Comenzado el	domingo, 17 de septiembre de 2023, 19:20
Estado	Finalizado
Finalizado en	domingo, 17 de septiembre de 2023, 19:35
Tiempo empleado	15 minutos 1 segundos
Calificación	**10** de 10 (**100**%)

Pregunta **1**

Correcta

Se puntúa 1 sobre 1

Suponga que la clase *Cliente* está convenientemente declarada. ¿Qué hace el siguiente script en Python?

```
n = 8
v = n * [None]
for i in range(n):
     v[i] = Cliente()
```

Seleccione una:

a. Crea un arreglo de *n = 8* componentes inicialmente valiendo *None*, y deja a ese vector con esas componentes en *None*.

b. Lanza un error de intérprete: la expresión *n * [None]* no tiene sentido.

c. Crea un arreglo de *n = 8* componentes con su casilleros valiendo *None*, y luego en cada uno de esos componentes asigna una referencia a un registro vacío del tipo *Cliente*. ✔

d. Crea un arreglo de *n = 8* componentes, para cada uno de esos componentes asigna referencias a objetos del tipo *Cliente*, y cada *Cliente* inicializa todos sus campos con el valor 8.

Pregunta **2**

Correcta

Se puntúa 1 sobre 1

Suponga que la clase *Materia* está convenientemente declarada para representar las asignaturas de una carrera universitaria. Asuma que también se ha definido un vector *mat* de referencias a objetos de tipo *Materia* y que ese vector fue creado con *n* casilleros valiendo *None* y cargado luego con objetos de ese tipo *Materia*, *pero no se tiene seguridad en cuanto a que todos los casilleros del vector hayan sido correctamente asignados con una referencia a un registro*. ¿Qué hace la siguiente función, suponiendo que está declarada en el mismo módulo que el vector?

```
def controlar(mat):
    n = len(mat)
    for i in range(n)):
        if mat[i] == None:
            return i
    return -1
```

Seleccione una:

- a. Si el vector *mat* contiene al menos una casilla valiendo *None*, retorna *None*. Si ninguna casilla es *None*, retorna el valor *-1*.
- b. Si el vector *mat* contiene al menos una casilla valiendo *None*, retorna el índice de la primera casilla que sea *None*. Si ninguna casilla es *None*, retorna el valor *-1*. ✔
- c. Si el vector *mat* contiene al menos una casilla valiendo *None*, retorna el índice de la última casilla que sea *None*. Si ninguna casilla es *null*, retorna el valor *-1*.
- d. Si el vector *mat* contiene al menos una casilla valiendo *None*, retorna un vector que contiene los índices de todas las casillas de *mat* que valen *None*. Si ninguna casilla es *None*, retorna el valor *-1*.

Pregunta **3**

Correcta

Se puntúa 1 sobre 1

Analice el siguiente programa en Python:

```
class Libro:
    def __init__(self, cod, tit, aut):
        self.isbn = cod
        self.titulo = tit
        self.autor = aut


def test():
    n = 10
    lib = n * [None]
    for i in range(n):
        lib[i].isbn = i
    print('Terminado...')


if __name__ == '__main__':
    test()
```

¿Qué tiene de malo este programa?

Seleccione una:

- a. No se han creado los objetos para cada casillero del arreglo. Cada casillero *est[i]* está valiendo None y por lo tanto es incorrecta la asignación est[i].isbn = i (provocará un error de intérprete y se interrumpirá el programa). ✔
- b. No se pueden crear arreglos que contengan objetos en Python.
- c. El problema es que en ninguna parte se invocó a la función *init()*, que debería ser invocada obligatoriamente.
- d. No hay nada de malo con ese segmento.

Pregunta **4**

Correcta

Se puntúa 1 sobre 1

Analice el siguiente script en Python:

```
class Libro:
    def __init__(self, cod, tit, aut):
        self.isbn = cod
        self.titulo = tit
        self.autor = aut


def test():
    a = Libro(1, 'AAA', 'aaa')
    b = Libro(2, 'BBB', 'bbb')
    c = Libro(3, 'CCC', 'ccc')

    n = 4
    v = n * [None]
    v[0] = a
    v[1] = b
    v[2] = c
    v[3] = v[1]

    a = None
    b = None
    c = None

    for i in range(n-1):
        v[i] = None

    print('Terminado...')


if __name__ == '__main__':
    test()
```

¿Cuál de las siguientes es correcta antes de llegar al *print()* final de la función *test()*?

Seleccione una:

a. Todos los objetos inicialmente apuntados por *a*, *b* y *c* quedan des-referenciados y son eliminados por el garbage collector.

b. Los objetos inicialmente apuntados por *a* y *c* quedan des-referenciados y son eliminados por el garbage collector. ✔

c. Ninguno de los objetos inicialmente apuntados por *a*, *b* y *c* queda des-referenciado.

d. Sólo el objeto inicialmente apuntado por *b* queda des-referenciado y es eliminado por el garbage collector.

Pregunta **5**

Correcta

Se puntúa 1 sobre 1

Suponga que la clase *Insumo* está convenientemente declarada para representar los insumos que se usan en la fabricación de una pieza. Asuma que también se ha definido un vector *pieza* de referencias a objetos de tipo *Insumo* y que ese vector fue creado con *n* casilleros valiendo *None*. El programa mostrado en la Ficha 20 (problema 48) prevé que el arreglo será cargado luego con objetos de ese tipo *Insumo*, entrando en la opción 1 del menú. Más abajo mostramos una variante de la función *opcion3()* para calcular el monto total insumido para fabricar la pieza completa, la cual a su vez invoca a la función *total_value()* (que se ha dejado sin cambios) ¿Qué puede decirse respecto de la forma en que afecta al programa este replanteo de la función *opcion3()*?

```
def total_value(pieza):
    tv = 0
    for insumo in pieza:
        monto = insumo.valor * insumo.cantidad
        tv += monto
    return tv


def opcion3(pieza):
    print('Monto total en insumos para la pieza:', total_value(pieza))
```

Seleccione una:

a. Si el vector pieza no hubiese sido efectivamente cargado antes, la función opcion3() invocará a total_value() enviándole un vector con todos sus casilleros valiendo None, y el ciclo iterador simplemente dejará el valor None en el acumulador tv. La función terminará en ese caso retornarndo None.

b. Si el vector *pieza* no hubiese sido efectivamente cargado antes, la función opcion3() invocará a total_value() enviándole un vector con todos sus casilleros valiendo None, y el ciclo iterador provocará un error y se interrumpirá ya que no existe en ese caso ningún campo llamado valor o cantidad. El programa se interrumpirá con un mensaje de error. ✔

c. Si el vector *pieza* no hubiese sido efectivamente cargado antes, la función opcion3() invocará a total_value() enviándole un vector con todos sus casilleros valiendo None, y el ciclo iterador automáticamente cambiará el None de cada casillero por un registro de tipo Insumo inicializado con valores cero. Por lo tanto en ese caso la función retornará el valor final 0.

d. El programa funcionará correctamente de todos modos. Si cada casillero del vector valiese None, el ciclo iterador de la función total_value() terminará sin hacer nada y la función retornará 0.

Pregunta **6**

Correcta

Se puntúa 1 sobre 1

Suponga que la clase registro *Inmueble* está convenientemente declarada. Suponga también que esa clase contiene un atributo *codigo* para almacenar el código de identificación del inmueble, otro para almacenar su precio, y algunos atributos más. Asuma que también se ha definido un vector *inm* de referencias a objetos de tipo *Inmueble* y que ese vector fue creado y cargado correctamente con objetos de tipo *Inmueble*. Se desea ordenar el arreglo en *orden creciente de códigos de identificación*. ¿Está bien planteada la siguiente función para lograr ese ordenamiento?

```
def ordenar(inm):
    n = len(inm)
    for i in range(n-1):
        for j in range(i+1, n):
            if inm[i].codigo > inm[j].codigo:
                inm[i], inm[j] = inm[j], inm[i]
```

Seleccione una:

- a. No, ya que esa función en realidad está ordenando el vector pero por precios en lugar de hacerlo por códigos.
- b. No. La función efectivamente ordena por códigos, pero lo hace en forma decreciente en lugar de hacerlo en forma creciente.
- c. No. La función hace incorrectamente los intercambios de objetos y deja mezclados todos los datos iniciales.
- d. Sí. La función está bien planteada. ✔

Pregunta **7**

Correcta

Se puntúa 1 sobre 1

Suponga que la clase *Insumo* está convenientemente declarada para representar los insumos que se usan en la fabricación de una pieza (ver problema 48 - Ficha 20). Asuma que también se ha definido un vector *pieza* de referencias a objetos de tipo *Insumo* y que ese vector fue convenientemente creado y cargado con datos de *n* objetos de tipo *Insumo*. En el programa de la Ficha 20 se incluyó una función *sort()* que ordenaba el arreglo de menor a mayor de acuerdo a los valores del atributo codigo de cada objeto. Mostramos más abajo esa función *sort()*, pero ligeramente modificada respecto de la versión original ¿Qué puede decirse respecto de esta nueva versión modificada? (Por razones de claridad, hemos transcripto también la definición original de la clase *Insumo*).

```
class Insumo:
    def __init__(self, cod=1, nom='', pre=0.0, cant=0):
        self.codigo = cod
        self.nombre = nom
        self.valor = pre
        self.cantidad = cant
```

```
def sort(pieza):
    n = len(pieza)
    for i in range(n-1):
        for j in range(i+1, n):
            if pieza[i].nombre > pieza[j].nombre:
                pieza[i], pieza[j] = pieza[j], pieza[i]
```

Seleccione una:

a. Así como está planteada, la función sigue ordenando el vector de menor a mayor, pero en lugar de hacerlo de acuerdo a los valores del atributo *codigo*, lo hace de acuerdo a los valores del atributo *valor*.

b. Así como está planteada, la función sigue ordenando el vector de menor a mayor, pero en lugar de hacerlo de acuerdo a los valores del atributo *codigo*, lo hace de acuerdo a los valores del atributo *cantidad*.

c. Así como está planteada, la función sigue ordenando el vector de menor a mayor de acuerdo a los valores del atributo *codigo*. No hay diferencia con lo que hacía la función original

d. Así como está planteada, la función sigue ordenando el vector de menor a mayor, pero en lugar de hacerlo de acuerdo a los ✔ valores del atributo *codigo*, lo hace de acuerdo a los valores del atributo *nombre* (el vector se ordenará en forma alfabética según los nombres de los insumos).

Pregunta **8**

Correcta

Se puntúa 1 sobre 1

Suponga que la clase *Materia* está convenientemente declarada para representar las asignaturas de una carrera universitaria. Suponga también que esa clase contiene un atributo *nivel* para almacenar el año de la carrera al que pertenece esa materia (será 1 si la materia es de primer año, 2 si es de segundo, etc.) y algunos atributos más. Asuma que se ha declarado también un vector *mat* de referencias a objetos de tipo *Materia* y que ese vector fue creado y cargado correctamente con objetos de tipo *Materia*. ¿Qué hace la siguiente función, suponiendo que está declarada en el mismo módulo que el vector?

```
def procesar(mat):
    n = len(mat)
    nuevo = []
    for i in range(n):
        if mat[i].nivel > 2:
            nuevo.append(mat[i])
    return nuevo
```

Seleccione una:

a. Crea un segundo vector que contendrá todas las materias que no sean de primero ni de segundo año, elimina esas materias del vector original, y retorna finalmente el nuevo vector.

b. Crea un segundo vector que contendrá todas las materias que no sean de primero ni de segundo año, y retorna finalmente el nuevo vector. ✔

c. Crea un segundo vector que contendrá todas las materias del vector original que estén desde la casilla 2 en adelante, y retorna finalmente el nuevo vector.

d. Crea un segundo vector que contendrá todas las materias de primero y segundo año, y retorna finalmente el nuevo vector.

Pregunta **9**

Correcta

Se puntúa 1 sobre 1

En el problema 50 de la Ficha 20 se introdujo la explicación de cómo generar en forma aleatoria un vector de objetos. ¿Cuáles podrían ser razones válidas por las que sería útil generar datos en forma automática? (Más de una puede ser cierta... marque TODAS las que considere correctas)

Seleccione una o más de una:

a. Para facilitar la prueba de un programa antes de dejarlo en su versión definitiva. ✔

b. No hay una razón válida para hacerlo. Los datos siempre deben ser cargados por teclado o recuperados desde un archivo externo si el mismo existe.

c. Para aplicar técnicas de generación de valores aleatorios y selección de valores aleatoriamente de una secuencia: aun cuando en un programa real los datos se carguen desde el teclado o desde un archivo, la generación automática es un recurso válido de aprendizaje y prueba. ✔

d. Para ganar tiempo (sobre todo en fase de prueba o cuando el programa es una simulación): la generacion automática ahorra mucho tiempo si los datos reales a cargar son muchos y/o si constan de combinaciones complicadas de valores. ✔

Pregunta **10**

Correcta

Se puntúa 1 sobre 1

En el problema 50 de la Ficha 20 se introdujo la explicación de cómo generar en forma aleatoria un vector de objetos. Uno de los atributos de ese registro debía contener el número de patente de un vehículo (en formato argentino de 1994). En el programa original se mostró una forma simple de hacerlo. Se indican ahora algunas otras posibles maneras de lograrlo. ¿Cuáles de las siguientes ideas efectivamente logran generar una patente argentina? (Más de una puede ser cierta... marque TODAS las que considere correctas)

Seleccione una o más de una:

a.
```
import random

letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + random.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = str(random.choice(digitos) + random.choice(digitos) + random.choice(digitos))

patente = p1 + p2
print(patente)
```

b. ✔
```
import random

letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + random.choice(letras)

digitos = '0123456789'
p2 = random.choice(digitos) + random.choice(digitos) + random.choice(digitos)

patente = p1 + p2
print(patente)
```

c.
```
import random

letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + random.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = random.choice(digitos) + random.choice(digitos) + random.choice(digitos)

patente = p1 + p2
print(patente)
```

d. ✔
```
import random

letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + random.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = str(random.choice(digitos)) + str(random.choice(digitos)) + str(random.choice(digitos))

patente = p1 + p2
print(patente)
```

◄ Materiales Adicionales para la Ficha 20

Ir a...

Guía de Ejercicios Prácticos 20 ►