Comenzado el	lunes, 17 de septiembre de 2018, 22:43
Estado	Finalizado
Finalizado en	lunes, 17 de septiembre de 2018, 23:03
Tiempo empleado	20 minutos 25 segundos
Puntos	25/25
Calificación	**10** de 10 (**100**%)

Pregunta 1

Correcta

Puntúa 1 sobre 1

Suponga que se quiere almacenar en un programa los datos de un *estudiante*, y para representar a ese estudiante se usa una *tupla* en la que el primer casillero guardará el legajo, el segundo el nombre y el tercero el promedio del estudiante. Suponga que parte del programa incluye una secuencia de instrucciones como la siguiente, en la cual se crea la tupla, se muestra su contenido, se cambia el nombre del estudiante y luego se muestran los datos modificados:

```
est = 12345, 'Juan', 8.76
print('Datos del estudiante:', est)

est[1] = 'Pedro'
print('Datos modificados del estudiante:', est)
```

¿Es correcto este enfoque o existe algún inconveniente? Si hay algún problema, indique cuál.

Seleccione una:

a.
No es correcto. Los elementos de una tupla no pueden accederse usando índices.

b.
Sí. Es correcto y el esquema mostrado funciona sin problemas.

c.
No es correcto. No se pueden almacenar valores de tipos diferentes en una tupla.

d.
No es correcto. Una tupla podría usarse para representar al estudiante, pero no permitirá modificar ningún componente al ser una secuencia de tipo inmutable. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:
No es correcto. Una tupla podría usarse para representar al estudiante, pero no permitirá modificar ningún componente al ser una secuencia de tipo inmutable.

Pregunta 2

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características básicas y generales de un *registro*? (Más de una respuesta puede ser válida. Marque todas las que considere correctas).

Seleccione una o más de una:

- a.
  Un registro es un conjunto mutable de datos que pueden ser de tipos diferentes. ✓

  ¡Correcto!

- b.
  Los elementos individuales de un registro se acceden por medio de un *nombre* o *identificador* declarado por el programador, en lugar de hacerlo por medio de índices. ✓

  ¡Correcto!

- c.
  Un registro es un conjunto inmutable de datos que pueden ser de tipos diferentes.

- d.
  Los elementos individuales de un registro se designan con el nombre genérico de *campos* o *atributos*. ✓

  ¡Correcto!

¡Correcto!

Las respuestas correctas son:
Un registro es un conjunto mutable de datos que pueden ser de tipos diferentes.,

Los elementos individuales de un registro se designan con el nombre genérico de *campos* o *atributos*.,

Los elementos individuales de un registro se acceden por medio de un *nombre* o *identificador* declarado por el programador, en lugar de hacerlo por medio de índices.

Pregunta 3

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de distintos libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el script de más abajo. ¿Hay algún problema con el programa siguiente?

```
class Libro:
    pass


def init(cod, nom, aut):
    libro = Libro()
    libro.codigo = cod
    libro.titulo = nom
    libro.autor = aut

    return libro


def write(libro):
    print('Datos del libro:')
    print('Código:', libro.codigo, ' - Título:', l
ibro.titulo, ' - Autor:', libro.autor)


def test():
    lib1 = init(2345, 'El Aleph', 'Jorge Luis Borg
es')
    lib2 = init(1267, 'Rayuela', 'Julio Cortázar')
    lib3 = init(1928, 'El Túnel', 'Ernesto Sábat
o')

    write(lib1)
    write(lib2)
    write(lib3)


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

Seleccione una:

a.

No es correcto. La función *init()* está creando nuevamente un registro vacío en su bloque de acciones (al hacer *libro = Libro()*). Cuando la función termina de ejecutarse, ese registro se pierde y el parámetro actual enviado a ella seguirá apuntando al registro vacío creado en *test()*. Cuando se invoque a la función *write()* se producirá entonces un error de intérprete porque intentará mostrar campos que no existen.

b.

No es correcto. Falta la creación del registro vacío en la función *test()* antes de invocar a *init()*.

c.

Sí. Es correcto y el esquema mostrado funciona sin problemas. ✓

¡Correcto!

d.

No es correcto. La función *init()* no puede invocar a la función constructora (sólo puede invocarse en la función *test()* o en cualquier función que sea usada como función de arranque del programa).

¡Correcto!

La respuesta correcta es:
Sí. Es correcto y el esquema mostrado funciona sin problemas.

Pregunta 4

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de distintos libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el script de más abajo. ¿Hay algún problema con el programa siguiente?

```
class Libro:
    pass


def init(libro, cod, nom, aut):
    libro = Libro()
    libro.codigo = cod
    libro.titulo = nom
    libro.autor = aut


def write(libro):
    print('Datos del libro:')
    print('Código:', libro.codigo, ' - Título:', l
ibro.titulo, ' - Autor:', libro.autor)


def test():
    lib1 = Libro()
    init(lib1, 2345, 'El Aleph', 'Jorge Luis Borge
s')

    lib2 = Libro()
    init(lib2, 1267, 'Rayuela', 'Julio Cortázar')

    lib3 = Libro()
    init(lib3, 1928, 'El Túnel', 'Ernesto Sábato')

    write(lib1)
    write(lib2)
    write(lib3)


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

Seleccione una:

a.

No es correcto. La función *init()* no puede invocar a la función constructora (sólo puede invocarse en la función *test()* o en cualquier función que sea usada como función de arranque del programa).

b.

Sí. Es correcto y el esquema mostrado funciona sin problemas.

c.

No es correcto. La función *init()* está creando nuevamente un registro vacío en su bloque de acciones (al hacer *libro = Libro()*). Cuando la función termina de ejecutarse, ese registro se pierde y el parámetro actual enviado a ella seguirá apuntando al registro vacío creado en *test()*. Cuando se invoque a la función *write()* se producirá entonces un error de intérprete porque intentará mostrar campos que no existen.

¡Correcto!

d.

No es correcto. Si la función *init()* está creando el registro (al hacer *libro = Libro()*) entonces no debe crearse el registro también en *test()*. En *test()* es suficiente con invocar a *init()* para crear cada registro.

¡Correcto!

La respuesta correcta es:
No es correcto. La función *init()* está creando nuevamente un registro vacío en su bloque de acciones (al hacer *libro = Libro()*). Cuando la función termina de ejecutarse, ese registro se pierde y el parámetro actual enviado a ella seguirá apuntando al registro vacío creado en *test()*. Cuando se invoque a la función *write()* se producirá entonces un error de intérprete porque intentará mostrar campos que no existen.

Pregunta 5

Correcta

Puntúa 1 sobre 1

Suponga el siguiente script elemental en el que se crea un tipo registro llamado *Libro*, y luego una variable de ese tipo:

```
class Libro:
    pass

lib = Libro()
```

¿Qué es lo que hace la función *Libro()* en este script?

Seleccione una:

a.
Define la variable *lib* y la deja asignada con el valor *None*.

b.
El script es incorrecto: la función *Libro()* no existe y se lanzará un error de intérprete.

c.
Crea un registro vacío de tipo *Libro*, lo ubica en memoria y retorna la dirección donde quedó ubicado. Esta dirección se asigna en la variable *lib*, con la cual luego se manejará el registro. ✓

¡Correcto!

d.
Retorna la dirección 0 (o dirección *nula*), que quedará asignada en la variable *lib*. No se crea ningún registro.

¡Correcto!

La respuesta correcta es:
Crea un registro vacío de tipo *Libro*, lo ubica en memoria y retorna la dirección donde quedó ubicado. Esta dirección se asigna en la variable *lib*, con la cual luego se manejará el registro.

Pregunta 6

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de distintos libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el script de más abajo. Suponga además que una vez creadas las variables *lib1*, *lib2* y *lib3* se quiere crear una nueva variable *lib4* que contenga los mismos datos que lib2, pero de tal forma que ambas se mantengan independientes: al modificar cualquier dato en una de las dos, no debe cambiar nada en la otra. ¿Hay algún problema con el programa siguiente, en base a estos requerimientos?

```
class Libro:
    pass


def init(libro, cod, nom, aut):
    libro.codigo = cod
    libro.titulo = nom
    libro.autor = aut


def write(libro):
    print('Datos del libro:')
    print('Código:', libro.codigo, ' - Título:', l
ibro.titulo, ' - Autor:', libro.autor)


def test():

    lib1 = Libro()
    init(lib1, 2345, 'El Aleph', 'Jorge Luis Borge
s')

    lib2 = Libro()
    init(lib2, 1267, 'Rayuela', 'Julio Cortázar')

    lib3 = Libro()
    init(lib3, 1928, 'El Túnel', 'Ernesto Sábato')

    write(lib1)
    write(lib2)
    write(lib3)

    lib4 = Libro()
    init(lib4, lib2.codigo, lib2.titulo, lib2.auto
r)
    lib4.autor = 'Adolfo Bioy Casares'

    write(lib2)
    write(lib4)

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

Seleccione una:

a.

No es correcto. En Python no hay forma de hacer que dos variables de tipo registro contengan los mismos datos y se mantengan independientes la una de la otra.

b.

No es correcto. La invocación a *init()* para inicializar *lib4* (`init(lib4, lib2.codigo, lib2.titulo, lib2.autor)`) hará que *lib2* y *lib4* apunten al mismo registro y desde allí en adelante cualquier cambio en una de ellas cambiará también la otra

c.

No es correcto. En lugar de invocar a init() para inizializar *lib4* con los datos de *lib2*, debería haberse asignado directamente *lib2* en *lib4* (o sea: *lib4* = *Libro()* y luego *lib4* = *lib2*).

d.

Sí. Es correcto y el esquema mostrado funciona sin problemas. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:
Sí. Es correcto y el esquema mostrado funciona sin problemas.

Pregunta 7

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de tres libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el siguiente script:

```
class Libro:
    pass

lib1 = Libro()
lib1.codigo = 2345
lib1.titulo = 'El Aleph'

lib2 = Libro()
lib2.codigo = 1267
lib2.titulo = 'Rayuela'
lib2.autor = 'Julio Cortázar'

lib3 = Libro()
lib3.isbn = 123456767
lib3.nombre = 'El Tunel'
lib3.precio = 145.56
```

¿Es correcto el script mostrado o existe algún inconveniente? Si hay algún problema, indique cuál.

Seleccione una:

a.
No es correcto. Las tres variables *lib1*, *lib2* y *lib3* se están definiendo con *campos de nombres o identificadores diferentes* y eso no es posible.

b.
No es correcto. Las tres variables *lib1*, *lib2* y *lib3* se están definiendo con *distinta cantidad de campos* y eso no es posible.

c.
Sí. Es correcto y el esquema mostrado funciona sin problemas. ✓

¡Correcto!

d.
No es correcto. Las tres variables *lib1*, *lib2* y *lib3* pueden tener campos con nombres diferentes o incluso distinta cantidad de campos, *pero al menos uno de los campos debe ser el mismo tipo para todos los registros*.

¡Correcto!

La respuesta correcta es:
Sí. Es correcto y el esquema mostrado funciona sin problemas.

Pregunta 8

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de tres libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el siguiente script:

```
class Libro:
    pass


def init(libro, cod, nom, aut):
    libro.codigo = cod
    libro.titulo = nom
    libro.autor = aut


def write(libro):
    print('Datos del libro:')
    print('Código:', libro.codigo, ' - Título:', l
ibro.titulo, ' - Autor:', libro.autor, ' - Preci
o:', libro.precio)


def test():

    lib1 = Libro()
    init(lib1, 2345, 'El Aleph', 'Jorge Luis Borge
s')

    lib2 = Libro()
    init(lib2, 1267, 'Rayuela', 'Julio Cortázar')

    lib3 = Libro()
    init(lib3, 1928, 'El Túnel', 'Ernesto Sábato')
    lib3.precio = 156.45

    write(lib1)
    write(lib2)
    write(lib3)


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

¿Es correcto el script mostrado o existe algún inconveniente? Si hay algún problema, indique cuál.

Seleccione una:

a.
Sí. Es correcto y el esquema mostrado funciona sin problemas.

b.

No es correcto. Luego de ser definida la variable *lib3*, se le está agregando el campo precio por separado y eso no es posible. El programa lanzará un error de intérprete en la linea *lib3.precio = 156.45*.

c.

No es correcto. Si se invoca a la función *init()* para inicializar los registros, entonces no debe invocarse a la función constructora *Libro()* (*init()* hace todo el trabajo).

d.

No es correcto. La función *write()* asume que el libro que toma como parámetro tiene siempre el campo *precio*, pero eso no es cierto al menos para las variables *lib1* y *lib2*. El programa lanzará un error de intérprete al intentar mostrar *lib1* o *lib2*.

¡Correcto!

¡Correcto!

La respuesta correcta es:
No es correcto. La función *write()* asume que el libro que toma como parámetro tiene siempre el campo *precio*, pero eso no es cierto al menos para las variables *lib1* y *lib2*. El programa lanzará un error de intérprete al intentar mostrar *lib1* o *lib2*.

Pregunta 9

Correcta

Puntúa 2 sobre 2

Suponga que se quiere representar datos de distintos libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el script de más abajo. Suponga además que una vez creadas las variables *lib1*, *lib2* y *lib3* se quiere crear una nueva variable *lib4* que contenga los mismos datos que *lib2*, pero de tal forma que ambas se mantengan independientes: al modificar cualquier dato en una de las dos, no debe cambiar nada en la otra. ¿Hay algún problema con el programa siguiente, en base a estos requerimientos?

```
class Libro:
    pass


def init(libro, cod, nom, aut):
    libro.codigo = cod
    libro.titulo = nom
    libro.autor = aut


def write(libro):
    print('Datos del libro:')
    print('Código:', libro.codigo, ' - Título:', libro.ti
tulo, ' - Autor:', libro.autor)


def test():

    lib1 = Libro()
    init(lib1, 2345, 'El Aleph', 'Jorge Luis Borges')

    lib2 = Libro()
    init(lib2, 1267, 'Rayuela', 'Julio Cortázar')

    lib3 = Libro()
    init(lib3, 1928, 'El Túnel', 'Ernesto Sábato')

    write(lib1)
    write(lib2)
    write(lib3)

    lib4 = lib2
    lib4.autor = 'Adolfo Bioy Casares'

    write(lib2)
    write(lib4)

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

Seleccione una:

- a.

No es correcto. La asignación *lib4 = lib2* hace que ambas variables queden referenciando al mismo registro. Cualquier cambio que se haga de allí en más en una de las dos, cambiará entonces también a la otra.

¡Correcto!

b.

Sí. Es correcto y el esquema mostrado funciona sin problemas.

c.

No es correcto. Antes de la asignación *lib4 = lib2* debería haberse creado la variable *lib4* con la función constructora *Libro()* (es decir: *lib4 = Libro()*) y sólo entonces asignar *lib2*.

d.

No es correcto. En Python no hay forma de hacer que dos variables de tipo registro contengan los mismos datos y se mantengan independientes la una de la otra.

¡Correcto!

La respuesta correcta es:
No es correcto. La asignación *lib4 = lib2* hace que ambas variables queden referenciando al mismo registro. Cualquier cambio que se haga de allí en más en una de las dos, cambiará entonces también a la otra.

Pregunta 10

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar una fecha en un programa, y se propone utilizar un esquema basado en el uso de un registro, como se muestra en el siguiente script:

```
class Fecha:
    pass

f = Fecha()
f.day = 27
f.month = 7
f.year = 2015

print('Fecha registrada: ', f.day, '/', f.month,
 '/', f.year, sep='')
```

¿Es correcto este enfoque o existe algún inconveniente? Si hay algún problema, indique cuál.

Seleccione una:

a.

No es correcto. La declaración de la clase *Fecha* no es válida: los campos deben declararse obligatoriamente dentro de ella (no puede dejarse vacía usando la instrucción *pass*).

b.

No es correcto. Los elementos de un registro no pueden accederse usando el operador *punto*.

c.

No es correcto. Un registro es un conjunto de datos que deben ser de tipos diferentes, y en este esquema todos los campos del registro *f* son del mismo tipo: *int*.

d.

Sí. Es correcto y el esquema mostrado funciona sin problemas. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:
Sí. Es correcto y el esquema mostrado funciona sin problemas.

Pregunta 11

Correcta

Puntúa 1 sobre 1

Suponga que el tipo registro *Cliente* está convenientemente declarado. ¿Qué hace el siguiente script en Python?

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

Seleccione una:

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

b.
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*. ✓

¡Correcto!

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

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

¡Correcto!

La respuesta correcta es:
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*.

Pregunta 12

Correcta

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 registros de tipo *Insumo* y que ese vector fue creado con ***n*** casilleros valiendo *None.* El programa mostrado en la Ficha 18 (problema 48) prevé que el arreglo será cargado luego con registros 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 piez
a:', 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 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.

c.
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.

d.

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.

¡Correcto!

¡Correcto!

La respuesta correcta es:
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.

Pregunta 13

Correcta

Puntúa 1 sobre 1

Suponga que *el* registro *Materia* está convenientemente declarado para representar las asignaturas de una carrera universitaria. Asuma que también se ha definido un vector *mat* de referencias a registros de tipo *Materia* y que ese vector fue creado con n casilleros valiendo *None* y cargado luego con registros 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 última casilla que sea *None*. Si ninguna casilla es *null*, retorna el valor *-1*.

c.
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*. ✓

¡Correcto!

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*.

¡Correcto!

La respuesta correcta es:
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*.

Pregunta 14

Correcta

Puntúa 2 sobre 2

Analice el siguiente script en Python:

```
__author__ = 'Cátedra de AED'


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.
Los registros inicialmente apuntados por *a* y *c* quedan des-referenciados y son eliminados por el garbage collector. ✓

¡Correcto!

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

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

d.

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

¡Correcto!

La respuesta correcta es:
Los registros inicialmente apuntados por *a* y *c* quedan des-referenciados y son eliminados por el garbage collector.

Pregunta 15

Correcta

Puntúa 1 sobre 1

El siguiente módulo contiene la declaración de una clase *Estudiante*, incluyendo *la función constructora __init__()* y la función *to_string()* para inicializar y convertir a cadena un registro. Analice la forma en que está planteada la función *to_string()*... ¿Qué efecto provocarán en la cadena retornada por ese método los *campos de reemplazo* usados al invocar al método *format()*?

```
__author__ = 'Cátedra de AED'


class Estudiante:
    def __init__(self, leg, nom, prom):
        self.legajo = leg
        self.nombre = nom
        self.promedio = prom


def to_string(estudiante):
    r = ''
    r += '{:<15}'.format('Legajo: ' + str(estudiante.legajo))
    r += '{:^30}'.format('Nombre: ' + estudiante.nombre)
    r += '{:>18}'.format('Promedio: ' + str(estudiante.promedio))
    return r
```

Seleccione una:

a.
El valor del campo *legajo* sale centrado, el *nombre* también sale centrado y el *promedio* se justifica la izquierda.

b.
El valor del campo *legajo* se justifica hacia la izquierda, el *nombre* sale centrado y el *promedio* se justifica también a la izquierda.

c.
El valor del campo *legajo* se justifica hacia la derecha, el *nombre* sale centrado y el *promedio* se justifica la izquierda.

d.
El valor del campo *legajo* se justifica hacia la izquierda, el *nombre* sale centrado y el *promedio* se justifica la derecha. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:
El valor del campo *legajo* se justifica hacia la izquierda, el *nombre* sale centrado y el *promedio* se justifica la derecha.

Pregunta 16

Correcta

Puntúa 1 sobre 1

Analice el siguiente programa en Python:

```
__author__ = 'Cátedra de AED'


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 pueden crear arreglos que contengan registros en Python.

b.
El problema es que en ninguna parte se invocó a la función *init()*, que debería ser invocada obligatoriamente.

c.
No se han creado los registros 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). ✓

¡Correcto!

d.
No hay nada de malo con ese segmento.

¡Correcto!

La respuesta correcta es:
No se han creado los registros 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).

Pregunta 17

Correcta

Puntúa 2 sobre 2

Suponga que el tipo registro *Materia* está convenientemente declarado para representar las asignaturas de una carrera universitaria. Suponga también que ese tipo registro contiene un campo *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 campos más. Asuma que se ha declarado también un vector *mat* de referencias a registros de tipo *Materia* y que ese vector fue creado y cargado correctamente con registros 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, y retorna finalmente el nuevo vector. ✓

¡Correcto!

b.
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.

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

d.
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.

¡Correcto!

La respuesta correcta es:
Crea un segundo vector que contendrá todas las materias que no sean de primero ni de segundo año, y retorna finalmente el nuevo vector.

Pregunta 18

Correcta

Puntúa 1 sobre 1

Suponga que el tipo registro *Inmueble* está convenientemente declarado. Suponga también que ese tipo contiene un campo *codigo* para almacenar el código de identificación del inmueble, otro para almacenar su precio, y algunos campos más. Asuma que también se ha definido un vector *inm* de referencias a registros de tipo *Inmueble* y que ese vector fue creado y cargado correctamente con registros 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.
Sí. La función está bien planteada. ✓

¡Correcto!

c.
No. La función efectivamente ordena por códigos, pero lo hace en forma decreciente en lugar de hacerlo en forma creciente.

d.
No. La función hace incorrectamente los intercambios de registros y deja mezclados todos los datos iniciales.

¡Correcto!

La respuesta correcta es:
Sí. La función está bien planteada.

Pregunta 19

Correcta

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 40 - Ficha 16). Asuma que también se ha definido un vector *pieza* de referencias a registros de tipo *Insumo* y que ese vector fue convenientemente creado y cargado con datos de ***n*** registros de tipo *Insumo.* En el programa de la Ficha 18 se incluyó una función sort que ordenaba el arreglo de menor a mayor de acuerdo a los valores del campo codigo de cada registro. 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].nomb
re:
                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 de acuerdo a los valores del campo *codigo*. No hay diferencia con lo que hacía la función original

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 campo *codigo*, lo hace de acuerdo a los valores del campo *valor*.

c.

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 campo *codigo*, lo hace de acuerdo a los valores del campo *nombre* (el vector se ordenará en forma alfabética según los nombres de los insumos). ✓

¡Correcto!

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 campo *codigo*, lo hace de acuerdo a los valores del campo *cantidad*.

¡Correcto!

La respuesta correcta es:
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 campo *codigo*, lo hace de acuerdo a los valores del campo *nombre* (el vector se ordenará en forma alfabética según los nombres de los insumos).

Pregunta 20

Correcta

Puntúa 1 sobre 1

En el problema 51 de la Ficha 18 se introdujo la explicación de cómo generar en forma aleatoria un vector de registros. ¿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:

- [x] a.
  Para aplicar técnicas de generación de valores aleatorios y selección de valores aleatoriamente de una secuencia: aun cuanddo 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. ✓

  ¡Correcto!

- [x] b.
  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. ✓

  ¡Correcto!

- [ ] c.
  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.

- [x] d. Para facilitar la prueba de un programa antes de dejarlo en su versión definitiva. ✓ ¡Correcto!

¡Correcto!

Las respuestas correctas son: Para facilitar la prueba de un programa antes de dejarlo en su versión definitiva.,
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.,

Para aplicar técnicas de generación de valores aleatorios y selección de valores aleatoriamente de una secuencia: aun cuanddo 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.

Pregunta 21

Correcta

Puntúa 2 sobre 2

En el problema 51 de la Ficha 18 se introdujo la explicación de cómo generar en forma aleatoria un vector de registros. Uno de los campos 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) + r
andom.choice(letras)

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

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

b.

```
import random

letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + r
andom.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)
```

c.

```
import random

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

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

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

✓ ¡Correcto!

d.

```
import random

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

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

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

✓ ¡Correcto!

¡Correcto!

Las respuestas correctas son:

```
import random

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

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

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

,

```
import random

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

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

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