Universidad Central de Venezuela.
Facultad de Ingeniería.
Dpto. Investigación de Operaciones y Computación.
Programación.
Prof. Carlos E. González C.

Primer Proyecto.

9 de agosto de 2019

1. Introducción

En el presente documento se muestran los lineamientos para la elaboración del primer proyecto de las secciones 06 y 07 de la materia **Programación (0790)**.

El proyecto será realizado en equipo. Se evaluará la correcta utilización de los estándares de identificadores, la originalidad de los algoritmos implementados, minimización de las líneas de código y la correcta documentación de las funciones y los módulos.

2. Módulo validation

Como actividad inicial, el estudiante debe implementar un módulo llamado validation cuyo fin será proveer un conjunto de funciones destinadas a la validación de variables (tipo de datos y contenido). El módulo debe contar con las siguientes funciones:

1. valInt()
2. valFloat()
3. valComplex()
4. valList()

2.1. Funciones de validación numérica

Las funciones valInt(), valFloat() y valComplex(), tienen como fin validar tipos numéricos. En caso de recibir *un solo argumento*, dicha función debe arrojar `True` si el argumento coincide con el tipo esperado (señalado explicitamente en el nombre de la función), de no coincidir, la función debe arrojar como resultado `False`.

En caso de recibir *dos argumentos*, se espera el primer argumento del tipo `int`, `float` o `complex` para las funciones `valInt()`, `valFloat()` y

`valComplex()` respectivamente. Como segundo argumento se espera un tipo **`tuple`** o**`list`** con dos valores numéricos ordenados de forma creciente, datos que representarán un intervalo. En caso de que el segundo argumento sea pasado con una tupla (e.g., `(3,8)`), la función no debe incluir a los extremos de dicho intervalo para la validación. Como contraparte, si los argumentos son pasados con una lista (e.g., `[3,8]`) la función debe considerar a los extremos del intervalo para la validación.

A continuación se resume el comportamiento **mínimo** esperado para cada una de estas funciones y se anexa una tabla de ejemplos:

2.1.1. valInt()

1. `True`: Si se introduce un solo argumento de tipo **`int`**. Si se introducen dos argumentos y el primero es de tipo **`int`** y está en el intervalo indicado por el segundo argumento.

2. `False`: Si se introduce un solo argumento y el mismo *no es* de tipo **`int`**. Si se introducen dos argumentos y el primero de ellos *no es* de tipo **`int`** ó no está en el intervalo indicado por el segundo argumento.

3. `TypeError`: En caso de recibir dos argumentos y que el segundo no sea de un tipo esperado.

4. `ValueError`: En caso de recibir dos argumentos y que el segundo no esté ordenado de manera creciente o tenga un tamaño no considerado.

5. **Tabla de ejemplos**:

Llamado	Salida
valInt(4)	True
valInt(4.0)	False
valInt(4,(4,10))	False
valInt(4,[3,9])	True
valInt(4,[4,10])	True
valInt(4,[10,4])	ValueError
valInt(4,5)	TypeError

2.1.2. valFloat()

1. `True`: Si se introduce un solo argumento de tipo **`float`**. Si se introducen dos argumentos y el primero de ellos es de tipo **`float`** y está en el intervalo indicado por el segundo argumento.

2. `False`: Si se introduce un solo argumento y el mismo no es de tipo **`float`**. Si se introducen dos argumentos y el primero de ellos no es de tipo **`float`** ó no está en el intervalo indicado por el segundo argumento.

3. `TypeError`: En caso de recibir dos argumentos y que el segundo no sea de un tipo esperado.

4. `ValueError`: En caso de recibir dos argumentos y que el segundo no esté ordenado de manera creciente o tenga un tamaño no considerado.

5. **Tabla de ejemplos**:

Llamado	Salida
valFloat(4.0)	True
valFloat(4)	False
valFloat(4.4,(4.4,10))	False
valFloat(4.4,(4,10))	True
valFloat(4.1,[4.1,9.05])	True
valFloat(4.2,[4,10])	True
valFloat(4.4,[10,4])	ValueError
valFloat(4,5)	TypeError

2.1.3. valComplex()

1. `True`: Si se introduce un solo argumento y el mismo es de tipo **`complex`**. Si se introducen dos argumentos, el primero de ellos es **`complex`** y su módulo está en el intervalo indicado por el segundo argumento.

2. `False`: Si se introduce un solo argumento y el mismo *no es* de tipo **`complex`**. Si se introducen dos argumentos y el primero de ellos *no es* de tipo **`complex`** ó su módulo no está en el intervalo indicado por el segundo argumento.

3. `TypeError`: En caso de recibir dos argumentos y que el segundo no sea de un tipo esperado.

4. `ValueError`: En caso de recibir dos argumentos y que el segundo no esté ordenado de manera creciente o tenga un tamaño no considerado.

5. **Tabla de ejemplos**:

Llamado	Salida
valComplex(3+4j)	True
valComplex(4.0)	False
valComplex(3+4j,(5,10))	False
valComplex(3+4j,[5,10])	True
valComplex(3+4j,[4,10])	True
valComplex(3+4j,[10,4])	ValueError
valComplex(3+4j,5)	TypeError

Recuerde que el módulo de un número complejo $a + bj$ está dando por:

$$|a + bj| = \sqrt[2]{a^2 + b^2} \qquad (1)$$

2.2. valList()

La función `valList()` debe recibir 1 ó 3 argumentos. Si la función recibe 3 argumentos, el segundo argumento debe ser una variable de tipo **`list`** y el tercer argumento debe ser una cadena de texto que reciba las siguientes opciones; `'len'` ó `'values'`.:

1. `True`: Si se introduce un argumento y el mismo es de tipo **`list`**. Si el tercer argumento es `'value'` y el primer argumento y el segundo son lista y tienen los mismos elementos. Si el tercer argumento es `'len'`, el primer argumento es una lista, el segundo argumento es un entero y la longitud del primer argumento coincide con el segundo argumento.

2. `False`: Si se pasa un único argumento y el mismo no es tipo lista. Si el tercer argumento es `'value'` y el primer argumento y el segundo no son listas ó no son iguales. Si el tercer argumento es `'len'` y el primer argumento no es lista, o si la longitud del primer argumento no coincide con el segundo.

3. `TypeError`: En caso de que se introduzcan 3 argumentos y sean diferentes a las siguientes combinaciones:

   a) El tipo del argumento 2 es **`int`** y el tipo del argumento 3 **`str`**.

   b) El tipo del argumento 2 es **`list`** y el tipo del argumento 3 es **`str`**.

4. `ValueError`: En caso de que el tercer argumento sea diferente a `'list'` o `'value'`.

3. Módulo algebra

Para este módulo debe programar funciones para lo siguiente

1. Multiplicación de matrices.
2. Inversión de matrices (gauss-jordan).
3. Producto vectorial.
4. Transposición de matrices.
5. Resolución de sistema de ecuaciones lineales (utilizando la función de inversión).
6. Determinante de una matriz (extra).

Todas las funciones deben realizarse, emulando a los arreglos como listas de listas.

4. Módulo crypto

Escriba un módulo con las funciones que usted considere necesarias para realizar encriptación de mensajes. En lineas generales (y poco rigurosas), entienda como encriptación el proceso de codificar un texto con una cadena de caracteres diferentes. Ejemplo:

Hola Mundo ->c-aj h|x¿-

En este caso la letra “H” fue cambiada por “c”, la letra o por “-” y así sucesivamente.

Este módulo debe incluir funciones para codificar y decodificar dichas cadenas de texto.

5. Comentarios

- Se debe realizar diagrama de flujo para todos los algoritmos del módulo algebra.py.
- Si considera que falta alguna especificación en las funciones, se deja a su criterio (con la respectiva argumentación). Intente considerar de forma exhaustiva los posibles casos.
- Fecha de entrega 23 y 24 de septiembre del 2019.