n = int(input('Numero: '))
for i in range(n):
    num = int(input('Numero: '))
    if i == 1:
        may = num
    elif num > may:
        may = num

print('El mayor es:', may)
```

Seleccione una:

- ☒ a. Sí, hay un problema: el rango generado con `range(n)` contiene efectivamente n números y el ciclo hará n repeticiones, pero el primer valor del rango será 0 (y no 1) con lo cual al preguntar si `i == 1` en la primera vuelta del ciclo se obtendrá un falso. La variable `may` quedará entonces sin definir, y en esa misma vuelta el programa se interrumpirá al comparar `num` con `may`. ¡Correcto!
- ☐ b. Sí, hay un problema: el rango generado con `range(n)` contiene efectivamente n números y el ciclo hará n repeticiones, pero el primer valor del rango será 0 (y no 1). Por lo tanto, la condición `i == 1` sólo verdadera en la segunda vuelta del ciclo y la variable `may` quedará inicializada con el segundo número en lugar del primero. El programa entonces, ignorará el primer número que se cargue, y mostrará el mayor de la secuencia pero sin incluir al primero.
- ☐ c. Sí, hay un problema: el rango generado con `range(n)` no contiene efectivamente n números, sino $n-1$ números y el ciclo hará entonces una repetición menos de las esperadas.
- ☐ d. No. No hay ningún problema. El funcionamiento es exactamente igual al de la versión original.

¡Correcto!

Pregunta 9

Correcta

Se puntúa 3 sobre 3

Supongamos que se nos pide buscar el mayor de una secuencia de n números, tal como se analizó en la Ficha 7, pero de modo que ahora además de mostrar el mayor, se muestre también en qué orden apareció en la carga (o sea, en qué vuelta del ciclo apareció ese mayor) Por ejemplo, si la secuencia a cargar fuese {2, 5, 1, 3} entonces el mayor sería el 5 y apareció en el lugar 2 (o en el orden 2, o en la vuelta 2 del ciclo) ¿Está bien planteado el siguiente programa para cumplir con este nuevo requerimiento?


```
__author__ = 'Catedra de AED'

print('Determinacion del mayor de una sucesion (variante 1)...')

n = int(input('n: '))
for v in range(1, n+1):
    num = int(input('Numero: '))
    if v == 1:
        may, pos = num, 1
    elif num > may:
        may, pos = num, v

print('El mayor es:', may)
print('Se cargó en la vuelta:', pos)
```

Seleccione una:

- ☐ a. No. No está bien planteado: siempre muestra que la vuelta en que se cargó el mayor fue la 1 del ciclo.
- ☒ b. Sí, está correctamente planteado.  ¡Correcto!
- ☐ c. No. No está bien planteado: los resultados se están mostrando al revés, ya que el mayor estaría en *pos* y su posición de carga estaría en *may*. Hay que invertir las asignaciones de las tuplas dentro del ciclo.
- ☐ d. No. No está bien planteado: si el valor mayor apareciese repetido más de una vez en la carga, la variable *pos* quedaría valiendo *None*.

¡Correcto!

Pregunta **10**

Incorrecta

Se puntúa 0 sobre 3

El siguiente programa es una variante del que mostramos en la [Ficha 7](#) para resolver el problema 20. En esta variante, hemos modificado la lógica para que al buscar el menor importe del vendedor 2 no se tenga que preguntar por el primer dato en cada vuelta, y eliminar la bandera que se aplicaba en la versión original. ¿Hay algún problema en el planteo de esta variante? *Seleccione la respuesta que mejor describa lo que está pasando con el programa propuesto.*

```
__author__ = 'Catedra de AED'

# pasó la primera venta del vendedor 2?
aviso = False

# si no se cargan ventas del vendedor 2, menor_importe queda en None...
menor_importe = None

# acumuladores de cantidades...
c1 = c2 = 0

# acumuladores de importes...
i1 = i2 = 0

print('Ventas de un Comercio... ingrese los datos de cada venta...')

# ingresar datos de la primera venta...
codigo = -1
while codigo < 0 or codigo > 2:
    codigo = int(input('Codigo de vendedor (1 o 2) (0 para cortar): '))
    if codigo > 2 or codigo < 0:
        print('Error... se pidio 1 o 2 o 0 para cortar...')

cantidad = int(input('Cantidad vendida: '))
importe = float(input('Importe: '))

menor_importe = importe

while codigo != 0:
    if codigo == 1:
        c1 += cantidad
        i1 += importe

    elif codigo == 2:
        c2 += cantidad
        i2 += importe

    # Aplicar mecanismo de cálculo del menor...
    if importe < menor_importe:
        menor_importe = importe

    # ingresar el siguiente codigo y volver al ciclo...
    codigo = -1
    while codigo < 0 or codigo > 2:
        codigo = int(input('Codigo de vendedor (1 o 2) (0 para cortar): '))
        if codigo > 2 or codigo < 0:
            print('Error... se pidio 1 o 2 o 0 para cortar...')

    cantidad = int(input('Cantidad vendida: '))
    importe = float(input('Importe: '))

# Calcular el importe promedio...
promedio = (i1 + i2) / 2

print('Cantidad de productos vendida por el vendedor 1:', c1)
print('Cantidad de productos vendida por el vendedor 2:', c2)
print('Importe total facturado por el vendedor 1:', i1)
print('Importe total facturado por el vendedor 2:', i2)
print('Importe de la menor venta del vendedor 2:', menor_importe)
print('Importe promedio entre los dos vendedores:', promedio)
```

Seleccione una:

- ☐ a. No. No hay ningún problema. Funcionará correctamente en todos los casos.
- ☐ b. Sí. Hay un único problema: se pedía el menor importe del vendedor 2, pero antes del ciclo la variable *menor_importe* se inicializa con el primer importe cargado, lo que sería un error si el código del vendedor en esa carga fuese el 1 y no el 2.
- ☐ c. Sí. De hecho, hay dos problemas: el primero es que los tres datos de la venta se cargan juntos. Si el código del vendedor en ese momento fuese 0, el ciclo debería cortar pero de todos modos el programa pide al usuario los otros dos datos... que no tendrán ningún sentido. Y el segundo problema (y más grave) es que se pedía el menor importe del vendedor 2, pero antes del ciclo la variable *menor_importe* se inicializa con el primer importe cargado, lo que sería un error si el código del vendedor en esa carga fuese el 1 y no el 2.
- ☒ d. Sí. Hay un único problema: los tres datos de la venta se cargan juntos. Si el código del vendedor en ese momento fuese 0, el ciclo debería cortar pero de todos modos el programa pide al usuario los otros dos datos... que no tendrán ningún sentido.

Incorrecto... el problema de los tres datos cargados juntos es efectivamente un inconveniente del programa mostrado... pero no es el único... hay al menos otro, más grave...

Revise la **Ficha 7**, página 148 y siguientes, y realice un análisis detallado del nuevo programa...

◀ [Ficha Opcional 07 \[Ciclo de Control de Eventos\] \(PDF - para lectura directa\)](#)

Ir a...



[Guía de Ejercicios Prácticos - Ficha 07 ▶](#)