FACULTAD DE
INGENIERÍA Y CIENCIAS

udp

Universidad Diego Portales

CIT2000

Estructuras de Datos

Tarea 1

Autor:
Ze Hui Fu Zeng

11 de septiembre de 2019

Índice

1. Introducción

En el siguiente informe encontrará la solución a la Tarea 1 del ramo Estructura de Datos en el cual se pide que se programe en Python 3 una Calculadora Polaca. La Calculadora Polaca solo podrá solucionar problemas que estén escritas en notación polaca (primero se escribe la operación y luego los dos números a operar), si esta no esta escrita de esta forma se indicará qué la operación esta mal definida. Las operaciones aceptadas por esta calculadora serán la multiplicación, división, suma y resta, también podrá contener paréntesis. Un ejemplo de notación polaca sería la siguiente:
1) + 1 2 : Se sumará 1 y 2.
2) - (* 3 4) (/ 10 -5) : Se resuelve el primer paréntesis, luego el segundo y finalmente se restarán ambos.

2. Solución

Para esta calculadora usaremos Python el cual es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.
Se importará el módulo ¨re¨ (Regular Expresión), el cual se usa para ver si un string tiene patrones específicos como números y caracteres, también se importará el módulo ¨sys¨ que provee información crucial de de como Python esta interactuando con el host system, esta la usaremos para salir de Python si es que la operación ingresada esta mal definida.
El programa solo tendrá un input, el cual será ¨Ingrese ecuación con notación polaca correctamente escrita: ¨ en donde se debe ingresar la ecuación a calcular correctamente como en los ejemplos, y tendrá un solo output el cual será el resultado de la ecuación ingresada correctamente.

```
import re
import sys
def Separar(ecuacion):
    ec=ecuacion.split()
    completo=[]
    while len(ec)!=0:
        if ec[0].isdigit()==False:
            if len(ec[0])>=2:
                ped=re.split('(\d+)',ec[0])
                while ped:
                    if ped.count(''):
                        ped.remove('')
                    while ped:
                        if len(ped)-1>0 and ped[0]=='-':
                            if ped[1].isdigit():
                                ped[0]=int(ped[0]+ped[1])
                                ped.pop(1)
                                completo.append(ped[0])
                                ped.pop(0)
                            elif ped[0].isdigit():
                                completo.append(int(ped[0]))
                                ped.pop(0)
                            else:
                                for c in ped[0]:
                                    completo.append(c)
                                ped.pop(0)
                ec.pop(0)
            else:
                completo.append(ec[0])
                ec.pop(0)
        else:
            completo.append(int(ec[0]))
            ec.pop(0)
    if completo.count('(')!=completo.count(')'):
            print('Operación mal definida 1.')
            sys.exit()
    if len(completo)>3:
        c=0
        for i in range(len(completo)):
            if type(completo[i])==int:
                c+=1
                if c>2:
                    print('Operación mal definida 2.')
                    sys.exit()
            else:
                c=0
    return completo
```

Figura 1: Función Separar.

En la primera función (figura 1) llamada “Separar” se introduce la ecuación a calcular y esta la descompone en un arreglo, en la cual cada posición habrá un string con un carácter o número omitiendo los espacios. Por ejemplo el usuario ingresa - (* 3 4) (/ 10 -5), la función retornará [’-’, ’(’, ’*’, ’3’, ’4’, ’)’, ’(’, ’/’, ’10’, ’-5’, ‘)’]. En esta función y también se verificará si la ecuación escrita es correcta o no.

```
    while len(ec)!=0:
        if ec[0].isdigit()==False:
            if len(ec[0])>=2:
                ped=re.split('(\d+)',ec[0])
                while ped:
                    if ped.count(''):
                        ped.remove('')
                    while ped:
                        if len(ped)-1>0 and ped[0]=='-':
                            if ped[1].isdigit():
                                ped[0]=int(ped[0]+ped[1])
                                ped.pop(1)
                                completo.append(ped[0])
                                ped.pop(0)
                            elif ped[0].isdigit():
                                completo.append(int(ped[0]))
                                ped.pop(0)
                            else:
                                for c in ped[0]:
                                    completo.append(c)
                                ped.pop(0)
                ec.pop(0)
            else:
                completo.append(ec[0])
                ec.pop(0)
        else:
            completo.append(int(ec[0]))
            ec.pop(0)
```

Figura 2: Ciclo para separar.

Dentro de la función se utilizará un ciclo (figura 2) que no parará hasta que migre toda la ecuación hacia otro arreglo llamado completo. Para separar los números y caracteres de la ecuación usaremos una función para verificar si el string en la posición X es un string de números o no, si este es un número se introducirá completamente al otro arreglo, pero no lo es significa que pueden haber números pegados a caracteres, números negativos o caracteres. Usaremos una función importada re.split(’(d+)’,string), esto hace que el string se separe en caracteres y números, por ejemplo si string=’-11))’, lo separará en un arreglo A=[‘-‘,’11’,’))’] y a veces le sale un espacio en blanco así que simplemente lo eliminamos si aparece, luego hacemos un ciclo que no acaba hasta que el arreglo A quede vacío, para saber si es un número negativo el arreglo tendrá que tener más de largo 1 y un string debe ser ‘-‘, si pasa eso y el que se encuentra a la derecha de ‘-‘ es un dígito entonces será un número negativo, si no es un número negativo entonces probamos con un número positivo y finalmente si ninguna de las dos es serán caracteres.

```
34      if completo.count('(')!=completo.count(')'):
35              print('Operación mal definida 1.')
36              sys.exit()
37      if len(completo)>3:
38          c=0
39          for i in range(len(completo)):
40              if type(completo[i])==int:
41                  c+=1
42                  if c>2:
43                      print('Operación mal definida 2.')
44                      sys.exit()
45              else:
46                  c=0
```

Figura 3: Verificar.

Para verificar si la operación esta correctamente definida (figura 3) contaremos sí los paréntesis ‘(‘ tienen la misma cantidad que los paréntesis opuestos ‘)’. La otra forma es verificar si hay 3 números enteros seguidos sin haber ningún paréntesis entre ellos.

```
49  def CalcularPolaca(ecuacion):
50      calculos=[]
51      while ecuacion.count('('):
52          ecuacion.pop(ecuacion.index('('))
53      while ecuacion.count(')'):
54          ecuacion.pop(ecuacion.index(')'))
55      while ecuacion or len(calculos)!=1:
56          if len(calculos)>=3:
57              if type(calculos[-1])==int and type(calculos[-2])==int:
58                  if calculos[-3]=='*':
59                      calculos[-3]=calculos[-2]*calculos[-1]
60                      calculos.pop()
61                      calculos.pop()
62                  elif calculos[-3]=='/':
63                      calculos[-3]=int(calculos[-2]/calculos[-1])
64                      calculos.pop()
65                      calculos.pop()
66                  elif calculos[-3]=='-':
67                      calculos[-3]=calculos[-2]-calculos[-1]
68                      calculos.pop()
69                      calculos.pop()
70                  elif calculos[-3]=='+':
71                      calculos[-3]=calculos[-2]+calculos[-1]
72                      calculos.pop()
73                      calculos.pop()
74              elif ecuacion:
75                  calculos.append(ecuacion[0])
76                  ecuacion.pop(0)
77          elif ecuacion:
78              calculos.append(ecuacion[0])
79              ecuacion.pop(0)
80          else:
81              print('Operación mal definida 3.')
82              sys.exit()
83      return int(calculos[0])
```

Figura 4: Función CalcularPolaca.

En la función CalcularPolaca (figura 4) se deberá llamar conjunto al arreglo que retorno la primera función (figura 1). Primero sacaremos todos los paréntesis del arreglo, para luego hacer un ciclo que no se detenga hasta que el arreglo ingresado sea nulo o que el largo del arreglo cálculos sea 1, cuando el arreglo cálculos llegue a largo 3 comenzará a revisar si el último y el penúltimo ingresado son números, si es así se calcularán los dos con la operación en la posición antepenúltima del arreglo, de no cumplirse estos requisitos se procederá a introducir más números o operaciones,

si el arreglo cálculos queda en más de 1 de largo significa que la función estuvo mal definida.

```
Ingrese la ecuacion: * (+ 1 -2) (- (* 2 -3) (/ 22 11))
8
>>>
============== RESTART: /Users/Zehui/Desktop/Python/po
Ingrese la ecuacion: - (* 3 4) (/ 10 -5)
14
>>>
============== RESTART: /Users/Zehui/Desktop/Python/po
Ingrese la ecuacion: + (- (* 1 2) (+ 9 9)) -3
-19
>>>
============== RESTART: /Users/Zehui/Desktop/Python/po
Ingrese la ecuacion: ((((((+ 1 1))))))
2
```

Figura 5: Ejemplos.

Cómo se puede ver el programa funciona con todos los ejemplos propuestos de la tarea (figura 5).

```
Ingrese la ecuacion: + (- 1 2
Operación mal definida 1.
>>>
============== RESTART: /Users/Zeh
Ingrese la ecuacion:  + + 1 2 3
Operación mal definida 2.
>>>
============== RESTART: /Users/Zeh
Ingrese la ecuacion: - 3 4 + 1 2
Operación mal definida 3.
>>>
============== RESTART: /Users/Zeh
Ingrese la ecuacion: 1 + 1
Operación mal definida 4.
```

Figura 6: Ejemplos de errores.

El programa detecta errores de mala definición de operaciones (figura 6) en el primer caso no cierra paréntesis, en el segundo no usa paréntesis, en el tercero falta una operación, y cuarto no lo escribe en notación polaca.

3. Conclusión

En esta tarea se familiarizó con el uso de listas en forma de stacks, pudiendo resolver ecuaciones complejas en notación polaca.
Se aprendió a usar la función .split() la cual separa string en un arreglo donde cada posición lo define un espacio, pero esto se puede cambiar dependiendo de la conveniencia.
Se utilizo el modulo sys en donde podemos ver información tales como versión y poder parar el programa cuando se necesite con sys.exit(). Se ejercito con el modulo re, el cual sirve para ver si un string tiene patrones específicos de dígitos o caracteres y poder separarlos en un arreglo con distintos comandos, pero se uso uno es especifico re.split('(d+)',string).
Se profundizo en el uso de ciclos en arreglos para recorrerlos y modificarlos según la necesidad que se tenga.