课程实践 20181111

实习二：2.5 算术表达式求值演示

郭宸 17081511

一、需求分析

表达式计算是实现程序设计语言的基本问题之一，也是栈的应用的一个典型例子。设计一个程序，演示用算符优先法对算术表达式求值的过程。 以字符序列的形式从终端输入语法正确的、不含变量的整数表达式。利用教科书表3.1给出的算符优先关系，实现对算术四则混合运算表达式的求值，并仿照教科书的例3-1演示在求值中运算符栈、运算数栈、输入字符和主要操作的变化过程。

二、概要设计

```
class Stack(object):
#创建栈类型
def operate(a,o,b):
#运算函数。a，b为进行运算的变量，o为运算符
def precede(a,b):
#运算符比较函数，仿照课本进行编写
def isInt(x):
#判断x是否为整数，在选座内容中会用到
def jisuan(s)
#算数表达式求值计算器，含演示功能
```

三、详细设计

biaodashi2.py

```
class Stack(object):
    def __init__(self):
        self.items = []
    def is_empty(self):
        return self.items == []
    def peek(self):
        return self.items[len(self.items) - 1]
    def size(self):
        return len(self.items)
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()
    def delete(self):
        self.items = []
```

```
def operate(a,o,b):
    if(o == '*'): return a*b
    elif(o == '/'): return a/b
    elif(o == '+'): return a+b
    elif(o == '-'): return a-b
def precede(a,b):
    if(a != '#'):
        if(a == '+' or a == '-'):
            if(b == '+' or b == '-' or b == ')' or b == '#'):
                return '>'
            elif(b == '*' or b == '/' or b == '('):
                return '<'
        elif(a == '*' or a == '/'):
            if(b == '('):
                return '<'
            else: return '>'
        elif(a == '('):
            if(b == ')'):
                return '='
            else: return '<'
        elif(a == ')'):
            return '>'
    elif(a == '#'):
        if(b == '#'): return '='
        else:
            return '<'
intNumber = []
for n in range(10):
    intNumber.append(str(n))
def isInt(x):
    for n in intNumber:
        if(x == n): return 1
    return 0
def jisuan(s):
    b = list(s)
    b.append('#')
    Number = Stack()
    operator = Stack()
    operator.push('#')
    i = 0
    NumberTemp = []
    while (True):
        print("本次输入为: ", b[i],end='\t')
        print("运算符栈为: ", operator.items, end='\t')
        if (b[i] == '*' or b[i] == '/' or b[i] == '+' or b[i] == '-' or b[i] == '('
or b[i] == ')' or b[i] == '#'):
            if (NumberTemp != []):
                Nt = "".join(NumberTemp)
                Number.push(float(Nt))
                NumberTemp = []
            if (precede(operator.peek(), b[i]) == '<'):
                operator.push(b[i])
            elif (precede(operator.peek(), b[i]) == '='):
```

```
                x = operator.pop()
            elif (precede(operator.peek(), b[i]) == '>'):
                theta = operator.pop()
                x2 = Number.pop()
                x1 = Number.pop()
                print("运算", x1, theta, x2, end='\t')
                Number.push(operate(x1, theta, x2))
                i -= 1
        elif (isInt(b[i]) == 1 or b[i] == '.'):
            NumberTemp.append(b[i])
        elif(b[i] <= 'z' and b[i] >= 'A'):
            cTemp = b[i]
            b[i] = input("输入" + b[i] + "的值\n")
            for j in range(i,len(b)):
                if(b[j] == cTemp):
                    b[j] = b[i]
            NumberTemp.append(b[i])
        else:
            NumberTemp.append(b[i])
        print("运算数栈为: ", Number.items, end='\n')
        i += 1
        if (operator.peek() == '#' and b[i] == '#'):
            print("本次输入为: ", b[i],end='\t')
            print("运算符栈为: ", operator.items, end='\t')
            print("运算数栈为: ", Number.items, end='\n')
            break
if(Number.is_empty() != True):
    print("\n\n运算结果为: ", Number.peek())
else: print("\n\n运算结果为: ", float("".join(NumberTemp)))
if __name__ == '__main__':
    s = input("输入表达式:" + "\n")
    jisuan(s)
```

四、调试分析

使用input读入表达式（为字符串类型）

使用python的列表list函数将表达式字符串转换为列表后存储为b，然后创建两个栈Number和operator分别用来存储运算数和运算符。

在此之前又定义了NumberTemp列表来存储多次循环中遇到的连续的数与小数点和字符。使用函数isInt判断其是否为一个数。当遇到运算符时，将NumberTemp列表转换为float型的数，并清空NumberTemp栈，以便下一次读入数。

当遇到既不是数也不是小数点也不是运算符的字符时，用cTemp存储读到的字符。然后使用输入函数，将列表b在该处的值输入进去。再使用for循环，将b中所有与cTemp相同的字符都赋值为输入的数字。

经过以上步骤可以实现：

①运算量可以是变量②运算量可以是字符

两个选做内容

其余设计与课本类似，不再赘述

五、用户手册

本程序在安装了python3和的所有系统上均可运行。

先输入表达式，不需要以'#'结尾。之后就可以看到运行结果

六、测试结果

1.教科书例3-1的算数表达式3*(7-2)

```
PS H:\data structures> python biaodashi2.py
输入表达式:
3*(7-2)
本次输入为：  3  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  *  运算符栈为：  ['#']       运算数栈为：  [3.0]
本次输入为：  (  运算符栈为：  ['#', '*']  运算数栈为：  [3.0]
本次输入为：  7  运算符栈为：  ['#', '*', '(']     运算数栈为：  [3.0]
本次输入为：  -  运算符栈为：  ['#', '*', '(']     运算数栈为：  [3.0, 7.0]
本次输入为：  2  运算符栈为：  ['#', '*', '(', '-']        运算数栈为：  [3.0, 7.0]
本次输入为：  )  运算符栈为：  ['#', '*', '(', '-']        运算 7.0 - 2.0  运算数栈为：  [3.0, 5.0]
本次输入为：  )  运算符栈为：  ['#', '*', '(']     运算数栈为：  [3.0, 5.0]
本次输入为：  #  运算符栈为：  ['#', '*']  运算 3.0 * 5.0  运算数栈为：  [15.0]
本次输入为：  #  运算符栈为：  ['#']       运算数栈为：  [15.0]


运算结果为：  15.0
PS H:\data structures>
```

2.1+2+3+4

```
PS H:\data structures> python biaodashi2.py
输入表达式:
1+2+3+4
本次输入为：  1  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  +  运算符栈为：  ['#']       运算数栈为：  [1.0]
本次输入为：  2  运算符栈为：  ['#', '+']  运算数栈为：  [1.0]
本次输入为：  +  运算符栈为：  ['#', '+']  运算 1.0 + 2.0  运算数栈为：  [3.0]
本次输入为：  +  运算符栈为：  ['#']       运算数栈为：  [3.0]
本次输入为：  3  运算符栈为：  ['#', '+']  运算数栈为：  [3.0]
本次输入为：  +  运算符栈为：  ['#', '+']  运算 3.0 + 3.0  运算数栈为：  [6.0]
本次输入为：  +  运算符栈为：  ['#']       运算数栈为：  [6.0]
本次输入为：  4  运算符栈为：  ['#', '+']  运算数栈为：  [6.0]
本次输入为：  #  运算符栈为：  ['#', '+']  运算 6.0 + 4.0  运算数栈为：  [10.0]
本次输入为：  #  运算符栈为：  ['#']       运算数栈为：  [10.0]


运算结果为：  10.0
```

3.1024/4*8

```
输入表达式:
1024/4*8
本次输入为：  1  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  0  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  2  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  4  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  /  运算符栈为：  ['#']       运算数栈为：  [1024.0]
本次输入为：  4  运算符栈为：  ['#', '/']  运算数栈为：  [1024.0]
本次输入为：  *  运算符栈为：  ['#', '/']  运算 1024.0 / 4.0       运算数栈为：  [256.0]
本次输入为：  *  运算符栈为：  ['#']       运算数栈为：  [256.0]
本次输入为：  8  运算符栈为：  ['#', '*']  运算数栈为：  [256.0]
本次输入为：  #  运算符栈为：  ['#', '*']  运算 256.0 * 8.0       运算数栈为：  [2048.0]
本次输入为：  #  运算符栈为：  ['#']       运算数栈为：  [2048.0]
```

4.1024/(4*8)

```
PS H:\data structures> python biaodashi2.py
输入表达式:
1024/(4*8)
本次输入为：  1  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  0  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  2  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  4  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  /  运算符栈为：  ['#']       运算数栈为：  [1024.0]
本次输入为：  (  运算符栈为：  ['#', '/']  运算数栈为：  [1024.0]
本次输入为：  4  运算符栈为：  ['#', '/', '(']     运算数栈为：  [1024.0]
本次输入为：  *  运算符栈为：  ['#', '/', '(']     运算数栈为：  [1024.0, 4.0]
本次输入为：  8  运算符栈为：  ['#', '/', '(', '*']        运算数栈为：  [1024.0, 4.0]
本次输入为：  )  运算符栈为：  ['#', '/', '(', '*']        运算 4.0 * 8.0  运算数栈为：  [1024.0, 32.0]
本次输入为：  )  运算符栈为：  ['#', '/', '(']     运算数栈为：  [1024.0, 32.0]
本次输入为：  #  运算符栈为：  ['#', '/']  运算 1024.0 / 32.0      运算数栈为：  [32.0]
本次输入为：  #  运算符栈为：  ['#']       运算数栈为：  [32.0]
```

5.$(((6+6)*6+3)*2+6)*2$

```
PS H:\data structures> python biaodashi2.py
输入表达式:
(((6+6)*6+3)*2+6)*2
本次输入为：  (  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  (  运算符栈为：  ['#', '(']  运算数栈为：  []
本次输入为：  (  运算符栈为：  ['#', '(', '(']     运算数栈为：  []
本次输入为：  6  运算符栈为：  ['#', '(', '(', '(']        运算数栈为：  []
本次输入为：  +  运算符栈为：  ['#', '(', '(', '(']        运算数栈为：  [6.0]
本次输入为：  6  运算符栈为：  ['#', '(', '(', '(', '+']   运算数栈为：  [6.0]
本次输入为：  )  运算符栈为：  ['#', '(', '(', '(', '+']   运算 6.0 + 6.0  运算数栈为：  [12.0]
本次输入为：  )  运算符栈为：  ['#', '(', '(', '(']        运算数栈为：  [12.0]
本次输入为：  *  运算符栈为：  ['#', '(', '(']     运算数栈为：  [12.0]
本次输入为：  6  运算符栈为：  ['#', '(', '(', '*']        运算数栈为：  [12.0]
本次输入为：  +  运算符栈为：  ['#', '(', '(', '*']        运算 12.0 * 6.0 运算数栈为：  [72.0]
本次输入为：  +  运算符栈为：  ['#', '(', '(']     运算数栈为：  [72.0]
本次输入为：  3  运算符栈为：  ['#', '(', '(', '+']        运算数栈为：  [72.0]
本次输入为：  )  运算符栈为：  ['#', '(', '(', '+']        运算 72.0 + 3.0 运算数栈为：  [75.0]
本次输入为：  )  运算符栈为：  ['#', '(', '(']     运算数栈为：  [75.0]
本次输入为：  *  运算符栈为：  ['#', '(']  运算数栈为：  [75.0]
本次输入为：  2  运算符栈为：  ['#', '(', '*']     运算数栈为：  [75.0]
本次输入为：  +  运算符栈为：  ['#', '(', '*']     运算 75.0 * 2.0 运算数栈为：  [150.0]
本次输入为：  +  运算符栈为：  ['#', '(']  运算数栈为：  [150.0]
本次输入为：  6  运算符栈为：  ['#', '(', '+']     运算数栈为：  [150.0]
本次输入为：  )  运算符栈为：  ['#', '(', '+']     运算 150.0 + 6.0        运算数栈为：  [156.0]
本次输入为：  )  运算符栈为：  ['#', '(']  运算数栈为：  [156.0]
本次输入为：  *  运算符栈为：  ['#']       运算数栈为：  [156.0]
本次输入为：  2  运算符栈为：  ['#', '*']  运算数栈为：  [156.0]
本次输入为：  #  运算符栈为：  ['#', '*']  运算 156.0 * 2.0        运算数栈为：  [312.0]
本次输入为：  #  运算符栈为：  ['#']       运算数栈为：  [312.0]


运算结果为：  312.0
```

6.((a+b)*c-d)/e

```
PS H:\data structures> python biaodashi2.py
输入表达式:
((a+b)*c-d)/e
本次输入为：  (  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  (  运算符栈为：  ['#', '(']  运算数栈为：  []
本次输入为：  a  运算符栈为：  ['#', '(', '(']     输入a的值
8
运算数栈为：  []
本次输入为：  +  运算符栈为：  ['#', '(', '(']     运算数栈为：  [8.0]
本次输入为：  b  运算符栈为：  ['#', '(', '(', '+']        输入b的值
6
运算数栈为：  [8.0]
本次输入为：  )  运算符栈为：  ['#', '(', '(', '+']        运算 8.0 + 6.0  运算数栈为：  [14.0]
本次输入为：  )  运算符栈为：  ['#', '(', '(']     运算数栈为：  [14.0]
本次输入为：  *  运算符栈为：  ['#', '(']  运算数栈为：  [14.0]
本次输入为：  c  运算符栈为：  ['#', '(', '*']     输入c的值
5
运算数栈为：  [14.0]
本次输入为：  -  运算符栈为：  ['#', '(', '*']     运算 14.0 * 5.0 运算数栈为：  [70.0]
本次输入为：  -  运算符栈为：  ['#', '(']  运算数栈为：  [70.0]
本次输入为：  d  运算符栈为：  ['#', '(', '-']     输入d的值
4
运算数栈为：  [70.0]
本次输入为：  )  运算符栈为：  ['#', '(', '-']     运算 70.0 - 4.0 运算数栈为：  [66.0]
本次输入为：  )  运算符栈为：  ['#', '(']  运算数栈为：  [66.0]
本次输入为：  /  运算符栈为：  ['#']       运算数栈为：  [66.0]
本次输入为：  e  运算符栈为：  ['#', '/']  输入e的值
11
运算数栈为：  [66.0]
本次输入为：  #  运算符栈为：  ['#', '/']  运算 66.0 / 11.0        运算数栈为：  [6.0]
本次输入为：  #  运算符栈为：  ['#']       运算数栈为：  [6.0]


运算结果为：  6.0
```

7.((a+b)*c-d)/e (float型的运算)

```
PS H:\data structures> python biaodashi2.py
输入表达式:
((a+b)*c-d)/e
本次输入为：  (  运算符栈为：  ['#']       运算数栈为：  []
本次输入为：  (  运算符栈为：  ['#', '(']  运算数栈为：  []
本次输入为：  a  运算符栈为：  ['#', '(', '(']     输入a的值
8.5
运算数栈为：  []
本次输入为：  +  运算符栈为：  ['#', '(', '(']     运算数栈为：  [8.5]
本次输入为：  b  运算符栈为：  ['#', '(', '(', '+']        输入b的值
4.3
运算数栈为：  [8.5]
本次输入为：  )  运算符栈为：  ['#', '(', '(', '+']        运算 8.5 + 4.3  运算数栈为：  [12.8]
本次输入为：  )  运算符栈为：  ['#', '(', '(']     运算数栈为：  [12.8]
本次输入为：  *  运算符栈为：  ['#', '(']  运算数栈为：  [12.8]
本次输入为：  c  运算符栈为：  ['#', '(', '*']     输入c的值
2.4
运算数栈为：  [12.8]
本次输入为：  -  运算符栈为：  ['#', '(', '*']     运算 12.8 * 2.4 运算数栈为：  [30.72]
本次输入为：  -  运算符栈为：  ['#', '(']  运算数栈为：  [30.72]
本次输入为：  d  运算符栈为：  ['#', '(', '-']     输入d的值
1.5
运算数栈为：  [30.72]
本次输入为：  )  运算符栈为：  ['#', '(', '-']     运算 30.72 - 1.5        运算数栈为：  [29.22]
本次输入为：  )  运算符栈为：  ['#', '(']  运算数栈为：  [29.22]
本次输入为：  /  运算符栈为：  ['#']       运算数栈为：  [29.22]
本次输入为：  e  运算符栈为：  ['#', '/']  输入e的值
14.61
运算数栈为：  [29.22]
本次输入为：  #  运算符栈为：  ['#', '/']  运算 29.22 / 14.61      运算数栈为：  [2.0]
本次输入为：  #  运算符栈为：  ['#']       运算数栈为：  [2.0]


运算结果为：  2.0
```

七、附录

源程序文件名清单：

biaodashi2.py #主程序