Comenzado el	lunes, 17 de septiembre de 2018, 23:00
Estado	Finalizado
Finalizado en	lunes, 17 de septiembre de 2018, 23:16
Tiempo empleado	16 minutos 8 segundos
Puntos	13/13
Calificación	**10** de 10 (**100**%)

Pregunta 1

Correcta

Puntúa 1 sobre 1

¿Cuál es la diferencia entre la *abstracción de datos y la abstracción funcional*?

Seleccione una:

a.
La abstracción de datos busca captar el conjunto de datos más relevante para representar un tipo abstracto, mientras que la funcional buscar determinar el conjunto de procesos relevante para esos datos.

¡Correcto!

b.
La abstracción de datos busca captar el conjunto de procesos relevante para el tipo abstracto que se quiere implementar, mientras que la funcional buscar determinar el conjunto de datos más relevante para implementar ese tipo.

c.
Ninguna. Son sólo dos formas de referirse al mecanismo de abstracción.

d.
No existe un mecanismo de abstracción de datos ni un mecanismo de abstracción funcional. Existe sólo un mecanismo de abstracción, sin dividir en asbtracción de datos y abstracción funcional.

¡Correcto!

La respuesta correcta es:
La abstracción de datos busca captar el conjunto de datos más relevante para representar un tipo abstracto, mientras que la funcional buscar determinar el conjunto de procesos relevante para esos datos.

Pregunta 2

Correcta

Puntúa 1 sobre 1

Suponga que se quiere implementar un nuevo tipo de datos abstracto llamado *Fecha*, para permitir la manipulación de fechas en un programa en Python para gestión de eventos sociales (casamientos, fiestas de cumpleños, reuniones de egresados, etc.) ¿Cuál de las siguientes estrategias de abstracción sería la más adecuada?

Seleccione una:

a.

Abstracción de datos: un registro con campos para el año, el mes, el día, el nombre del cliente, el tipo de evento, el monto presupuestado y una proyección metereológica para ese día. *Abstracción funcional*: funciones para buscar una fecha dada, reservar una fecha para un evento, liberar una fecha para un evento y modificar los datos de una fecha para un evento.

b.

Abstracción de datos: un registro con tres campos para el año, el mes y el día. *Abstracción funcional*: funciones para buscar una fecha dada, reservar una fecha para un evento, liberar una fecha para un evento y modificar los datos de una fecha para un evento. ✓

¡Correcto!

c.

Abstracción de datos: una tupla con tres componentes para el año, el mes y el día. *Abstracción funcional*: funciones para buscar una fecha dada, reservar una fecha para un evento, liberar una fecha para un evento y modificar los datos de una fecha para un evento.

d.

Abstracción de datos: una variable de tipo *cadena de caracteres* para representar toda la fecha (por ejemplo: '2015/08/31'). *Abstracción funcional*: funciones para mostrar la fecha en distintos colores, imprimir la fecha en forma de cartel anunciador, cambiar el número del mes por el nombre del mes.

¡Correcto!

La respuesta correcta es:
Abstracción de datos: un registro con tres campos para el año, el mes y el día. *Abstracción funcional*: funciones para buscar una fecha dada, reservar una fecha para un evento, liberar una fecha para un evento y modificar los datos de una fecha para un evento.

Pregunta 3

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expresiones describe la forma de trabajo general de una *Pila*?

Seleccione una:

a.
FIFO (First In - First Out)

b.
LIFO (Last In - First Out) ✓

¡Correcto!

c.
OIFO (Ordered In - First Out)

d.
OIRO (Ordered In - Random Out)

¡Correcto!

La respuesta correcta es:
LIFO (Last In - First Out)

Pregunta 4

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes situaciones de programación es la principal aplicación de una *Pila*?

Seleccione una:

a.
Procesar una secuencia de datos en el mismo orden en el que ingresaron.

b.
Procesar una secuencia de datos en orden aleatorio.

c.
Procesar una secuencia de datos en orden de menor a mayor.

d.
Procesar una secuencia de datos en orden inverso al de su entrada.

¡Correcto!

¡Correcto!

La respuesta correcta es:
Procesar una secuencia de datos en orden inverso al de su entrada.

Pregunta 5

Correcta

Puntúa 2 sobre 2

Suponga que se tiene una pila *p* con capacidad para almacenar números enteros y que las operaciones básicas *pop()*, *peek()* y *push()* están correctamente implementadas en el módulo *stack.py* presentado en clases. Suponga que se han insertado los siguientes valores: [8 - 6 - 5 - 3 - 7] (el número 8 es el valor del frente o tope de la Pila). ¿Cuál de las siguientes secuencias de instrucciones permite retirar el valor 5 de la pila, pero dejando el 6 y 8 nuevamente arriba? (es decir: ¿cuál de las siguientes secuencias, dejaría la pila *p* en el estado [8 - 6 - 3 - 7]?)

Seleccione una:

a.

```
n1 = stack.pop(p)
n2 = stack.pop(p)
x = stack.peek(p)
stack.push(p, n2)
stcak.push(p, n1)
```

b.

```
n1 = stack.pop(p)
n2 = stack.pop(p)
x = stack.pop(p)
stack.push(p, n2)
stack.push(p, n1)
```

¡Correcto!

c.

```
n1 = stack.pop(p)
n2 = stack.pop(p)
x = stack.pop(p)
stack.push(p, n1)
stack.push(p, n2)
```

d.

```
n1 = stack.pop(p)
n2 = stack.pop(p)
x = stack.pop(p)
```

¡Correcto!

La respuesta correcta es:

```
n1 = stack.pop(p)
n2 = stack.pop(p)
x = stack.pop(p)
stack.push(p, n2)
stack.push(p, n1)
```

Pregunta 6

Correcta

Puntúa 2 sobre 2

Suponga que se tiene una pila *p* en la cual se almacenaron ya una cierta cantidad de datos. ¿Cuál de las siguientes secuencias de instrucciones permite *invertir* la pila? (o sea: si la pila original *p* era: [3 - 4 - 6 - 7] (con el 3 al frente) ¿cuál de las siguientes permitiría dejar la pila *p* en el estado [7 - 6 - 4 - 3] (con el 7 al frente)?) (Suponga que *p2* y *p3* también son pilas, inicialmente vacías y listas para usar)

Seleccione una:

a.

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p, stack.pop(p2))
```

b.

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p3, stack.pop(p2))
```

c.

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p3, stack.pop(p2))
while not stack.is_empty(p3):
    stack.push(p2, stack.pop(p3))
```

d.

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p3, stack.pop(p2))
while not stack.is_empty(p3):
    stack.push(p, stack.pop(p3))
```

✓ ¡Correcto!

¡Correcto!

La respuesta correcta es:

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p3, stack.pop(p2))
while not stack.is_empty(p3):
    stack.push(p, stack.pop(p3))
```

Pregunta 7

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expresiones describe la forma de trabajo general de una *Cola*?

Seleccione una:

- a. FIFO (First In - First Out) ✓

  ¡Correcto!

- b. OIRO (Ordered In - Random Out)

- c. OIFO (Ordered In - First Out)

- d. LIFO (Last In - First Out)

¡Correcto!

La respuesta correcta es:
FIFO (First In - First Out)

Pregunta 8

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes situaciones de programación es la principal aplicación de una *Cola*?

Seleccione una:

a.
Procesar una secuencia de datos en orden inverso al de su entrada.

b.
Procesar una secuencia de datos en orden de menor a mayor.

c.
Procesar una secuencia de datos en el mismo orden en el que ingresaron. ✓

¡Correcto!

d.
Procesar una secuencia de datos en orden aleatorio.

¡Correcto!

La respuesta correcta es:
Procesar una secuencia de datos en el mismo orden en el que ingresaron.

Pregunta 9

Correcta

Puntúa 2 sobre 2

Suponga que se tiene una cola *c* en la cual se almacenaron ya una cierta cantidad de datos. ¿Cuál de las siguientes secuencias de instrucciones permite *invertir* la cola? (o sea: si la cola original *c* era: [3 - 4 - 6 - 7] (con el 3 al frente), ¿cuál de las siguientes permitiría dejar la cola *c* en el estado [7 - 6 - 4 - 3] (con el 7 al frente)?)

Seleccione una:

a.

```
# suponga que c2 es OTRA cola vacía y lista para usa
r...
while not queue.is_empty(c):
    queue.add(c2, queue.remove(c))
while not queue.is_empty(c2):
    queue.add(c, queue.remove(c2))
```

b.

```
# suponga que p1 y p1 son dos PILAS inicialmente vacía
s, y listas para usar
while not queue.is_empty(c):
    stack.push(p1, queue.remove(c))
while not stack.is_empty(p1):
    stack.push(p2, stack.pop(p1))
while not stack.is_empty(p2):
    queue.add(c, stack.pop(p2))
```

c.

```
# suponga que c2 y c3 son dos colas inicialmente vacía
s, y listas para usar
while not queue.is_empty(c):
    queue.add(c2, queue.remove(c))
while not queue.is_empty(c2):
    queue.add(c3, queue.remove(c2))
while not queue.is_empty(c3):
    queue.add(c, queue.remove(c3))
```

d.

```
# suponga que p es una PILA vacía y lista para usar...
while not queue.is_empty(c):
    stack.push(p, queue.remove(c))
while not stack.is_empty(p):
    queue.add(c, stack.pop(p))
```

✓ ¡Correcto! Efectivamente... si desea la inversión de una secuencia, una pila le permitirá lograrlo...

¡Correcto!

La respuesta correcta es:

```
# suponga que p es una PILA vacía y lista para usar...
while not queue.is_empty(c):
    stack.push(p, queue.remove(c))
while not stack.is_empty(p):
    queue.add(c, stack.pop(p))
```

Pregunta 10

Correcta

Puntúa 1 sobre 1

Una *Cola de Prioridad* es una cola en la cual los elementos se insertan en algún orden, pero tal que cuando se pide retirar un elemento se obtiene siempre *el menor* de los valores almacenados en la cola ¿Cuál de las siguientes estrategias podría ser una forma básica de implementar una *Cola de Prioridad* en las condiciones aquí expresadas?

Seleccione una:

a.
Soporte: una variable de tipo *list* en Python. Inserción: *ordenada de mayor a menor* (mantener el arreglo siempre ordenado de mayor a menor: para insertar un nuevo valor *x*, recorrer el arreglo y al encontrar el primer menor, a *x* detener el ciclo y añadir allí a *x* con un corte de índices. Eliminación: siempre el primer elemento.

b.
Soporte: una variable de tipo *list* en Python. Inserción: siempre al final. Eliminación: siempre el último elemento.

c.
Soporte: una varable de tipo *list* en Python. Inserción: siempre al frente. Eliminación: siempre el primer elemento.

d.
Soporte: una variable de tipo *list* en Python. Inserción: *ordenada de menor a mayor* (mantener el arreglo siempre ordenado de menor a mayor: para insertar un valor *x*, recorrer el arreglo y al encontrar el primer mayor a *x* detener el ciclo y añadir allí a *x* con un corte de índices. Eliminación: siempre el primer elemento. ✓

¡Correcto! Aunque entienda que si se hace así, logrará el objetivo pero la inserción tendrá un tiempo de ejecución O(n) mientras que la eliminación será de tiempo constante (O(1)).

¡Correcto!

La respuesta correcta es:
Soporte: una variable de tipo *list* en Python. Inserción: *ordenada de menor a mayor* (mantener el arreglo siempre ordenado de menor a mayor: para insertar un valor *x*, recorrer el arreglo y al encontrar el primer mayor a *x* detener el ciclo y añadir allí a *x* con un corte de índices. Eliminación: siempre el primer elemento.