Python计算课后题

20121034 胡才郁

第1章 基础知识

1.1 简单说明如何选择正确的Python版本。
在选择Python的时候，一定要先考虑清楚自己学习Python的目的是什么，打算做哪方面的开发，有哪些扩展 库可用，这些扩展库最高支持哪个版本的Python，是Python 2.x还是Python 3.x。

1.2 为什么说Python采用的是基于值的内存管理模式?
内存中只有一份值，不同变量指向该值

1.3 解释Python中的运算符/和//的区别。
在Python中/表示普通除法（也叫真除法），其结果是实数，而//表示整除，得到的结果是整数，并且自动向下取整。

1.4 在Python中导入模块中的对象有哪几种方式?
常用的有三种方式，分别为

```
import 模块名 [as 别名]
from 模块名 import 对象名[ as 别名]
from math import *
```

1.5 _pip_是目前比较常用的Python扩展库管理工具。

1.6 解释Python脚本程序的__name__变量及其作用。
作为模块导入时，其值为程序文件名。

```
if __name__ == "__main__":
```

1.7 运算符%_可以_浮点数进行求余数操作。

1.8 一个数字5_是_合法的Python表达式。

1.9 在Python 2.x中，input()函数接收到的数据类型由_界定符_确定，而在Python3.x 中该函数则认为接收到的用户输入数据一律为_字符串_。

1.10 编写程序，用户输入一个三位以上的整数，输出其百位以上的数字。例如用户输入1234，则程序输出

```
if __name__ == "__main__":
    x = input('输入3位以上的数字: ')
    print(x // 100)
```

第2章 Python数据结构

2.1 为什么应尽量从列表的尾部进行元素的增加与删除操作?
效率快，类比于数组中在队尾的插入与删除操作，效率为O(n)

2.2 range()函数在Python 2.x中返回一个_列表_，而在Python 3.x中的range()函数返回一个_range对象 _。

2.3 编写程序，生成包含1000个0到100之间的随机整数，并统计每个元素的出现次数。

```
import random
from collections import Countr
x = [ranfom.randint(100) for i in range(100)]
counter = Counter(x)
d = dict(counter)
```

2.4 表达式“[3] in [1,2,3,4]”的值为_False_。

2.5 编写程序，用户输入一个列表和2个整数作为下标，然后输出列表中介于2个下标之间的元素组成的子列表。例如用户输入[1,2,3,4,5,6]和2,5，程序输出[3,4,5,6]。

```
x = eval(input())
start, end = eval(input())
print(x[start: end + 1])
```

2.6 列表对象的sort()方法用来对列表元素进行原地排序，该方法的返回值为_None_。

2.7 列表对象的_remove()_方法删除首次出现的指定元素，如果列表中不存在要删除的元素，则抛出异 常。

2.8 假设列表对象sList的值为[3, 4, 5, 6, 7, 9, 11, 13, 15, 17]，那么切片aList[3:7]得到的值是_[6, 7, 9, 11]_。

2.9 设计一个字典，并编写程序，用户输入内容作为键，然后输出字典中对应的值，如果用户输入的键不存 在，则输出“您输入的键不存在！”

```
dic = {1 : 'a', 2 : 'b'}
a = input()
print(dic.get(a, "您输入的值不存在"))
```

2.10 编写程序，生成包含20个随机数的列表，然后将前10个元素升序排列，后10个元素降序排列，并输出结果。

```
from random import randint
x = [randint(100) for i in range(20)]
x[:10] = sorted(x[:10])
x[10:] = sorted([10:], reverse = True)
print(x)
```

2.11 在Python中，字典和集合都是用一对_大括号_作为定界符，字典的每个元素有两部分组成，即_键_和_值_，其中_键_不允许重复。

2.12 使用字典对象的_items()_方法可以返回字典的“键-值对”列表，使用字典对象的_keys()_方法可以返回字典的“键”列表，使用字典对象的_values()_方法可以返回字典的“值”列表。

2.13 假设有列表a = ['name','age','sex']和b = ['Dong',38,'Male']，请使用一个语句将这两

个列表的内容转换为字典，并且以列表a中的元素为键，以列表b中的元素为值，这个语句可以写为 _c = dict(zip(a,b))_

2.14 假设有一个列表a，现要求从列表a中每3个元素取1个，并且将取到的元素组成新的列表b，可以使用语句_b = a[::3]_

2.15 使用列表推导式生成包含10个数字5的列表，语句可以写为_[5 for i in range(10)]_

2.16 _不可以_使用del命令来删除元组中的部分元素。

第3章 选择结构与循环结构

3.1 分析逻辑运算符“or”的短路求值特性。

假设有表达式“表达式1 or 表达式2”，如果表达式1的值等价于True，那么无论表达式2的值是什么，整个表达式的值总是等价于True。因此，不需要再计算表达式2的值。

3.2 编写程序，运行后用户输入4位整数作为年份，判断其是否为闰年。如果年份能被400整除，则为闰年；如果年份能被4整除但不能被100整除也为闰年。

```
x = eval(input('请输入一个年份:'))
if x % 400 == 0 or (x % 4== 0 and not x % 100 == 0):
    print('Yes')
else:
    print('No')
```

3.3 Python提供了两种基本的循环结构：_for_和_while_。

3.4 编写程序，生成一个包含50个随机整数的列表，然后删除其中所有奇数。

```
import random
x = [random.randint(0,100) for i in range(50)]
print(x)
for i in range(len(x))[::-1]:
    if x[i]%2 == 1:
        del x[i]
print(x)
```

3.5 编写程序，生成一个包含20个随机整数的列表，然后对其中偶数下标的元素进行降序排列，奇数下标的元素不变。

```
import random
```

```
    x = [random.randint(0,100) for i in range(20)]
    print(x)
    x[::2] = sorted(x[::2], reverse=True)
    print(x)
```

3.6 编写程序，用户从键盘输入小于1000的整数，对其进行因式分解。例如，10=2×5，60=2×2×3×5。

```
    x = input('请输入小于1000的整数:')
    x = eval(x)
    t = x
    i = 2
    result = []
    while True:
        if t == 1:
            break
        if t % i == 0:
            result.append(i)
            t = t // i
        else:
            i += 1
    print(x,'=', '*'.join(map(str, result)))
```

3.7 编写程序，至少使用2种不同的方法计算100以内所有奇数的和。

```
    方法1: print(sum([i for i in range(1,100) if i % 2 == 1]))
    方法2: print(sum(range(1, 100)[::2]))
```

3.8 编写程序，输出所有由1、2、3、4这4个数字组成的素数，并且在每个素数中每个数字只使用依次。

```
    from itertools import permutations
    def isPrime(n):
        if n == 1:
            return False
        if n == 2:
            return True
        if n%2 == 0:
            return False
        for i in range(3, int(n ** 0.5) + 1, 2):
            if n%i == 0:
                return False
            return True

    if __name__ == "__main__":
        digits = (1, 2, 3, 4)
        for i in range(1, len(digits)+1):
            for number in permutations(digits, i):
                number = int(''.join(map(str, number)))
                if isPrime(number):
                    print(number)
```

3.9 编写程序，实现分段函数计算，如下表所示。

```
"""
-------------------------
|     x      |     y    |
-------------------------
|     x<0    |     0    |
|  0≤x<5     |     x    |
|  5≤x<10    |   3x-5   |
| 10≤x<20    |  0.5x-2  |
| 20≤x       |     0    |
-------------------------
"""

x = input('Please input x:')
x = eval(x)
if 0<= x < 5:
    print(x)
    elif 5<= x < 10:
        print(3 * x - 5)
        elif 10<= x < 20:
```

```
            print(0.5 * x - 2)
        else:
            print(0)
```

第4章　字符串与正则表达式

4.1 假设有一段英文，其中有单独的字母“I”误写为“i”，请编写程序进行纠正。

```
x = "i am a human, and i am a man."
x = x.replace('i ','I ')
x = x.replace(' i ',' I ')
print(x)
```

4.2 假设有一段英文，其中有单词中间的字母“i”误写为“I”，请编写程序进行纠正。

```
import re
x = "I am a busInessman."
print(x)
result = re.sub(r'\BI\B', 'i', x)
print(result)
```

4.3 有一段英文文本，其中有单词连续重复了2次，编写程序检查重复的单词并只保留一个。例如文本内容为“This is is a desk.”，程序输出为“This is a desk.”

(1) 方法一

```
import re
x = 'This is is a desk.'
pattern = re.compile(r'\b(\w+)(\s+\1){1,}\b')
matchResult = pattern.search(x)
x = pattern.sub(matchResult.group(1),x)
print(x)
```

(2) 方法二

```
import re
x = 'This is is a desk.'
pattern = re.compile(r'(?P<f>\b\w+\b)\s(?P=f)')
matchResult = pattern.search(x)
x = x.replace(matchResult.group(0),matchResult.group(1))
print(x)
```

4.4 简单解释Python的字符串驻留机制。

字符串驻留是一种仅保存一份相同且不可变字符串的方法。这种方法可以节省大量内存，并且在字符串比较时，非驻留比较效率o(n)，驻留时比较效率o(1)。系统维护interned字典，记录已被驻留的字符串对象。当字符串对象a需要驻留时，先在interned检测是否存在，若存在则指向存在的字符串对象，a的引用计数减1；若不存在，则记录a到interned中。

4.5 编写程序，用户输入一段英文，然后输出这段英文中所有长度为3个字母的单词。

```
import re
x = input('请输入一段英文')
pattern = re.compile(r'\b[a-zA-Z]{3}\b')
print(pattern.findall(x))
```

第5章　字符串与正则表达式

5.1 运行 5.3.1小节最后的示例代码，查看结果并分析原因。

代码如下：

```
"""
def demo(newitem, old_list=None):
    if old_list is None:
        old_list = []
    old_list.append(newitem)
    return old_list
print(demo('5', [1, 2, 3, 4]))
print(demo('aaa', ['a', 'b']))
print(demo('a'))
print(demo('b'))
"""
```

结果：

```
[1, 2, 3, 4, '5']
['a', 'b', 'aaa']
['a']
['b']
```

原因：
　　对于函数的默认值参数只会被处理一次，下次再调用函数并且不为默认值参数赋值时会继续使用上一次的结果，对于列表这样的结构，如果调用函数时为默认值参数的列表插入或删除了元素，将会得到保留，从而影响下一次调用。

5.2 编写函数，判断一个整数是否为素数，并编写主程序调用该函数。

```
import math
def IsPrime(v):
    n = int(math.sqrt(v) + 1)
    for i in range(2, n):
        if v % i == 0:
            return 'No'
        else:
            return 'Yes'
if __name__ == "__main__":
    print(IsPrime(37))
    print(IsPrime(60))
    print(IsPrime(113))
```

5.3 编写函数，接收一个字符串，分别统计大写字母、小写字母、数字、其他字符的个数，并以元组的形式返回结果。

```
def demo(v):
    capital = little = digit = other =0
    for i in v:
        if 'A'<=i<='Z':
            capital+=1
            elif 'a'<=i<='z':
                little+=1
                elif '0'<=i<='9':
                    digit+=1
                    else:
                        other+=1
                        return (capital,little,digit,other)
if __name__ == "__main__":
    x = 'jdsofj12041JDSAadnAQQ**77%#@'
    print(demo(x))
```

5.4 在函数内部可以通过关键字_global_来定义全局变量。

5.5 如果函数中没有return语句或者return语句不带任何返回值，那么该函数的返回值为_None_。

5.6 调用带有默认值参数的函数时，不能为默认值参数传递任何值，必须使用函数定义时设置的默认值。　（错）

5.7 在Python程序中，局部变量会隐藏同名的全局变量吗？请编写代码进行验证。
会。

```
def demo():
    a=3
    print a
    a=5
    demo()#此处结果为3
```

5.8 lambda表达式只能用来创建匿名函数，不能为这样的函数起名字。（错）

5.9 编写函数，可以接收任意多个整数并输出其中的最大值和所有整数之和。

```
def demo(*v):
    print(v)
    print(max(v))
    print(sum(v))
```

5.10 编写函数，模拟内置函数sum()。

```
def Sum(v):
    s = 0
    for i in v:
        s += i
        return s
```

5.11 包含_yield_语句的函数可以用来创建生成器对象。

5.12 编写函数，模拟内置函数sorted()。

```
def Sorted(lst, reverse=False):
    lst = lst[:]
    length = len(lst)
    for i in range(0, length):
        flag = False
        for j in range(0, length - i - 1):
            exp = 'lst[j] > lst[j+1]'
            if reverse:
                exp = 'lst[j] < lst[j+1]'
                if eval(exp):
                    lst[j], lst[j + 1] = lst[j + 1], lst[j]
                    flag = True
                    if not flag:
                        break
                        return lst
```

第6章 面向对象程序设计

6.1 继承6.5节例6-2中的Person类生成Student类，填写新的函数用来设置学生专业，然后生成该类对象,并显示信息。

```
import types
class Person(object):
    def __init__(self, name = '', age = 20, sex = 'man'):
        self.setName(name)
        self.setAge(age)
        self.setSex(sex)

    def setName(self, name):
        if not isinstance(name,str):
            print('name must be string.')
            self.__name = name

    def setAge(self, age):
        if not isinstance(age,int):
            print('age must be integer.')
            self.__age = age


    def setSex(self, sex):
         if sex != 'man' and sex != 'woman':
            print('sex must be "man" or "woman"')
            self.__sex = sex

    def show(self):
        print(self.__name)
        print(self.__age)
        print(self.__sex)

class Student(Person):
    def __init__(self, name='', age = 30, sex = 'man', major = 'Computer'):
        super(Student, self).__init__(name, age, sex)
        self.setMajor(major)

    def setMajor(self, major):
        if not isinstance(major, str):
            print('major must be a string.')
            self.__major = major

    def show(self):
        super(Student, self).show()
        print(self.__major)
```

```
if __name__ =='__main__':
    zhangsan = Person('Zhang San', 20, 'woman')
    zhangsan.show()
    lisi = Student('Li Si',13, 'man', 'Math')
    lisi.show()
```

6.2 设计一个三维向量类，并实现向量的加法、减法以及向量与标量的乘法和除法运算

```
class Vecter3:
    def __init__(self, x=0, y=0, z=0):
        self.X = x
        self.Y = y
        self.Z = z

    def __add__(self, n):
        r = Vecter3()
        r.X = self.X + n.X
        r.Y = self.Y + n.Y
        r.Z = self.Z + n.Z
        return r

    def __sub__(self, n):  # 调用__add__方法，
        tmp = Vecter3(-n.x, -n.y, -n.z)
        self = self + tmp
        return self

    def __mul__(self, n):
        r = Vecter3()
        r.X = self.X * n
        r.Y = self.Y * n
        r.Z = self.Z * n
        return r

    def __truediv__(self, n):
        r = Vecter3()
        r.X = self.X / n
        r.Y = self.Y / n
        r.Z = self.Z / n
        return r


    def show(self):
        print((self.X,self.Y,self.Z))

if __name__ == "__main__":
    v1 = Vecter3(1, 2, 3)
    v2 = Vecter3(9, 10, 11)
    v3 = v1 + v2
    v3.show()
    v4 = v1 - v2
    v4.show()
    v5 = v1 * 3
    v5.show()
    v6 = v1 / 2
    v6.show()
```

6.3 面向对象程序设计的三要素分别为_封装_、_继承_和_多态_。

6.4 简单解释Python中以下划线开头的变量名特点。

在Python中，以下划线开头的变量名有特殊的含义，尤其是在类的定义中。用下划线作为变量前缀和后缀来表示类的特殊成员：

_xxx：这样的对象叫做保护变量，不能用'from module import *'导入，只有类对象和子类对象能访问这些变量；

__xxx__：系统定义的特殊成员名字；

__xxx：类中的私有成员，只有类对象自己能访问，子类对象也不能访问到这个成员，但在对象外部可以通过“对象名._类名__xxx”这样的特殊　　方式来访问。Python中没有纯粹的类似于C++、JAVA等语言中的私有成员

6.5 与运算符“**”对应的特殊方法名为__pow__()，与运算符“//”对应的特殊方法名为 __floordiv__()。

6.6 假设a为类A的对象且包含一个私有数据成员“__value”，那么在类的外部通过对象a直接将其私有数据 成员“__value”的值设置为3的语句可以写作a._A__value = 3。