

Comenzado el	domingo, 25 de junio de 2023, 13:21
Estado	Finalizado
Finalizado en	domingo, 25 de junio de 2023, 15:45
Tiempo empleado	2 horas 23 minutos
Puntos	18/21
Calificación	8 de 10 (84%)

Pregunta **1**

Correcta

Se puntúa 2 sobre 2

Suponga que se desea un programa que cargue dos números, y muestre el producto y el cociente entre ellos ¿Cuál es el problema con el siguiente programa desarrollado en base a funciones que manejan variables globales?

```
__author__ = 'Cátedra de AED'


def cargar():
    global a, b
    a = int(input('A: '))
    b = int(input('B: '))

def calcular():
    c = a * b
    d = a / b

# script principal...
cargar()
calcular()

print('Producto:', c)
print('Cociente:', d)
```

Seleccione una:

- ☐ a. No lanza errores de intérprete, pero funciona mal: en el *script principal* están al revés las invocaciones a las funciones *cargar()* y *calcular()*: primero debería invocar a *calcular()* y luego a *cargar()*.
- ☐ b. Lanza un error de intérprete en la carga de datos: la carga de datos DEBE hacerse en el script principal y no en una función separada.
- ☒ c. El programa lanza un error al ser ejecutado, en el *script principal*: no reconoce las variables *c* y *d* ya que fueron definidas en la  función *calcular()*, pero como variables locales a ella.
- ☐ d. Lanza un error de intérprete en el mismo inicio: todas las variables debieron ser definidas (asignadas) en el script principal.

Pregunta **2**

Correcta

Se puntúa 2 sobre 2

El siguiente programa utiliza variables definidas como globales. ¿Qué hace exactamente el programa (y por qué hace eso)?

```
__author__ = 'Catedra de AED'

def funcion01():
    a = b * 2

def funcion02():
    a = b + c

def test():
    global a, b, c

    a, b, c = 10, 20, 30
    funcion01()
    funcion02()
    print('Valor final de a:', a)

# script principal
test()
```

Seleccione una:

- ☐ a. Muestra el mensaje: *Valor final de a: 40* - Porque la variable *a* queda valiendo el valor asignado al invocar a *funcion01()*.
- ☒ b. Muestra el mensaje: *Valor final de a: 10* - Porque la variable *a* cuyo valor se muestra en *test()*, es *global* y queda valiendo el valor 10 asignado en la misma *test()*. ✓
- ☐ c. Muestra el mensaje: *Valor final de a: 50* - Porque la variable *a* queda valiendo el valor asignado al invocar a *funcion02()*.
- ☐ d. Lanza un error de intérprete al intentar ejecutarlo: una función no puede declarar variables locales con el mismo nombre que una variable definida como global.

Pregunta **3**

Correcta

Se puntúa 2 sobre 2

Suponga que se desea desarrollar un programa básico en Python que cargue dos números y los muestre ordenados de menor a mayor. El programa que se muestra a continuación tiene ese objetivo. ¿Hay algún problema con el programa mostrado? (tenga cuidado con el concepto de variable local vs. el concepto de variable global).

```
__author__ = 'Cátedra de AED'

def ordenar():
    if a > b:
        men = b
        may = a
    else:
        men = a
        may = b

# script principal...
men, may = 0, 0

a = int(input('Primer número: '))
b = int(input('Segundo número: '))

ordenar ()

print('Menor:', men)
print('Mayor:', may)
```

Seleccione una:

- ☐ a. No hay ningún problema. Muestra los números ordenados correctamente.
- ☒ b. El programa compila y ejecuta, pero muestra resultados incorrectos: se carguen los datos que se carguen, siempre muestra que el mayor y el menor valen 0. ✓
- ☐ c. El programa lanza un error de intérprete: las variables *men* y *may* se definieron dos veces en dos ámbitos distintos, y eso no puede hacerse en Python.
- ☐ d. El programa lanza un error de intérprete: la función *ordenar()* debió ser declarado debajo del script principal.

Pregunta **4**

Correcta

Se puntúa 1 sobre 1

La siguiente función está inspirada en un subproblema de la Ficha 5, pero de forma que ahora se plantea como una función en la cual todas sus variables se definen como globales y contiene además uno o más errores en su escritura. Identifique cuál o cuáles son esos errores (como puede haber más de uno, entonces más de una respuesta podría ser válida... marque todas las que considere aplicables):

```
def ordenar_dos:  
    global a, b, men, may  
    if a > b:  
        men = b  
        may = a  
    else:  
        men = a  
        may = b
```

Seleccione una o más de una:

- ☐ a. La declaración *global* debe ir en la misma línea que la cabecera.
- ☒ b. Está mal indentado el bloque de acciones de la función: debería encolumnarse hacia la derecha del inicio de la cabecera. ✓
- ☐ c. No corresponde colocar el símbolo *dos puntos (:)* al finalizar la cabecera.
- ☒ d. Faltan los paréntesis vacíos en la cabecera de la función. ✓

Pregunta **5**

Correcta

Se puntúa 2 sobre 2

Considere el problema del cálculo de los montos a pagar por el viaje de estudios de tres cursos de secundaria, tal como se presentó en la Ficha 10. El programa que se muestra más abajo, está replanteado de forma de usar variables globales en lugar de parámetros y retornos, y la función *mayor()* (que determinaba cuál era el curso con más cantidad de alumnos, y además, cuál era el monto a pagar por ese curso), está definida de la siguiente forma:

```

__author__ = 'Cátedra de AED'

def mayor():
    global may, may_cur
    if c1 > c2 and c1 > c3:
        may = m1
        may_cur = 'Primero'
    else:
        if c2 > c3:
            may = m2;
            may_cur = 'Segundo'
        else:
            may = m3
            may_cur = 'Tercero'

def montos():
    global m1, m2, m3
    m1 = c1 * 1360
    m2 = c2 * 1360
    m3 = c3 * 1360

    if c1 > 40:
        m1 = m1 - m1/100*5

    if c2 > 40:
        m2 = m2 - m2/100*5

    if c3 > 40:
        m3 = m3 - m3/100*5

def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = may / mtot * 100
    else:
        porc = 0

def test():
    global c1, c2, c3

    # título general y carga de datos...
    print('Cálculo de los montos de un viaje de estudios...')
    c1 = int(input('Ingrese la cantidad de alumnos del primer curso: '))
    c2 = int(input('Ingrese la cantidad de alumnos del segundo curso: '))
    c3 = int(input('Ingrese la cantidad de alumnos del tercer curso: '))

    # procesos... invocar a las funciones en el orden correcto...
    montos()
    mayor()
    porcentaje()

    # visualización de resultados
    print('El curso mas numeroso es el', may_cur)
    print('El monto del viaje del primer curso es:', m1)
    print('El monto del viaje del segundo curso es:', m2)
    print('El monto del viaje del segundo curso es:', m3)
    print('El porcentaje del monto del mas numeroso en el total es:', porc)


# script principal: sólo invocar a test()...
test()

```

¿Qué efecto causaría en el programa que la función *mayor()* fuese reemplazada por esta otra versión que se muestra a continuación, y qué cambios debería hacer el programador para que su programa siga funcionando y cumpliendo con los mismos requerimientos? (Más de una respuesta puede ser válida. Marque todas las que considere correctas):

```
def mayor():
    global salida
    if c1 > c2 and c1 > c3:
        salida = 'Primero', m1
    else:
        if c2 > c3:
            salida = 'Segundo', m2
        else:
            salida = 'Tercero', m3
```

Seleccione una o más de una:

- ☒ a. En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un  cambio que debe hacer el programador es en la visualización de resultados dentro de la función *test()*, que debería verse así:


```
# función test()... visualización de resultados...
print('El curso mas numeroso es el', salida[0])
print('El monto del viaje del primer curso es:', m1)
print('El monto del viaje del segundo curso es:', m2)
print('El monto del viaje del segundo curso es:', m3)
print('El porcentaje del monto del mas numeroso en el total es:', porc)
```

- ☐ b. En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la visualización de resultados dentro de la función *test()*, que debería verse así:

```
# función test()... visualización de resultados...
print('El curso mas numeroso es el', salida[1])
print('El monto del viaje del primer curso es:', m1)
print('El monto del viaje del segundo curso es:', m2)
print('El monto del viaje del segundo curso es:', m3)
print('El porcentaje del monto del mas numeroso en el total es:', porc)
```

- ☐ c. En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la función *porcentaje()*, que debería verse así:

```
def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = salida[0] / mtot * 100
    else:
        porc = 0
```

- ☒ d. En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un  cambio que debe hacer el programador es en la función *porcentaje()*, que debería verse así:

```
def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = salida[1] / mtot * 100
    else:
        porc = 0
```

Pregunta 6

Correcta

Se puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones son CIERTAS respecto del concepto de *reutilización de piezas de software* en programación?
[Aclaración: varias respuestas pueden ser válidas. Marque **todas** las que considere correctas]

Seleccione una o más de una:

- ☒ a. La reutilización permite hacer el trabajo en forma más eficiente, ordenada y profesional a un programador. ✓
- ☐ b. La reutilización es el proceso por el cual un mismo segmento de código fuente se copia y se pega en cuanto lugar el programador necesite volver a contar con dicho segmento.
- ☒ c. La reutilización es el proceso por el cual una misma función o módulo ya desarrollado para un programa, vuelve a usarse en otro programa en forma directa, por simple mecanismo de invocación y sin tener que modificar el nuevo programa o la propia función/módulo. ✓
- ☒ d. Los lenguajes de programación proveen grandes cantidades de funciones reutilizables en sus librerías de funciones predefinidas o estándar. ✓

Pregunta 7

Correcta

Se puntúa 2 sobre 2

En la columna de la izquierda se muestran diversos planteos posibles para una función que debe calcular el promedio de todos los números enteros contenidos en el intervalo [1, n]. Algunas (o todas) presentan inconvenientes en cuanto sus posibilidades de reutilización. Para cada versión, seleccione desde la lista de la columna derecha la opción que mejor describe su problema o planteo:

```
def promedio(n):  
    ac = 0  
    for x in range(1, n+1):  
        ac += x  
    p = ac / n  
    return p
```

Planteo genérico y reutilizable



```
def promedio(n):  
    global p  
    ac = 0  
    for x in range(1, n+1):  
        ac += x  
    p = ac / n
```

Dependencia externa en relación a los resultados generados por la función



```
def promedio():  
    n = int(input('N: '))  
    ac = 0  
    for x in range(1, n+1):  
        ac += x  
    p = ac / n  
    print('Promedio:', p)
```

Acoplamiento procesos/interfaz respecto de los datos y los resultados que debe gestionar la función



```
def promedio():  
    n = int(input('N: '))  
    ac = 0  
    for x in range(1, n+1):  
        ac += x  
    p = ac / n  
    return p
```

Acoplamiento procesos/interfaz respecto de los datos que debe procesar la función





Pregunta **8**

Correcta

Se puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones son verdaderas respecto de la relación entre *Programación Estructurada* y *Programación Modular*? (Más de una respuesta puede ser cierta. Marque todas las que considere correctas)

Seleccione una o más de una:

- ☐ a. La Programación Estructurada y la Programación Modular son dos paradigmas de programación diferentes, sin conexión entre ellos.
- ☒ b. La Programación Modular enuncia a la Programación Estructurada como uno de sus principios de trabajo, pero no se limita a  ella.
- ☐ c. Los conceptos de Programación Estructurada y Programación Modular son equivalentes.
- ☒ d. La Programación Estructurada enuncia a la Programación Modular como uno de sus principios de trabajo, pero no se limita a  ella.





Pregunta **9**

Correcta

Se puntúa 1 sobre 1

¿Cuáles de los siguientes elementos favorecen la *reutilización de una función*? [Aclaración: varias respuestas pueden ser válidas. Marque **todas** las que considere correctas]

Seleccione una o más de una:

- ☒ a. La independencia de la función en referencia a procesos de carga de datos y visualización de resultados. 
- ☒ b. La independencia de la función en relación a elementos definidos fuera de ella. 
- ☒ c. La declaración de la función de forma que acepte parámetros para gestionar sus datos, y emplee el mecanismo de retorno para  devolver sus resultados.
- ☒ d. La correcta escritura de la función apegándose a principios, reglas y consejos de buenas prácticas (usar nombres descriptivos,  emplear comentarios, dejar espacios en blanco donde contribuya a una mejor lectura, etc).

Pregunta **10**

Correcta

Se puntúa 1 sobre 1

¿Cuál de las siguientes funciones creará una tupla conteniendo exclusivamente los números pares divisibles por 3 y por 7 al mismo tiempo, del intervalo $[0, n]$?

Seleccione una:

☐ a.

```
def prueba(n):  
    r = ()  
    for i in range(2, 3*(n+1), 7):  
        r += i,  
    return r
```

☒ b.

```
def prueba(n):  
    r = ()  
    for i in range(0, n+1, 42):  
        r += i,  
    return r
```

 ✓

☐ c.

```
def prueba(n):  
    r = ()  
    for i in range(1, n+1, 21):  
        r += i,  
    return r
```

☐ d.

```
def prueba(n):  
    r = ()  
    for i in range(0, n+1, 21):  
        r += i,  
    return r
```

Pregunta **11**

Correcta

Se puntúa 1 sobre 1

¿Cuál de las siguientes funciones creará una tupla conteniendo exclusivamente los números divisibles por 4 y por 5 al mismo tiempo, del intervalo [1, n]?

Seleccione una:

- ☐ a.

```
def prueba(n):  
    r = ()  
    for i in range(0, n+1, 4):  
        r += i,  
    return r
```
- ☐ b.

```
def prueba(n):  
    r = ()  
    for i in range(4, n+1, 5):  
        r += i,  
    return r
```
- ☐ c.

```
def prueba(n):  
    r = ()  
    for i in range(0, n+1, 5):  
        r += i,  
    return r
```
- ☒ d.

```
def prueba(n):  
    r = ()  
    for i in range(0, n+1, 20):  
        r += i,  
    return r
```

 ✓

Pregunta **12**

Parcialmente correcta

Se puntúa 1 sobre 1

¿Cuál es el motivo por el cual el *acoplamiento entre procesos e interfaz de usuario* es un problema en cuanto a las posibilidades de reuso de una función? [Más de una opción puede ser válida... marque **todas** las que considere correctas]

Seleccione una o más de una:

- ☐ a. En la medida de lo posible, el acoplamiento entre procesos e interfaz de usuario debería ser evitado, pero en algún momento el programador necesitará funciones que hagan el trabajo de gestionar la interfaz con el usuario. Tendrá entonces ciertas funciones con objetivos de control de interfaz, y ciertas otras funciones genéricas y reutilizables.
- ☒ b. Si la función procede a ingresar desde la interfaz de usuario el valor de alguna variable, pero el programador no necesitaba esa carga debido a que sus variables venían con valores definidos previamente, entonces deberá modificar la función para adaptarla al funcionamiento de su programa. Tendrá un problema similar si la función despliega en pantalla algún resultado. ✓
- ☒ c. Si en la función se supone que la interfaz de usuario estará soportada en la consola estándar, y luego se pretendiese reusarla pero en un contexto de ventanas y componentes visuales de alto nivel, entonces la función deberá ser rediseñada completamente. ✓
- ☐ d. El concepto de acoplamiento entre procesos e interfaz de usuario no es aplicable a funciones, sino a diagramas de flujo y pseudocódigos. Por lo tanto, la pregunta en sí misma carece de sentido.

Pregunta 13

Correcta

Se puntúa 1 sobre 1

¿Cuál es el motivo por el cual la **dependencia externa** es un problema en cuanto a las posibilidades de reuso de una función?

Seleccione una:

- ☒ a. Si la función utiliza elementos definidos fuera de ella, entonces los programas donde se quiera usar esa función deberán hacer declaraciones especiales para poder usarla, y/o se deberá modificar la propia función ✓
- ☐ b. El concepto de dependencia externa no es aplicable a funciones, sino a ciclos. Por lo tanto, la pregunta en sí misma carece de sentido.
- ☐ c. Una función puede tener dependencia externa, pero esa dependencia externa no es un problema en cuanto a la reutilización de una función.
- ☐ d. Es peor que eso: si la función utiliza elementos definidos fuera de ella, no hay forma alguna de poder usarla en ningún programa: no sólo no es reutilizable, sino que simplemente no puede usarse.

Pregunta 14

Incorrecta

Se puntúa 0 sobre 3

En la columna de la izquierda se muestran diversas situaciones que requieren algún tipo de análisis y cálculo combinatorio. Seleccione de la lista de la columna de la derecha el tipo de cálculo que correspondería aplicar para cada caso.

Calcular cuántos arreglos diferentes de $m = 3$ letras se pueden formar con las $n = 6$ letras de la palabra 'Python'. ✗

Principio básico de conteo [$t = e_1 * e_2 * e_3 * \dots * e_k$]

Calcular en cuántas formas podría quedar organizado el orden de mérito final de los $m = 5$ cargos a cubrir de un concurso docente en el que participan $n = 12$ posibles candidatos. ✗

Combinaciones de n elementos tomados de m , sin reposición [$c(n, m) = n! / (m! * (n - m)!)$]

Calcular cuántos grupos diferentes de $m = 4$ colores, pueden formarse a partir de un conjunto de $n = 7$ colores, admitiendo colores repetidos. ✗

Permutaciones de n elementos tomados de m , con reposición [$pr(n, m) = \text{pow}(n, m)$]

Calcular en cuántas formas podrían organizarse equipos de $m = 4$ participantes de un evento de búsqueda del tesoro, sabiendo que hay $n = 28$ inscriptos en total. ✗

Combinaciones de n elementos tomados de m , con reposición [$cr(n, m) = (m + (n - 1))! / (m! * (n - 1)!)$]

Calcular cuántas variantes pueden formarse con las letras y los dígitos de la patente de un automóvil en Argentina (tres letras del alfabeto español, sin la ñ, 26 letras en total) y tres dígitos al final. ✗

Permutaciones de n elementos tomados de m , sin reposición [$p(n, m) = n! / (n - m)!]$]

Ir a...

