Python 1~6章 课后习题

第一章 习题

1.简单说明如何选择正确的Python版本?

应根据自己所学习的需求和目的选择最合适的版本。并不一定版本号越大，版本就越新。例如，Python2.7.9比Python3.2.6晚几个月发布。同时，也不一定使用最新的版本就是最合适的，因为有些机器并不一定兼容Python3.x的版本。

而对于不知道如何选择python版本的初学者，可以选择Python3.x的最新版作为学习版本。因为Python3.x是大势所趋，也比较好使用。

2.为什么说Python采用的是基于值的内存管理模式?

因为Python对于不同变量赋相同值，只会在内存中保存一份值的信息。两个变量指向同一个地址单元。

整数

因为对于[-5~256]之间的整数，不同变量但是值相同，则他们的id地址是相同的。但是对于范围外的整数，则无效。

```
# 范围内
>>> x = -6
>>> y = -6
>>> id(x)
2166372015600
>>> y = -6
>>> id(y)
2166372015504
# 范围外
>>> x = 10
>>> y = 10
>>> id(x)
2166333008464
>>> id(y)
2166333008464
```

复数

对于复数，这个并不成立

```
>>> x = 3 + 1j
>>> y = 3 + 1j
>>> id(x)
2166372015472
>>> id(y)
2166372015600
```

字符串

```
>>> x = 'abcsdwef'
>>> y = 'abcsdwef'
>>> id(x)
2166372908144
>>> id(y)
2166372908144
```

Python具有自动内存管理功能，会跟踪所有的值。（有看到资料说：Python会记录每一个变量的引索值，当引索值为0时，则自动删除）

3.解释Python中的运算符/和//的区别。

/：得到的结果永远是浮点数，正常除法

```
>>> 5/10
0.5
>>> 10/5
2.0
>>> 1/3
0.3333333333333333
```

//(地板除)：结果均是整数,取整数商

```
>>> 5//10
0
>>> 10//5
2
>>> 50//2
25
```

4.在Python中导入模块中的对象有哪几种方式?

import 模块名[as别名]

```
>>> import math as m
>>> m.sqrt(9)
3.0
```

from 模块名 import 对象名[as别名]

```
>>> from math import sqrt as f
>>> f(9)
3.0
```

5._____是目前比较常用的Python扩展库管理工具。

pip

6.解释Python脚本程序的__name__变量及其作用。

__name__是每个Python脚本程序自带的一个属性。如果脚本作为模块被导入，则其__name__属性的值自动设置为模块名。如果脚本独立运行，则其__name__属性值被自动设置为"**main**"

可以通过这控制，仅执行当本脚本为主线程的程序，而当本脚本为作为模块导入时，则不使用。

案例： 在同一目录下新建两个文件first.py和import_first.py

```
# first.py
print("I\'am the first")


if __name__ == "__main__"
    print("I\'am the second")
```

```
# import_first.py
import first
```

运行first.py时输出结果：

```
I'am the first
I'am the second
```

运行import_first.py时输出结果：

```
I'am the first
```

原因：运行first.py时，该程序的__name___是main；而后者运行是，则变成了__first__.

7.运算符%_____(可以、不可以)对浮点数进行求余数操作。

可以。

```
>>> 3.5%2
1.5
>>> 4.2 % 1.5
1.2000000000000002
```

8.一个数字5_____(是、不是)合法的Python表达式。（p13）

是

9.在Python2.x中，input()函数接收到的数据类型由_____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_____。

输入值使用的界定符；字符串类型。

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12（提示：使用整除运算）

```
# biggerThanThree.py
def getTop(x=input("Please enter a number(contain at least 4 number:)")):
    print(int(x[::]) // 100)
```

第二章 习题

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?

因为python中元素的增加与删除操作是原地操作，如果在列表的头部进行元素的删除与增加，需要将整个数组的元素移动。浪费时间，程序的运行效率降低。而尾部增加与删除，避免了这个问题。

2. range()函数在Python2.x 中返回一个_____, 而Python3.x 的range() 函数返回一个_____。

Python 2.x: 包含若干整数的列表 list

Python 3.x: 可迭代对象 Iterable

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

```
import random


def generateThousand():
    a = [random.randint(0, 101) for x in range(1000)]
    return sorted({x: a.count(x) for x in a}.items(), key=lambda x: x[0])


print(generateThousand())
```

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]。

```
'''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]。'''


def cutList(li, x, y):
    if x < 0 or x >= len(li) or y < 0 or y >= len(li):
        print('index is out of range.')
        return
    elif isinstance(x, int) and isinstance(y, int):
        print(li[x:y + 1])
        return


arr = input('please enter a list:')
li = [int(num) for num in arr[1:-1].split(',')]
print(li)
x = input('x:')
y = input('y:')
cutList(li, int(x), int(y))
```

6. 列表对象的sort()方法用来对列表元素进行原地排序，该函数返回值为_____。

没有返回值

7. 列表对象的_____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。

del

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”。

```
"""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”"""


def getValue():
    d = {'a': 0, 'b': 1, 'c': 2, 'd': 3}
    print(d)
    x = input('请输入键值: ')
```

```
    if d.get(x, -1) == -1:
        print("您输入的键不存在")
    else:
        print(d.get(x))
```

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。

```
"""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。"""


def sort_my():
    import random
    a = [random.randint(0, 10) for i in range(20)]  # 创建包含20个0-9的随机数列表
    print(sorted(a[:10], reverse=False) + sorted(a[:-10], reverse=True))    # 利用切片和sorted()函数实现排序


sort_my()
```

11. 在Python中，字典和集合都是用一对_____作为界定符，字典的每个元素有两部分组成，即_____和_____,其中_____不允许重复。

- 大括号 {}
- 键 key
- 值 value
- 键值 key

12. 使用字典对象的_____方法可以返回字典的“键-值对”列表，使用字典对象的_____方法可以返回字典的“键”列表，使用字典对象的_____方法可以返回字典的“值”列表。

- 使用item()方法，返回“键-值对”
- 使用keys()方法，返回“键”列表
- 使用values()方法，返回“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中的元素组成新的列表b，可以使用语句_____。

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，并且将渠道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. _____(可以、不可以)使用del命令来删除元组中的部分元素。

不可以。

第三章 习题

1. 分析逻辑运算符or的短路求值特性

or的原理是有一个为1，则表达式的判值为真。例如：对于`if condition1 or condition2 or...`，如果condition1为True，则整个条件语句为真，之后条件不再进行判断。

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。

```
"""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。"""


def isLunar(year):
    if year % 400 == 0 or (year % 4 == 0 and year % 100 != 0):  # 判断是否为闰年
        print(year, '是闰年')
        return
    else:
        print(year, '不是闰年')


year = input('请输入年份（4位）：')
isLunar(int(year))
```

3. Python提供了两种基本的循环结构: _____ 和 _____。

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，然后删除其中所以奇数（提示：从后向前删）。

```
"""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（提示：从后向前删）。"""
import random


def deleteOdd():
    a = [random.randint(0, 10) for i in range(50)]
    print(a)
    for i in a[::-1]:
        if i % 2 == 1:
            a.remove(i)
    print(a)
```

```
deleteOdd()
```

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）

```
"""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


def sort_odd():
    a = [random.randint(0, 10) for i in range(20)]  # 构造随机列表
    print('origin:', a)
    b = a[1::2]  # 取a中偶数位的数字，生成列表
    b = sorted(b, reverse=True)  # 降序排序
    i = 1   # 循环标记
    j = 0   # 循环标记
    while i < len(a):   # 进行列表成员替换
        a[i] = b[j]
        i += 2
        j += 1
    print('sorted:', a)
    print('even list:', b)
```

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，$10 = 2*5$, $60 = 2*2*3*5$。

```
"""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。"""


def getPrime():
    """利用filter()过滤和生成器，输出素数列表"""

    def _odd_iter():
        n = 1
        while True:
            n = n + 2
            yield n

    def _not_divisible(n):
        return lambda x: x % n > 0

    def primes():
        yield 2
        it = _odd_iter()  # 初始化序列
        while True:
            n = next(it)
            yield n
            it = filter(_not_divisible(n), it)  # 构造新序列
```

```
    a = []
    for n in primes():
        if n < 1000:
            a.append(n)
        else:
            break
    return a


def factorization(x):
    print(x, ' =', end=' ')
    p = getPrime()

    if x in p:      # 如果x时素数，则直接输出x
        print(x)
        return

    for i in p:     # 如果x可以被因式分解，则进入循环
        if x == 1:  # 因式分解完成，跳出循环
            break
        while x % i == 0:   # 寻找x的因子
            if x in p:      # 找到x的最后一个因子，输出该因子，并给x赋值为1
                print(x)
                x = 1
            else:
                print(i, ' * ', end=' ')
                x = x // i


x = input('请输入一个小于1000的正整数: ')
factorization(int(x))
```

7. 编写程序，至少使用两种不同的方法计算100以内所以奇数的和。

```
from functools import reduce
from random import random

"""利用sum函数"""


def method1():
    return sum([i for i in range(1, 100, 2)])


"""利用reduce函数"""


def method2():
    return reduce(lambda x, y: x + y, [i for i in range(1, 100, 2)])


"""利用if条件表达式"""
```

```
def method3():
    sum = 0
    i = 1
    while i < 100:
        sum += i
        i += 2
    return sum


print(method1())
print(method2())
print(method3())
#
# 结果都是 2500
```

8. 编写程序，输出所有由1、2、3、4这四个数字组成的素数，并且在每个素数中每个数字只使用一次。

```
"""8. 编写程序，输出所有由1、2、3、4这四个数字组成的素数，并且在每个素数中每个数字只使用一次。"""


def isPrime(n):
    """ 判断一个数是否为素数
        本例主要演示循环结构中else子句的用法"""
    import math

    m = math.ceil(math.sqrt(n) + 1)  # 取n开根号的值，+1是因为range是闭开区间
    for i in range(2, m):
        if n % i == 0 and i < n:
            return False
    else:
        return True


def findPrime():
    """ 利用循环求和，得到1,2,3,4的全排列序列"""
    digits = (1, 2, 3, 4)
    all = []
    for i in digits:
        for j in digits:
            if i == j:
                continue
            for k in digits:
                if i == k or j == k:
                    continue
                for m in digits:
                    if i == m or j == m or k == m:
                        continue
                    all.append(i * 1000 + j * 100 + k * 10 + m)
    prime = [i for i in all if isPrime(i)]
    print(prime)
    return
```

```
def findPrime2():
    """利用字符串拼接 和列表生成式，得到1,2,3,4的全排序序列"""
    digits = ['1', '2', '3', '4']
    all = [x + y + z + m for x in digits for y in digits for z in digits for m
in digits \
           if x != y and x != z and x != m and y != z and y != m and z != m]
    prime = [int(i) for i in all if isPrime(int(i))]
    print(prime)
    return
# [1423, 2143, 2341, 4231]
```

9. 编写程序，实现分段函数计算，如表3-1所示。

表 3-1 分段函数计算

x	y
$x<0$	0
$0\leqslant x<5$	x
$5\leqslant x<10$	$3x-5$
$10\leqslant x<20$	$0.5x-2$
$20\leqslant x$	0

```
"""9. 编写程序，实现分段函数计算，如表3-1所示。"""


def piecewiseFuction(x):
    if x < 0:
        print(0)
        return 0
    elif 0 <= x < 5:
        print(x)
        return x
    elif 5 <= x < 10:
        print(3 * x - 5)
        return 3 * x - 5
    elif 10 <= x < 20:
        print(0.5 * x - 2)
        return 0.5 * x - 2
    elif 20 <= x:
        print(0)
        return 0

x = input('x:')
piecewiseFuction(int(x))
```

第四章 习题

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。

```
"""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。"""
import re


def correct1(s):
    print(re.sub(r'\bi\b', 'I', s))  # 利用正则替换


s = input("s: ")
correct1(s)
# s: i am a girl. i. i,
# I am a girl. I. I,
```

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。

```
"""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。"""
import re


def correct1(s):
    print(re.sub(r'\bI\b', 'i', s))  # 利用正则替换


s = input("s: ")
correct1(s)
# s: I am a girl. I. I,
# i am a girl. i. i,
```

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.”

```
"""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。"""
import re


def moveRepeat(s):
    for v in {x: s.count(x) for x in re.findall(r'\b\w+\b', s)}.keys():
        print(v, end=' ')


s = input('s:')
moveRepeat(s)
```

4. 简单解释Python的字符串驻留机制。

- **定义**：在计算机科学中，字符串驻留一种仅保存一份相同且不可变字符串的方法。不同的值被存放在字符串驻留池中。
- **限制**：仅包含下划线（_）、字母和数字的字符串会启用字符串驻留机制驻留。因为解释器仅对看起来像python标识符的字符串使用intern()方法，而python标识符正是由下划线、字母和数字组成。python只会针对整数范围为[-5, 256]的整数启用字符串驻留
- **字符串驻留机制的优缺点**如下：
  - **优点**：能够提高一些字符串处理任务在时间和空间上的性能，
  - **缺点**：在创建或驻留字符串时的会花费更多的时间。

举例：string1 = “aabbcc” string2 = “aabbcc” 使用id（string1）和id（string2）得到的内存地址是一样的。

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。

```
"""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


def moveRepeat(s):
    """利用正则和字典生成式，生成一个字典。key是单词，value是单词的长度。"""
    for k, v in {x: len(x) for x in re.findall(r'\b\w+\b', s)}.items():
        print(k, end=' ') if v == 3 else print('', end='')


s = input('s:')
moveRepeat(s)
```

第五章 习题

1. 运行5.3.1节最后的示例代码，查看结果并分析原因。

- Python函数在定义的时候，默认参数old_list的值就被计算出来了，即[]，因为默认参数L也是一个变量，它指向对象[]，每次调用该函数，如果改变了L的内容，则下次调用时，默认参数的内容就变了，不再是函数定义时的[]了。
- 而当设置成None的时候，None是一个不变对象。不变对象一旦创建，对象内部的数据就不能修改，这样就减少了由于修改数据导致的错误。

```
'''多次调用函数并且不为默认参数传递值时，默认参数只在第一次调用时进行解释。'''


def demo(newitem, old_list=[]):  # old_list指向的是一个空列表，在第一次初始化的时候，就已经指定了一个地址单元。之后不再被修改。
    old_list.append(newitem)
    return old_list


def demo1(newitem, old_list=None):  # old_list置空，在函数体内再分配内存单元
    if old_list is None:
        old_list = []
    old_list.append(newitem)
```

```
    return old_list


if __name__ == '__main__':
    # test demo
    print(demo('5', [1, 2, 3, 4]))
    print(demo('aaa', ['a', 'b']))
    print(demo('a'))
    print(demo('b'))    # 故在第一次调用demo('a')之后，再次调用demo('b')会将原来
的['a']作为默认的list.

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

    # test demo1
    print(demo1('5', [1, 2, 3, 4]))
    print(demo1('aaa', ['a', 'b']))
    print(demo1('a'))
    print(demo1('b'))

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

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。

```
def isPrime():
    """ 判断一个数是否为素数
        本例主要演示循环结构中else子句的用法"""
    import math

    n = input('Input an integer:')
    n = int(n)
    m = math.ceil(math.sqrt(n) + 1)  # 取n开根号的值，+1是因为range是闭开区间
    for i in range(2, m):
        if n % i == 0 and i < n:
            print('No')
            break
    else:
        print('Yes')


if __name__ == '__main__':
    isPrime()
```

3. 编写函数，接受一个字符串，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 countWord(s):
    c = [0 for i in range(4)]    # 初始化存储四个需要统计字符类型的数组
    for w in s:                  # 通过 if 语句判断，在范围内 +1
        if 'A' <= w <= 'Z':
            c[0] += 1
        elif 'a' <= w <= 'z':
            c[1] += 1
        elif '0' <= w <= '9':
            c[2] += 1
        else:
            c[3] += 1
    return tuple(c)              # 返回元组类型


if __name__ == '__main__':
    s = '12345ab67abcdefgAHIJKLM,.,.,.'
    print('大写字母个数: ', countWord(s)[0])
    print('小写字母个数: ', countWord(s)[1])
    print('数字: ', countWord(s)[2])
    print('其他字符个数: ', countWord(s)[3])
```

4. 在函数内部可以通过关键字_____来定义全局变量

global

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.

6. 调用带有默认参数的函数时，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。（判断对错）

错误。在调用有默认值的函数时，如果不给默认值参数传递任何值，则使用设定的默认值；否则，使用传入的值作为默认参数的值。

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

```
global a
a = 100


def testGlobal():
    a = 3
    print(a)
    return
```

```
if __name__ == '__main__':
    testGlobal()  # 全局变量被局部变量覆盖
    print(a)      # 局部变量a被销毁，全局变量a显示值

    '''
    输出:
        3
        100
    '''
```

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（判断对错）。

错。

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 SumAndMax(*p):
    print(max(p))
    print(sum(p))


if __name__ == "__main__":
    SumAndMax(0, 9, 2, 4, 523, 23)
```

10. 编写函数，模拟内置函数sum()。

```
'''模拟sum:
描述
sum() 方法对序列进行求和计算。

语法
以下是 sum() 方法的语法:

sum(iterable[, start])
参数
iterable -- 可迭代对象，如：列表、元组、集合。
start -- 指定相加的参数，如果没有设置这个值，默认为0。'''


def my_Sum(p, a=0):
    try:
        from collections.abc import Iterator
        s = 0
        if a != 0 and isinstance(a,int):
            s += a
        elif not isinstance(a,int):
            print('add data type error.')
            return
        if isinstance(iter(p), Iterator):  # 判断传入的参数是一个迭代器类型的参数
            for i in p:
```

```
                if isinstance(i, int):      # 判断传入的迭代器中的每个成员都是int
类型
                    s += i
                else:
                    print('list member type error.')    # 对于非int类型成员，进
行异常处理
                    return
            print(s)
        return
    except TypeError:
        print('TypeError:  object is not iterable.') # 对于非迭代器成员进行异常处
理


if __name__ == '__main__':
    l = [1, 2, 3, 4]
    li = (1, 2, 3, 4)
    # 对列表求和测试
    my_Sum(l)
    my_Sum(l, -5)
    # 对元组求和测试
    my_Sum(li)
    # 对range迭代器求和测试
    my_Sum(range(0, 6))
    # 对类型异常进行测试
    my_Sum(0)
    my_Sum([1,2,'a'])
    my_Sum([1,2,3],'a')
```

11.包含_____语句的函数可以用来创建生成器。

field

12. 编写函数，模拟内置函数sorted。

```
""" 12. 编写函数，模拟内置函数sorted。"""


def my_sorted(iterable):
    from collections.abc import Iterable
    if not isinstance(iterable, Iterable):
        print('TypeError: only receive iterable')
        return
    u = []
    temp = [x for x in iterable]  # 复制原列表
    while temp:
        Min = min(temp)
        u.append(Min)
        temp.remove(Min)
    return u


if __name__ == '__main__':
```

```
    # l = [2, 3, 1, 4]
    l = (2, 3, 1, 4)
    # [1, 2, 3, 4]
    print(my_sorted(l))
    l1 = ['a', 'd', 'b', 'z', 'q']
    print(my_sorted(l1))
    # ['a', 'b', 'd', 'q', 'z']
```

第六章 习题

1.继承6.5节例6-2中的Person类生成Student类，编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。

```
''' 6.2 在派生类中调用基类方法
    首先设计Person类，然后以Person为基类派生Teacher类，分别创建Person类和Teacher类的对象，
    并在派生类对象中调用基类方法。'''


class Person:
    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.')
            return
        self.__name = name

    def setAge(self, age):
        if not isinstance(age, int):
            print('name must be 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 Teacher(Person):  # 派生类
    def __init__(self, name="", age=30, sex='man', department='Computer'):
        super(Teacher, self).__init__(name, age, sex)
        # #or, use another method like below:
```

```
        # Person.__init__(self, name, age, sex)
        self.setDepartment(department)

    def setDepartment(self, department):
        if not isinstance(department, str):
            print('department must be a string.')
            return
        self.__department = department

    def show(self):
        # Person.show()
        super(Teacher, self).show()
        print('Department:', self.__department)


class Student(Person):
    def __init__(self, name="", age=18, sex='man', major='Computer Science'):
        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.')
            return
        self.__major = major

    def show(self):
        Person.show(self)
        print('Major', self.__major)


if __name__ == '__main__':
    w = Student()
    w.show()
    w = Student('wendy', 20, 'woman', 'Data Analyse')
    w.show()
''' 输出结果
Name
Age 18
Sex man
Major Computer Science
Name wendy
Age 20
Sex woman
Major Data Analyse
'''
```

2. 设计一个三位向量类，并实现向量的加法、减法以及向量和标量的乘法和除法运算。

```
''' 2. 设计一个三位向量类，并实现向量的加法、减法以及向量和标量的乘法和除法运算。'''


class Vector(object):
    __vector = []   # 创建类的私有成员
```

```
    def __init__(self, *args):  # 初始化三位向量
        if not args:    # 判断参数是否为空
            self.__vector = []
        elif len(args) > 3: # 向量成员超过3个，不能创建
            print('the member in the victor is bigger than three.')
            return
        else:
            for i in args:  # 判断向量成员，数据类型
                if not self.__IsNumber(i):
                    print('member is not a number')
                    return
                self.__vector = list(args)

    def __IsNumber(self, n):    # 辅助函数，判断数据n是否是一个整型或者浮点数
        if not isinstance(n, int) and not isinstance(n, float):
            return False
        return True

    @property   # 设置get属性
    def vector(self):
        return self.__vector

    @vector.setter  # 设置set属性
    def vector(self, v):
        if not isinstance(v, Vector):
            print('Only receive Vector.')
            return
        self.__vector = v.__vector

    def show(self): # 显示向量数据
        print(self.__vector)

    def __add__(self,v):
        if not isinstance(v,Vector):    # v如果不是Vector类型不提供计算
            print('v is not Vector.')
            return
        elif isinstance(v,Vector):  # v是Vector类型，进行加法计算
            a = Vector()
            for x,y in zip(self.__vector,v.__vector):
                a.__vector.append(x + y)
            return a.__vector
        else:
            print('Not supported')

    def __sub__(self,v):
        if not isinstance(v,Vector):    # v如果不是Vector类型不提供计算
            print('v is not Vector.')
            return
        elif isinstance(v,Vector):  # v是Vector类型，进行减法计算
            a = Vector()
            for x,y in zip(self.__vector,v.__vector):
                a.__vector.append(x - y)
            return a.__vector
        else:
            print('Not supported')
```

```
    def __mul__(self,v):
        if isinstance(v,int):   # 标量乘法
            a = Vector()
            for x in self.__vector:
                a.__vector.append(x * v)
            return a.__vector
        elif isinstance(v,Vector):  # 向量乘法
            a = Vector()
            for x,y in zip(self.__vector,v.__vector):
                a.__vector.append(x * y)
            return sum(a.__vector)
        else:
            print('Not supported')

    def __truediv__(self, v):
            if isinstance(v, int):  # 标量除法
                a = Vector()
                for x in self.__vector:
                    a.__vector.append(x / v)
                return a.__vector
            elif isinstance(v, Vector):  # 向量乘法
                a = Vector()
                for x, y in zip(self.__vector, v.__vector):
                    a.__vector.append(x / y)
                return a.__vector
            else:
                print('Not supported')

if __name__ == '__main__':
    v1 = Vector(1, 2, 3)
    v1.show()
    # [1, 2, 3]
    v2 = Vector(4,5,6)
    v2.show()
    # [4, 5, 6]
    # 向量加
    print(v1+v2)
    # [5, 7, 9]
    # 向量减
    print(v1-v2)
    # [-3, -3, -3]
    # 向量乘
    print(v1 * v2)
    # 32
    # 向量标量乘法
    print(v1 * 3)
    # [3, 6, 9]
    # 向量除法
    print(v1 / v2)
    # [0.25, 0.4, 0.5]
    # 向量标量除法
    print(v1 / 2)
    # [0.5, 1.0, 1.5]
```

3. 面向对象程序设计的三要素分别为_____，_____和_____。

封装，继承和多态

4. 简单解释Python中以下划线开头的变量名的特点。

- _xxx: 这样的对象叫做保护成员，不能用"from module import *"导入，只有类对象和子类对象能访问这些成员。
- __xxx__: 系统定义的特殊成员。
- __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__私有成员名”这样的特殊方式类访问。

5. 与运算符“”对于的特殊方法名为_____,与运算符“//”对于的特殊方法名为_____。**

__pow__; __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”的值设置为3的语句可以写作____。

a._A__value = 3

测试代码：

```
>> class A:
    __value = 5
    def show(self):
        print(self.__value)


>>> a = A()
>>> a.show()
5
>>> a._A__value = 3
>>> a.show()
3
```