22121630　汪江豪

第一章 基础知识

1. 简单说明如何选择正确的python版本。

在选择python的时候，要先考虑自己学习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。这样才能事半功倍，同时，当python新版本推出后，不要急于更新，要确定自己所使用的扩展库也推出了较新版本后再进行更新。但当下python3是大势所趋，如果是为了尝试一种新语言，可以选择python3系列最高版本。

2. 为什么说python采用的是基于值的内存管理模式?

如果为不同变量赋值相同值，则在内存中只有一份该值，多个变量指向同一块内存地址。如：

```
x = 3
y = 3
id(x)
id(y)
10417624
10417624
```

3. 解释python中的运算符/和//的区别。

/表示普通除法（真除法），结果是实数；//表示整除，结果是整数，并自动向下取整。

4. python中导入模块中的对象有哪几种方式?

三种方式：

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

5. (pip)是目前比较常用的python扩展库管理工具。

6. 解释python脚本程序的__name__变量及其作用。

每个python程序都有一个name变量，用来表示程序的运行。当作为模块导入时，name变量的值等于程序文件名，当直接运行程序时其值为字符串main。

7. 运算符%（可以）对浮点数进行求余数操作。

8. 一个数字5（是）合法的python表达式。

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python 3.x中，该函数则认为接收到的用户数据一律为（字符串）。

10. 编写程序，用户输入一个三位数以上的整数，输出其百位以上的数字。例如用户输入1234，则程序输出12。（提示：使用整除运算）

```
x = input('输入3位以上的数字：')
if len(x) >= 3:
    x = int(x)
    print(x // 100)
else
    print('输入格式错误。')
```

第二章 python数据结构

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?

增加或删除元素时，列表对象自动进行内存扩展或收缩。从中部操作需要遍历列表，复杂度O(n)，从尾部操作复杂度O(1)，效率更高。

2. range()函数在python2中返回一个（列表），而在python3中返回一个（range对象）。

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
x = [random.randint(0,100) for i in range(1000)]
x = set(x) # 使用集合提高查询效率
for i in x:
    print(i,':',x.count(i))
```

4. 表达式"[3] in [1,2,3,4]"的值为（False）。

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 = input('Please input a list:')
x = list(map(int, x.split()))
y = input('Please input 2 index')
y = list(map(int, y.split()))
print(x[y[0]:y[1]+1])
```

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）。

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。

8. 假设列表对象aList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]）。

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！"

```
d = {1:'a', 2:'b', 3:'c', 4:'d', '666':'this is str', 666: 'this is number'}
v = input('Please input a key:')
v = eval(v) # eval()函数能将字符串转为对应类型
print(d.get(v,'您输入的的键不存在'))
```

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。

```
import random
'''
列表推导式格式：expression for varible in iterable
expression是每个元素都要执行的操作，varible是每次迭代中使用的变量，iterable是可迭代的序列
'''
x = [random.randint(0,1000) for i in range(20)]
print(x)

x[:10] = sorted(x[:10])
x[-10:] = sorted(x[-10:], reverse = True)
print(x)
```

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（key）和（value），其中（key）不允许重复。

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“间”列表，使用字典对象的（values()）方法可以返回字典的“值”列表。

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))
```

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]
```

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]
```

16. （不可以）使用del命令来删除元组中的部分元素。

第三章 选择结构与循环结构

1. 分析逻辑运算符“or”的短路求值特性。

```
[表达式1] or [表达式2]
# 如果表达式1的值为True，则无论表达式2的值是什么，整体返回True
```

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 = input('Please input year:')
x = eval(x)
if x % 400 == 0 or x % 4 == 0 and x % 100 != 0:
    print('yes')
else:
    print('no')
```

3. python提供了两种基本的循环结构（for循环）和（while循环）。

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)
x = [num for num in x if num % 2 ==0] # 列表推导式选出偶数
print(x)
```

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。（提示：使用切片）

```
import random
x = [random.randint(0,100) for i in range(20)]
print(x)

x[::2] = sorted(x[::2], reverse = True)
for pair in enumerate(x): # enumerate自动生成索引
    print(pair)
```

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('Please input a number(number < 1000):')
x = eval(x)
t = x
i = 2
factors = []
while True:
    if t == 1:
        break;
    if t % i == 0:
        factors.append(i)
        t /= i
    else:
        i += 1
print(x, '=', '*'.join(map(str, factors)))
```

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。

```
print(sum([i for i in range(1, 100) if i % 2 == 1])) # 列表推导式
print(sum(list(range(1,100))[::2])) # 切片操作
```

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一次。

```
from itertools import permutations
from math import sqrt
def is_prime(x):
    if x < 2:
        return False
    for i in range(2, int(sqrt(x) + 1)):
        if x % i == 0:
            return False;
    return True
```

```
numbers = [1,2,3,4]
primes = set()
for r in range(1, 5):
    for it in permutations(numbers, r): # 生成长度分别为1，2，3，4的1，2，3，4的全排列
        t = int(''.join(map(str, it)))
        if is_prime(t):
            primes.add(t)
for prime in sorted(primes):
    print(prime)

```

9. 编写程序，实现分段函数计算。

```
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)
```

第四章 字符串与正则表达式

1. 假设有一段英文，其中有单独的字母“l”误写成“i”，请编写程序进行纠正。

```
import re
x = "i am a student,i am a person, and i wanna to say hello."
result = re.sub(r'\bi\b', 'I', x) # 要匹配的正则表达式、替换后的内容、原始字符串
print(result)
```

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

```
import re
x = "thIs am a strIng, aIm prIorItIes"
result = re.sub(r'\BI\B', 'i', x) # 要匹配的正则表达式、替换后的内容、原始字符串
print(result)
```

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.“。

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

```

4. 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('Please input a string:')
pattern = re.compile(r'\b[a-zA-Z]{3}\b')
print(pattern.findall(x))
```

第五章 函数设计与使用

1. 运行5.3.1小节最后的示例代码（函数参数默认值只在函数定义时被处理一次），查看结果并分析原因。

```
i = 3
def f(n = i):
    print(n)
f()
i = 5
f()
# 两个输出值都是3
```

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(x):
    n = int(math.sqrt(x) + 1)
    for i in range(2, n):
        if x % i == 0:
            return 'No'
    return 'Yes'
print(IsPrime(7))
print(IsPrime(10))
print(IsPrime(13))
# 输出 Yes No Yes
```

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 func(s):
    capital = little = digit = other = 0
    for i in s:
        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)
print(func('Hello123!'))
```

4. 在函数内部可以通过关键字（global）来定义全局变量。

5. 如果函数中没有return语句或return语句不带任何返回值，那么该函数的返回值为(None)。

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）。

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()
print(a)
# 输出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）

可以将该函数赋值给一个变量。

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 func(*a):
    print(a)
    print(max(a))
    print(sum(a))
func(1,2,3)
func(1,2,3,4)
func(*[i for i in range(1,101)])
```

10. 编写函数，模拟内置函数sum()

```
def Sum(a):
    s = 0
    for i in a:
        s += i
    return s
x = [1,2,3,4,5]
print(Sum(x))
x = [i for i in range(1, 101)]
print(Sum(x))
```

11. 包含（yield）语句的函数可以用来创建生成器对象。

12. 编写函数，模拟内置函数sorted()。

```
def Sorted(q, reverse = False):
    length = len(q)
    for i in range(length):
        for j in range(length - 1 - i):
            if (q[j] > q[j + 1] and not reverse) or (q[j] < q[j + 1] and
reverse):
                q[j], q[j + 1] = q[j + 1], q[j]
```

```
    return q
print(Sorted([5, 4, 3, 2, 1]))
```

第六章 面向对象程序设计

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。

```
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):
            raise TypeError('name must be a string')
            print('name must be a string')
            return
        self.__name = name # __表示私有变量

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

    def setSex(self, sex):
        if sex != 'man' and sex != 'woman':
            print('sex must be "man" or "woman"')
            return
        self.__sex = sex
    def show(self):
        print('name:', self.__name)
        print('age:', self.__age)
        print('sex:',self.__sex)

class Student(Person):
    #调用基类构造方法初始化基类的私有数据成员
    def __init__(self, name='', age = 30, sex = 'man', major = 'Computer'):
        super(Student, self).__init__(name, age, sex)
        self.setMajor(major) #初始化派生类的数据成员

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

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

Bob = Person('Zhang San', 19, 'man')
Bob.show()
```

```
Tom = Student('Li Si',32, 'man', 'Math')
Tom.show()
'''
结果如下:
name: Zhang San
age: 19
sex: man
name: Li Si
age: 32
sex: man
major: Math
'''
```

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触发运算。

```
class Vector:
    def __init__(self, x=0, y=0, z=0):
        self.x = x
        self.y = y
        self.z = z

    def __add__(self, n):
        r = Vector()
        r.x = self.x + n.x
        r.y = self.y + n.y
        r.z = self.z + n.z
        return r

    def __sub__(self, n):
        r = Vector()
        r.x = self.x - n.x
        r.y = self.y - n.y
        r.z = self.z - n.z
        return r

    def __mul__(self, n):
        r = Vector()
        r.x = self.x * n.x
        r.y = self.y * n.y
        r.z = self.z * n.z
        return r

    def __truediv__(self, n):
        r = Vector()
        r.x = self.x / n.x
        r.y = self.y / n.y
        r.z = self.z / n.z
        return r

    def __floordiv__(self, n):
        r = Vector()
        r.x = self.x // n.x
        r.y = self.y // n.y
        r.z = self.z // n.z
        return r
```

```
    def show(self):
        print(f'({self.x},{self.y},{self.z})')

v1 = Vector(1, 2, 3)
v2 = Vector(4, 5, 6)
v3 = v1 + v2
v3.show()
v4 = v1 - v2
v4.show()
v5 = v1 * v2
v5.show()
v6 = v1 / v2
v6.show()
v7 = v1 // v2
v7.show()
'''
结果：
(5,7,9)
(-3,-3,-3)
(4,10,18)
(0.25,0.4,0.5)
(0,0,0)
'''
```

3. 面向对象程序设计的三要素分别是（封装）、（继承）和（多态）。

4. 简单解释python中以下划线开头的变量名特点：

_xxx表示对象的保护变量，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该成员，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中没有像C++的纯粹意义上的私有成员。

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. 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"的值设为3的语句可以写作（a.__value = 3）。