

<b>Comenzado el</b>	domingo, 11 de junio de 2023, 20:33
<b>Estado</b>	Finalizado
<b>Finalizado en</b>	domingo, 11 de junio de 2023, 21:44
<b>Tiempo empleado</b>	1 hora 11 minutos
<b>Calificación</b>	10 de 10 (100%)

Pregunta **1**

Correcta

Se puntúa 1 sobre 1

¿Cuál de las siguientes es claramente **FALSA** respecto de la estrategia de dividir un problema en subproblemas?

Seleccione una:

- ☐ a. La división en subproblemas permite dividir el trabajo entre varios programadores.
- ☐ b. La división en subproblemas facilita la comprensión del problema por parte del programador.
- ☒ c. La división en subproblemas garantiza que se encontrará una solución para el problema en estudio. ¡Ok! Lamentablemente, nada garantiza que encontremos una solución para cualquier problema que se nos plantee...
- ☐ d. La división en subproblemas permite generar subrutinas que luego podrán usarse nuevamente en otros planteos.

¡Correcto!

La respuesta correcta es:

La división en subproblemas garantiza que se encontrará una solución para el problema en estudio.



Pregunta **2**

Correcta

Se puntúa 1 sobre 1

¿Cuáles de las siguientes son ciertas en relación al concepto general de *función*? (Más de una puede ser cierta, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

- ☐ a. Una función es un segmento de programa escrito por separado, pero que solo es aplicable a procesos que impliquen mostrar resultados en pantalla.
- ☒ b. Una función es una subrutina: un segmento de programa que se escribe en forma separada y al cual se asocia un  **¡Ok!** identificador para poder activarla.
- ☒ c. En general, los datos que una función recibe son tomados en los *parámetros* que listan entre los paréntesis de su  **¡Ok!** cabecera, y los resultados que la función genera son devueltos mediante el mecanismo de retorno con la instrucción *return*.
- ☐ d. Una función es un segmento de programa escrito por separado, pero que solo es aplicable a cálculos matemáticos.

¡Correcto!

Las respuestas correctas son:

Una función es una subrutina: un segmento de programa que se escribe en forma separada y al cual se asocia un identificador para poder activarla.,

En general, los datos que una función recibe son tomados en los *parámetros* que listan entre los paréntesis de su cabecera, y los resultados que la función genera son devueltos mediante el mecanismo de retorno con la instrucción *return*.


Pregunta **3**

Correcta

Se puntúa 1 sobre 1

¿Qué significa decir que una función puede entenderse como una *caja negra*?

Seleccione una:

- ☐ a. Significa que los procesos que lleva a cabo la función siempre son incomprensibles para quien usa la función desde el exterior (sin importar si esos procesos son visibles o no desde el exterior).
- ☐ b. Significa que los procesos que lleva a cabo la función son visibles y evidentes para quien usa la función desde el exterior.
- ☐ c. No significa nada porque la designación de *caja negra* para referirse a una función no existe.
- ☒ d. Significa que los procesos que lleva a cabo la función permanecen ocultos para quien usa la función desde el exterior.  **¡Ok!**

¡Correcto!

La respuesta correcta es:

Significa que los procesos que lleva a cabo la función permanecen ocultos para quien usa la función desde el exterior.

Pregunta **4**

Correcta

Se puntúa 1 sobre 1

¿Qué hace el siguiente programa en Python?

```
__author__ = 'Cátedra de AED'

def comparar():
    if a > p:
        print('El valor', a, 'es mayor...' )

    if b > p:
        print('El valor', b, 'es mayor...')

    if c > p:
        print('El valor', c, 'es mayor...')


def promedio():
    global p
    p = (a + b + c) / 3

a = int(input('A: '))
b = int(input('B: '))
c = int(input('C: '))

promedio()
comparar()

print('Programa terminado...')
```

Seleccione una:

- ☐ a. Calcula el promedio de los valores de *a*, *b* y *c*, y luego muestra los valores de todas las variables.
- ☐ b. Lanza error de intérprete: las variables *a*, *b*, *c* y *p* están definidas incorrectamente (debieron definirse todas en el script principal, debajo de las funciones).
- ☒ c. Calcula el promedio de los valores de *a*, *b* y *c*, y luego muestra sólo los valores de de las variables que sean mayores al  ¡Ok!
- ☐ d. Muestra los valores de *a*, *b*, *c*, y también el *promedio* de los valores de esas tres variables.

¡Correcto!

La respuesta correcta es:

Calcula el promedio de los valores de *a*, *b* y *c*, y luego muestra sólo los valores de de las variables que sean mayores al promedio.

## Pregunta 5

Correcta

Se puntúa 1 sobre 1

En la columna de la izquierda se describen algunas situaciones de programación que necesitan ser resueltas mediante el desarrollo de alguna función, y se muestra el bloque de acciones de las posibles funciones, sin sus cabeceras. Y en la columna de la derecha, figuran las posibles cabeceras de esas funciones. Para cada situación, seleccione la cabecera que sería adecuada.

Tomar como parámetro dos números y un valor boolean. Si el boolean es true, calcular y retornar el cociente de los dos números. Si el boolean es false, retornar el producto de ambos números:

# Cabecera??

```
if flag:
    r = x / y
else:
    r = x * y
return r
```

def proceso(x, y, flag): ↕



Tomar como parámetro las edades de tres personas, y retornar la menor edad:

# Cabecera??

```
if e1 < e2 and e1 < e3:
    m = e1
else:
    if e2 < e3:
        m = e2
    else:
        m = e3
return m
```

def menor(e1, e2, e3): ↕



Tomar como parámetro los valores de dos temperaturas, y retornar un valor lógico que indique si ambas son mayores a cero:

# Cabecera??

```
if t1 > 0 and t2 > 0:
    r = True
else:
    r = False
return r
```

def chequear(t1, t2): ↕



Tomar como parámetro una cadena de caracteres y dos números enteros. Retornar los dos caracteres de la cadena que estén ubicados en las posiciones indicadas por los dos números:

# Cabecera??

```
return s[i1], s[i2]
```

def valores(s, i1, i2): ↕



¡Ok!

La respuesta correcta es:

Tomar como parámetro dos números y un valor boolean. Si el boolean es true, calcular y retornar el cociente de los dos números. Si el boolean es false, retornar el producto de ambos números:

# Cabecera??

```
if flag:
    r = x / y
else:
    r = x * y
return r
```

→ def proceso(x, y, flag);

Tomar como parámetro las edades de tres personas, y retornar la menor edad:

# Cabecera??

```
if e1 < e2 and e1 < e3:
    m = e1
else:
    if e2 < e3:
        m = e2
    else:
        m = e3
return m
```

→ def menor(e1, e2, e3);

Tomar como parámetro los valores de dos temperaturas, y retornar un valor lógico que indique si ambas son mayores a cero:

# Cabecera??

```
if t1 > 0 and t2 > 0:  
    r = True  
else:  
    r = False  
return r
```

→ def chequear(t1, t2);

Tomar como parámetro una cadena de caracteres y dos números enteros. Retornar los dos caracteres de la cadena que estén ubicados en las posiciones indicadas por los dos números:

# Cabecera??

```
return s[i1], s[i2]
```

→ def valores(s, i1, i2 )

Pregunta **6**

Correcta

Se puntúa 1 sobre 1

Suponga que se desea desarrollar un programa que cargue dos números, y muestre el mayor de los números pero también el valor de multiplicar por dos y por tres a ese mayor. ¿Está bien planteado el programa que sigue?

```
__author__ = 'Cátedra de AED'

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


def comparar(a, b):
    if a > b:
        may = a
    else:
        may = b
    return may

def procesar(may):
    doble = 2 * may
    triple = 3 * may
    return doble, triple

def mostrar(may, doble, triple):
    print('El mayor es:', may)
    print('El doble del mayor es:', doble)
    print('El triple del mayor es:', triple)

# script principal...
a, b = cargar()
doble, triple = procesar(may)
may = comparar(a, b)
mostrar(may, doble, triple)
```

Seleccione una:

- ☐ a. Sí. El programa está correctamente planteado.
- ☒ b. El programa está mal planteado: la función *comparar()* debería ser invocada antes de invocar a la función *procesar()*. Así  ¡Ok!
- ☐ c. El programa lanza un error de intérprete: el script principal no puede consistir sólo en llamadas a funciones.
- ☐ d. El programa está mal planteado: la visualización de los resultados finales DEBE hacerse en el script principal y NO en una función separada.

¡Correcto!

La respuesta correcta es:

El programa está mal planteado: la función *comparar()* debería ser invocada antes de invocar a la función *procesar()*. Así como está, lanza un error al intentar ejecutar la función *procesar()* pues no reconoce a la variable *may*.

Pregunta **7**

Correcta

Se puntúa 1 sobre 1

Suponga que se quiere desarrollar un programa que cargue por teclado el nombre de una persona, su edad, y su sueldo anual promedio, y que luego se desea invocar a una función llamada **convertir()** que tome como parámetros a esos valores, los procese, y retorne una cadena con todos esos datos convertidos a una cadena de caracteres y concatenados. Tomando como modelo de uso a la función *test()* que se muestra en el programa de mas abajo... ¿cuál de las que se proponen como opciones podría ser la cabecera de la función **convertir()**?

```
__author__ = 'Cátedra de AED'

# Suponga que aquí está definida la función convertir()...
# def .....
# .....
# .....

def test():
    nombre = input('Ingrese el nombre: ')
    edad = int(input('Ingrese edad: '))
    sueldo = float(input('Ingrese sueldo:'))

    resultado = convertir(nombre, edad, sueldo)
    print('Datos registrados:', resultado)

# script principal
test()
```

Seleccione una:

- ☐ a. def convierte(nombre, edad, sueldo):
- ☐ b. def convertir(edad, sueldo):
- ☐ c. def convertir():
- ☒ d. def convertir(nombre, edad, sueldo): ✓ ¡Ok!

¡Correcto!

La respuesta correcta es:

def convertir(nombre, edad, sueldo):

Pregunta **8**

Correcta

Se puntúa 1 sobre 1

Suponga que se quiere desarrollar un programa para cargar una cadena de caracteres por teclado y mostrar esa cadena en la consola de salida mediante una función. ¿Cuál de las posibles respuestas *describe mejor* lo que ocurre con el programa que mostramos aquí?


```
__author__ = 'Cátedra de AED'

def mensaje(m):
    print('Mensaje:', m)

def test():
    mens = input('Ingrese un mensaje a mostrar: ')
    r = mensaje(mens)
    print('Programa terminado...')

# script principal...
test()
```

Seleccione una:

- ☐ a. El programa funciona, hace exactamente lo esperado, y no presenta ningún tipo de inconveniente ni elementos extraños en su código fuente.
- ☐ b. El programa compila (el intérprete no lanza error alguno al comenzar a ejecutarlo) y ejecuta, pero no muestra correctamente el mensaje esperado: en su lugar, muestra el valor *None*.
- ☐ c. El programa no compila (el intérprete lanza un error de sintaxis antes comenzar a ejecutarlo): No se puede invocar a la función *mensaje()* y asignarla en una variable, ya que esa función es de la forma sin retorno de valor.
- ☒ d. El programa funciona y hace exactamente lo esperado, pero la función *mensaje()* es una función sin retorno de valor, por  ¡Ok!

¡Correcto!

La respuesta correcta es:

El programa funciona y hace exactamente lo esperado, pero la función *mensaje()* es una función sin retorno de valor, por lo que no debería ser asignada en una variable cuando se la invoca.



Pregunta **9**

Correcta

Se puntúa 1 sobre 1

Suponga que se quiere desarrollar un programa para cargar un número por teclado y mostrar el triple de ese número en consola de salida  
¿Está bien planteado el siguiente programa?


```
__author__ = 'Cátedra de AED'

def triple(n):
    t = 3 * n

def test():
    a = int(input('Ingrese un numero: '))
    r = triple(a)
    print('El triple del numero es:', r)

# script principal
test()
```

Seleccione una:

- ☐ a. El programa lanza un error de intérprete: la función *triple()* está definida como una función sin retorno de valor, y se producirá un error al invocarla y asignar su retorno en *t*.
- ☐ b. Sí, el programa hace exactamente lo esperado.
- ☐ c. El programa lanza un error de intérprete: el parámetro formal de la función *triple()* no puede llamarse *n* (debería llamarse *a*).
- ☒ d. El programa ejecuta sin lanzar error, pero muestra un resultado *None* en lugar del triple del número cargado: falta la  ¡Ok!

¡Correcto!

La respuesta correcta es:

El programa ejecuta sin lanzar error, pero muestra un resultado *None* en lugar del triple del número cargado: falta la instrucción *return t* al final de la función *triple()*.

Pregunta **10**

Correcta

Se puntúa 1 sobre 1

¿Qué hace la siguiente función en Python?

```
def check(n):  
    if n % 2 == 0:  
        return True  
    else:  
        return False
```

Seleccione una:

- ☐ a. Retorna True si el número n tomado como parámetro es impar.
- ☐ b. Retorna True si el número n tomado como parámetro es mayor a 2.
- ☒ c. Retorna True si el número n tomado como parámetro es par. ✓ ¡Ok!
- ☐ d. Retorna True si el número n tomado como parámetro es primo.

¡Correcto!

La respuesta correcta es:

Retorna True si el número n tomado como parámetro es par.

◀ [Materiales Adicionales para la Ficha 10](#)

Ir a...



[Desafío 02 \[Problema: La Sucesión  \$3n + 1\$  \(o Sucesión de Collatz\)\]](#) ▶