| | |
|---|---|
| **Instituto Tecnológico de Costa Rica**<br>**Área Académica de Ingeniería en Computadores**<br>**Programa de Licenciatura en Ingeniería en Computadores**<br>**Curso:** CE-1101 Introducción a la programación<br>**Profesor:** Lic. Ing. Fabián Zamora Ramírez<br>**Semestre:** I, 2018 | **Práctica de temas:**<br>• Programación recursiva con Python<br>Para 2do examen. |

**INDICACIONES:**

Para todos los siguientes ejercicios muestre su solución en lenguaje de programación Python. Toda función debe estar documentada con una descripción breve de lo que realiza y mostrar sus entradas (E), salidas (S) y restricciones (R). Si no se especifican restricciones, debe realizar las restricciones lógicas según el problema. Recuerde utilizar nombres significativos para sus funciones y variables. En caso de no cumplir con lo anterior se calificará que nota de cero.

1. Escriba una función **multdig(num1, num2)** que recibe 2 números enteros del mismo tamaño y forma un nuevo número con la multiplicación de cada dígito del primer número con cada dígito del segundo número. Si la multiplicación de 2 dígitos es mayor a 9, se toma el dígito menos representativo del resultado de la multiplicación. Los siguientes ejemplos muestran cómo debe comportarse la ejecución de la función:

   ```
   >>> multdig(24, 42)
   88
   >>> multdig(323, 388)
   964
   >>> multdig(153, 632)
   656
   ```

2. Escribir una función **dig_medio(num)** que reciba un número entero positivo y obtenga el siguiente resultado: si el número tiene una cantidad par de dígitos devuelve 0, en caso contrario devuelve el dígito que se encuentra en la mitad del número. Ejemplo: si num=3456 devuelve 0, si num=6501201 devuelve 1. Debe construir todas las funciones requeridas para resolver este problema.

3. Un número es palíndromo si puede leerse igual de izquierda a derecha que de derecha a izquierda. Escribir una función llamada **palindromo(num)** que reciba un número y retorne True si el número recibido es un palíndromo y False si no. Por ejemplo:

   >>> palindromo(38583)

   True

   >>> palindromo(2442)

   True

   >>> palindromo(4)

   True

   >>> palindromo(1010010)

   False

4. Escriba una función **parejas(num)** en la cual num es un valor entre 2 y 100 que encuentre todas las parejas (sin importar el orden) de números que suman el número dado. Ejemplos:

   >>> parejas(4)

   [[0, 4], [1, 3], [2, 2]]

5. Escriba una función llamada **digitos(n)** que reciba un número entero n y retorne una lista que tenga la siguiente forma: (el cero se considera par)

   (suma-dígitos-pares  suma-dígitos-impares)

   >>> digitos(482401)

   (18, 1)

6. Escriba una función **cambie_repetidos(lista)** que reciba una lista y sustituya todos los valores repetidos por -1 (salvo la primera ocurrencia del elemento repetido). Debe cambiar los repetidos de las listas anidadas también, sin importar el nivel de profundidad.

   >>> cambie_repetidos ([ 1, 2, 5, 3, 1, 5, 1] )

   [1, 2, 5, 3, -1, -1, -1]

7. Escriba una función en recursividad de cola llamada **elimine_todos(ele, lista)** que reciba un elemento y una lista y elimine todas las apariciones del elemento en la lista, aún si el elemento se encuentra en una sub-lista (a cualquier nivel).

   >>> elimine_todos(4, [4, 5, [[[3, 4], 5], [3]], 4, 8])

   [5, [[[3], 5], [3]], 8]

8. Construya una función a llamar **permutaciones(lista)** que recibe una lista y devuelve todos los posibles ordenamientos de esa lista. La función debe comportarse de la siguiente forma:

```
>>> permutaciones([])
[]
>>> permutaciones([1, 2, 3])
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]
```

**OBJETOS.**

9. Se va a definir un objeto empleado que tenga los siguientes datos: número de empleado, nombre, salario por hora. El objeto va a contener los siguientes métodos: mostrar() que despliega los datos de una instancia y calculoSalario(horas) que recibe una cantidad de horas y muestra cual debería ser el salario. Implementar esta clase en Python.
10. En una empresa que vende suministros de oficina se desea controlar el saldo de los clientes de crédito. Se va a definir un objeto Cliente con los siguientes datos: número de cliente, nombre, saldo (inicialmente es cero). El objeto va a contener los siguientes métodos: mostrar(), registraDoc(tipo, monto) que recibe tipo de documento (que va a ser “fa” o “re”) y monto de documento que actualiza el saldo del cliente. Los documentos tipo factura “fa” suman al saldo del cliente y los documentos recibo “re” disminuyen el saldo del cliente. En caso de registrarse un documento que convierta el saldo en negativo debe indicarse un mensaje de advertencia y procesar la transacción. Debe implementar las funciones necesarias para el manejo de esta clase.
11. Se desea construir una clase para representar los objetos de tipo carro. Se debe considerar la marca, modelo y año de cada objeto. La forma de crear objetos sería por ejemplo l = Carro(“Toyota”, “Rav4”, 2000). Cada carro va a contar con un contador de kilómetros (empieza en cero cuando se crea el objeto), útil para definir aspectos relacionados con garantía, revisiones y otros. Estos objetos deben responder a los siguientes mensajes: l.marca() retorna Toyota, l.modelo() retorna Rav4, l.año() retorna 2000. El kilometraje se actualiza por medio de un método llamado viaje(kms) que recibe como argumento la cantidad de kilómetros y actualiza el contador de kilómetros del vehículo. Además existe un método llamado estado(), que retorna “Revisión” si los kilómetros del vehículo son múltiplos de 5000, retorna “Fuera Garantía” si los kilómetros son mayores a 100,000 y “Normal” en otro caso. Debe programar la clase en Python.

**12.** Para el control de una sala de cines de TerraMedia se va a manejar (entre muchos otros) un objeto para representar las presentaciones o funciones que se realizan. Va a contener los siguientes datos:

- nombre película: un string con el nombre
- sala: un número entre 1 y 14
- fecha: una lista con 3 elementos: [dia, mes, ano]
- hora: una lista con 2 elementos: [hora, minutos]
- categoría: un string con “R” para mayor 18 años, “L” para Todo público).

a. Defina una clase para manejar este objeto Presentacion, asumiendo que se van a tener los siguientes métodos: mostrar() que despliega los datos de una instancia, retornaFecha() que retorna la fecha de una instancia, actualizaHora(hora) que recibe una hora y la modifica, actualizaFecha(fecha) que recibe una fecha y la cambia.

b. Asuma la creación de diferentes instancias de este objeto, que se crean tomando como base una lista similar a la siguiente:

```
>>> lista = [Presentacion(“Angeles y Demonios”, 1, [1, 06, 2009], [6, 45], “L”),
            Presentacion(“Terminator the salvation”,  1, [1, 06, 2004], [9, 15], “L”),
            Presentacion((“Up”,  2,  [2, 06, 2009] [7, 15], “L”),
Presentacion(“Hanna Montana”, 2, [2, 06, 2009], [9, 15], “R”)]
```

En este ejemplo la variable lista va a contener varias instancias de la clase Presentacion. Defina una función (fuera del objeto) llamada busque(lista, fecha, indicador), que reciba la lista de instancias, una fecha y un indicador de categoría (“R” = mayores, “L” = Libre, “*” = Todas) y muestre los nombres de las películas, número de sala, hora de la función y categoría que están programadas para esa fecha.

**13.** Asuma que se tiene una lista de instancias de un objeto tipo camión, conteniendo el número de placa, marca, año, estado camión (libre, ocupado, reparación) capacidad de carga en kg. y cantidad de viajes realizados. Defina una clase para el manejo de estos objetos, los cuales pertenecen a una empresa de transportes. Los métodos a implementar son: tipo, mostrar, reset que pone en cero la cantidad de viajes realizados, enviar a reparación, recibir de reparación y actualizar viajes. Luego de tener definida la clase, se asume que se tiene la lista de instancias y que se quiere construir una función pedido, para saber cuál o cuáles camiones podrán transportar una determinada mercadería. En este caso se recibiría la

cantidad de kg a transportar y se debe indicar cuales camiones (libres) podrían cubrir el pedido.

**14.** Se tiene un objeto tipo maletín cuyos datos van a ser: una capacidad y una lista de útiles (inicialmente vacía). Los objetos que podrían estar en la lista van a contener los siguientes datos: tipo, nombre, peso, importancia. Por ejemplo, una instancia de maletín podría tener capacidad 8 y lista de útiles vacía. Las instancias de objetos tipo útil podrían ser: [[oficina, palm, 1, 5], [oficina, calculadora, 1, 1], [personal, cepillo-pasta, 1, 4], [personal, pasaporte, 0.5, 5]] y muchas otras. Implementar para el objeto maletín los métodos: mostrar() que despliega los datos y llenar() que busca en la lista de instancias de útiles una forma de llenar (en forma completa) el maletín sin sobrepasar su capacidad. Puede asumirse que la lista de instancias de los objetos tipo útiles ya está creada.

**15.** Los siguientes ejercicios de árboles binarios deben realizarse como métodos adicionales de una clase Arbol, que tiene la siguiente definición:

```
class Nodo:
    def __init__(self, hijoizq=None, hijoder=None, valor=None):
        self.hijoizq = hijoizq
        self.hijoder = hijoder
        self.valor = valor

class Arbol:
    def __init__(self):
        self.raiz = None
```

a. Escriba un método maxValor() que obtenga el valor mayor de un árbol binario ordenado.
b. Escriba un método minValor() que obtenga el valor menor de un árbol binario ordenado.
c. Escriba un método llamado frontera() que reciba la instancia de un árbol binario y obtenga la frontera del árbol, es decir, la secuencia de hojas, haciendo el recorrido de izquierda a derecha. Si a es una instancia del objeto Arbol y tiene la siguiente estructura de árbol:


```
>>> a.frontera()
[3, 4, 12, 14]
```

d. Escriba un método **nodos()** que calcule el número de nodos internos (no hojas) de un árbol binario.