华侨大学工学院实验报告

课程名称: ______Pyhton 编程(实验)

实验项目名称:			Python 课程设计(4 课时)		
	兴	吃.	工学院		
	4	19/L:	工子例		
	专业	班级:	物联网工程1班		
	姓	名:	李昊唐		
	学	号:	1995131017		

指导教师: ______王佳斌_____

预习报告

一、 实验目的

使用 python 编写计算器程序

二、 实验仪器

三、 实验原理

- 1、 用正则表达式处理字符串,只提取其中的数字和运算符,并转 换成列表
- 2、 编写一个函数,处理没有括号的基本运算的基本表达式
- 3、 再写一个函数递归处理带有括号的函数,先计算最内部括号中的表达式,然后将最内部的括号替换为计算后的结果,在递归外部一层的,最后返回的就是所需的结果

预习报告

- 1. 字符串的处理
- 2. 正则表达式的运用
- 3. 函数递归

四、实验内容及步骤

基本功能:

- 1、用户输入一个类似这样 3*(4+ 50) -((100 + 40)*5/2- 3*2* 2/4+9)*(((3+4)-4)-4) 这样的表达式,假设表达式里面除了包含空格、'+'、'-'、'*'、'/'和括号再无其他特殊符号,然后自己动手写代码解析其中的表达式,实现加减乘除四则基本运算。
- 2、需要具备基本人机交互界面,可以是文本模式的,也可以是图形模式的。该人机界面可以方便数学式子的输入。

增强功能:

可进行包括三角函数,对数运算等科学计算方法。

最后得出的结果与真实的计算机所算的结果必须一致。

实验报告

五、 实验原始数据

```
import re
class Expression:
   def __init__(self):
      self.raw = input('请输入表达式:')
      self.format expression = None
      self. expression = []
      self.result = 0
   def format(self):
      self.format expression = re.sub('', '', self.raw)
      expression = [i for i in re.split('(-\d+\.?\d*)',
self.format expression) if i]
       for item in expression:
          if len(self. expression) == 0 and re.search('^-
d+\.?\d*$', item):
             self. expression.append(item)
             continue
          if len(self. expression) > 0 and re.search('[+\-*/(]$'],
self. expression[-1]):
             self. expression.append(item)
             continue
          item split = [i for i in re.split('([+\-*/()])', item)
if i]
          self. expression += item split
   def calculate(self):
      num stack = []
      op_stack = []
       for e in self._expression:
          operator = self.is operator(e)
          if not operator:
             num stack.append(float(e))
          else:
             while True:
                 if len(op_stack) == 0:
                    op stack.append(e)
                    break
                 tag = self.decision(op stack[-1], e)
```

实验报告

六、 数据处理

```
if tag == -1:
                 op stack.append(e)
                break
             elif tag == 0:
                 op stack.pop()
                break
             elif tag == 1:
                 op = op stack.pop()
                 num2 = num stack.pop()
                 num1 = num stack.pop()
                 num stack.append(self.calc(num1, num2, op))
   while len(op stack) != 0:
      op = op stack.pop()
      num2 = num stack.pop()
      num1 = num_stack.pop()
      num stack.append(self.calc(num1, num2, op))
   self.result = num stack[0]
@staticmethod
def calc(n1, n2, operator):
   result = 0
   if operator == "+":
      result = n1 + n2
   if operator == "-":
      result = n1 - n2
   if operator == "*":
      result = n1 * n2
   if operator == "/":
      result = n1 / n2
   return result
@staticmethod
def is operator(e):
   opes = ['+', '-', '*', '/', '(', ')']
   return True if e in opes else False
@staticmethod
def decision(tail_op, now_op):
   rate1 = ['+', '-']
   rate2 = ['*', '/']
   rate3 = ['(']
```

实验报告

七、实验结论及分析讨论

```
rate4 = [')']
      if tail op in rate1:
          if now_op in rate2 or now_op in rate3:
             return -1
          else:
             return 1
      elif tail op in rate2:
          if now_op in rate3:
            return -1
          else:
             return 1
      elif tail_op in rate3:
          if now op in rate4:
            return 0
          else:
            return -1
      else:
         return -1
if __name__ == '__main__':
   exp = Expression()
   exp.format()
   print('输入为: ', exp.format_expression)
   exp.calculate()
   print('结果: ', exp.result)
```

预习报告成绩	实验报告成绩	实验操作成绩	总成绩