VNIVERSITAT DE VALÈNCIA

Laboratorio de ESTRUCTURAS DE DATOS Y ALGORITMOS
Grado en Ciencia de Datos(1º)
Curso 2024-25

Práctica Nº 4: Programación con Pilas y Colas

Ejercicio de aula (L3)

Se asume que se ha realizado el trabajo previo indicado en el guion: programa "`pr4_fase1.py`".

Como tarea previa a este laboratorio se ha realizado un programa que permite analizar las líneas de programa escrito en Python, identificando aquellas líneas que contienen paréntesis y los errores que pudieran contener. Ahora, se debe realizar un nuevo programa que sea capaz de extraer información más concreta sobre este tipo de líneas del programa. En particular, sobre las que contienen paréntesis bien formados.

Para realizar el programa te puedes basar en el programa "`pr4_fase1.py`", pero es recomendable escribirlo como un programa nuevo incluyendo del programa anterior solo los elementos necesarios y adaptarlos adecuadamente. El nuevo programa, llamado "**`pr4_final.py`**", cumplirá los siguientes requisitos:

1. Se debe ampliar la definición de la clase **`Linea`** realizada previamente, de manera que:
    - Contendrá 3 datos:
        1. Número de la línea en el texto analizado.
        2. Texto completo de la línea (sin "\n").
        3. Lista con las posiciones de apertura y cierre de cada una de las parejas de paréntesis correctamente formadas en la línea. Cada elemento de la lista será una tupla **`(ini,fin)`**, donde **`ini`** será la posición del paréntesis abierto y **`fin`** la posición del paréntesis cerrado.
    - Toda la información descrita deberá ser proporciona a los objetos de la clase en el propio constructor.
    - Tendrá 3 operaciones públicas, cada una de las cuales permitirán devolver los diferentes datos almacenados en estos objetos (`GetNumero`, `GetTexto`, `GetPosParentesis`). La última devolverá la lista indicada anteriormente.
2. Se debe incorporar al programa la función `AnalizarParentesis`. Copia la función que tienes en "`pr4_fase1.py`" y modifícala de acuerdo con los siguientes criterios:
    - La función debe devolver una lista que contendrá las posiciones de apertura y cierre de cada una de las parejas de paréntesis correctamente formadas en la cadena. Cada elemento de la lista será una tupla **`(ini,fin)`**, donde **`ini`** será la posición del paréntesis abierto y **`fin`** la posición del paréntesis cerrado.
      La información que devuelve la función puede ser utilizada para almacenarla en un objeto de la clase **`Linea`**.
    - La lista devuelta **estará vacía** si la línea analizada contiene algún **error** de paréntesis, no importa cuál. Por lo tanto, se puede saber que una línea contiene paréntesis bien formados y no contiene ningún error porque la lista devuelta no estará vacía.
    - La función no debe devolver ningún valor booleano. Solo una lista.

3. Incorporados los elementos anteriores, el programa principal debe ajustarse a la siguiente estructura:
    - Leer el archivo "`test.py`" y para cada una de sus líneas:
        1. Analizar la concordancia de paréntesis.
        2. Si (y solo si) la línea contiene paréntesis correctamente formados, y sin errores, guardar en una cola un objeto de la clase `Linea` con la información relevante sobre esa línea correctamente formada.
    - Al finalizar la lectura, procesar la cola obtenida (leer nota al final del documento) en el paso anterior para mostrar la información solo de aquellas **líneas que contienen paréntesis bien formados** (y sin errores). Para cada una estas líneas se debe mostrar la siguiente información:
        1. Número de línea.
        2. Para cada paréntesis correcto en la línea, el texto contenido entre esos paréntesis. Sabes las posiciones de inicio y final, así que será sencillo extraer esa información del texto de la línea.
4. Comprueba que los resultados obtenidos con el archivo "`test.py`" concuerdan con los indicados a continuación:

```
Linea: 19
        Contenido = self,item,next=None
Linea: 23
        Contenido = self
Linea: 26
        Contenido = self
Linea: 29
        Contenido = self, item
Linea: 32
        Contenido = self,next
Linea: 35
        Contenido = self,contents=[]
Linea: 36
        Contenido = None,None
Linea: 40
        Contenido = e
Linea: 42
        Contenido = self,index
Linea: 44
        Contenido =
Linea: 45
        Contenido = index
Linea: 46
        Contenido =
Linea: 50
        Contenido =
Linea: 52
        Contenido = self,index,val
Linea: 54
        Contenido =
Linea: 55
        Contenido = index
Linea: 56
        Contenido =
Linea: 57
        Contenido = val
Linea: 63
        Contenido = self,other
Linea: 64
        Contenido = self
        Contenido = other
Linea: 67
        Contenido =
Linea: 68
        Contenido =
Linea: 70
        Contenido =
```

```
        Contenido = cursor.GetItem()
Linea: 71
        Contenido =
Linea: 72
        Contenido =
Linea: 74
        Contenido =
        Contenido = cursor.GetItem()
Linea: 75
        Contenido =
Linea: 78
        Contenido = self,item
Linea: 79
        Contenido = item
Linea: 80
        Contenido = node
Linea: 84
        Contenido = self,index,item
Linea: 87
        Contenido = index
Linea: 88
        Contenido =
Linea: 89
        Contenido =
        Contenido = item, cursor.GetNext()
Linea: 90
        Contenido = node
Linea: 93
        Contenido = item
Linea: 95
        Contenido = self
Linea: 97
        Contenido = self.__numItems
Linea: 98
        Contenido = self[i]
Linea: 103
        Contenido =
Linea: 104
        Contenido = [1,2,3]
Linea: 105
        Contenido = milista
Linea: 108
        Contenido = milista
Linea: 110
        Contenido = [4,5,6,7,8,9]
Linea: 113
        Contenido = l
Linea: 115
        Contenido = l[3]
Linea: 119
        Contenido =
```

Solo la línea 66 del archivo contiene paréntesis bien formados y errores, por lo tanto, esa línea no debe aparecer en los resultados. Compruébalo explícitamente.

Comentario sobre el uso de colas: Al analizar el contenido de una cola deberás ir extrayendo uno a uno los elementos que la componen. En este proceso la cola quedará vacía. Para que esto no ocurra, deberás realizar previamente una copia "profunda" de la cola en otra variable y trabajar sobre esa copia, dejando la cola original intacta. Para ello, debes importar la función `deepcopy` del módulo `copy`:

```
from copy import deepcopy
```

y cuando sea necesario hacer una copia puedes usarla así:

```
aux = deepcopy(c)
```

La cola `aux` será una copia de la cola `c`.