Python实验报告4

班级：　　姓名：　　学号：

要求：

1. 在文件开头部分填写自己的信息；
2. 在每个题下方的代码块中书写该题的代码，并运行出结果；
3. 在2节课的时间内完成前8个题；打印为pdf文件并提交，文件名改为“Python实验4+班级姓名.pdf”。

一、实验目的：

- 掌握Python函数定义和调用方法。
- 掌握Python匿名函数的定义方法。
- 熟悉和应用Python函数的多种参数类型。

```
1. 下列关于Python中函数的说法，哪些是正确的？（ B,C,E,F,G,H ）
A. 函数的输入参数必须指定类型；
B. 函数的返回不须指定类型；
C. 函数定义中，没有return语句，或return语句没有执行，则返回空值；
D. 函数如果没有输入参数，则函数名后面的圆括号()不需要；
E. 在同一文件中，函数的定义必须放在函数的调用之前；
F. 函数在调用时，位置参数、关键字参数、默认值参数可以混合使用；
G. 函数在调用时，位置参数必须放在关键字参数前面；
H. 函数在调用时，可以使用序列或字典作为实参。
```

```
2. 给定下列的字符串，
     as, bit, break, case, continue, def, define, do, elif, end, for, function, get, if, in, map, max, mode, not, range，return, sqrt, start, sum, switch, while, yes, zip.
(1) 哪些是Python中的关键字（  ）
(2) 哪些是Python的内置函数（  ）
(3) 哪些可以做Python的变量名（  ）
```

```
上述的字符串中，哪些是Python的关键字、内置函数，哪些可以作为变量名，只需将它们放在jupyter notebook的一个代码单元中，或放在某个集成编辑器中，即可根据其字体看出。如，下方可以看到，绿色粗体表示关键字，绿色正常字体表示内置函数，而黑色字体是可以作为变量名的。
```

In [ ]:

```
as, bit, break, case, continue, def, define, do, elif, end, for,
function, get, if, in, map, max, mode, not, range，return, sqrt,
start, sum, switch, while, yes, zip
```

3. 用map函数和匿名函数求级数和 $\sum_{n=1}^{100} \frac{n}{6^n}$.

In [1]:
```
s1 = sum(map(lambda n:n/6**n, range(1, 101)))
print(s1)
```

0.24

4. 编写一个函数profac(x)，对输入的正整数x，输出其所有真因数的序列。如 profac(18)将返回[1,2,3,6,9]。

In [1]:
```
def profac(x):
    out = []
    for k in range(1, x // 2 + 1):
        if x % k == 0:
            out.append(k)
    return out

profac(18)
```

Out[1]: [1, 2, 3, 6, 9]

5. 编写一个函数dot，用于求向量x,y的内积，即 $\sum_{i=1}^{n} x_i y_i$的值。这里x,y是长度为n的数值列表或元组。并对x=[1,2,3,4]，y=[3,2,1,0]调用该函数。

In [28]:
```
def dot(x, y):
    if len(x) != len(y):
        print('x,y的长度要一致!')
    else:
        out = 0
        for i in range(len(x)):
            out += x[i]*y[i]
        return out

x = [1, 2, 3, 4]
y = [3, 2, 1, 0]
dot(x, y)
```

Out[28]: 10

In [1]:
```
def dot(x, y):
    if len(x) != len(y):
        print('x,y的长度要一致!')
    else:
        return sum([i*j for i,j in zip(x, y)])

x = [1, 2, 3, 4]
y = [3, 2, 1, 0]
dot(x, y)
```

Out[1]: 10

In [7]:
```
# 如果能确定x, y长度相同，则可简化为
dot = lambda x,y: sum(i*j for i,j in zip(x,y))

x = [1, 2, 3, 4]
y = [3, 2, 1, 0, 3]
dot(x, y)
```

Out[7]: 10

6. 编写一个函数 gcd(x,y)，对输入的2个正整数，求其最大公约数。并分别用 gcd(48,84) 和 gcd(105,28) 检验该函数。

In [4]:
```
def gcd(x, y):
    """穷举法求最大公约数"""
    for k in range(min(x,y),0,-1):  # 从大到小试探
        if x%k==0 and y%k==0:
            return k

print(gcd(48, 84))
print(gcd(105, 28))
```

```
12
7
```

In [9]:
```
def gcd(x, y):
    """运用辗转相除法求最大公约数"""
    while y:  # 相当于 while y!=0
        x, y = y, x % y  # 若x < y则自动交换x, y
    return x  # 返回最大公约数

gcd(48, 84), gcd(105, 28)
```

Out[9]: (12, 7)

In [10]:
```
def gcd(x, y):
    """递归法实现辗转相除求最大公约数"""
    if y:
        return gcd(y, x % y)
    else:
        return x

gcd(48, 84), gcd(105, 28)
```

Out[10]: (12, 7)

7. 矩形法是数值方法求定积分的最基本方法。即 $\int_a^b f(x)dx \approx \sum_{k=0}^{n-1} f(x_k)\cdot\frac{b-a}{n}$，这里 n 为区间$[a,b]$ 的等分数，x_k，$(k=0,1,\cdots,n-1)$ 是 $[a,b]$ 上的等分点。试定义一个用矩形法求$\int_a^b f(x)dx$的近似值的函数 quad(f, a, b, n)，f代表被积函数 $f(x)$，a,b是积分下限和上限，默认取0和1，n默认取1000。 并用 $\int_0^{\pi} x\sin(x)dx$ 检验，其准确值为 π.

In [12]:
```
def quad(f, a=0, b=1, n=1000):
    '''矩形法求函数f在[a,b]上的数值积分，[a,b]上子区间划分为n个'''
    d,s = (b-a)/n,0
    for k in range(n):
        s += f(a)
        a += d   # 这里a从左至右依次取点，每次自增d
    return s*d

import math
f = lambda x: x*math.sin(x)
s = quad(f, b=math.pi)
s
```

Out[12]: 3.141590069733054

In [14]:
```
def quad(f, a=0, b=1, n=1000):
    '''矩形法求函数f在[a,b]上的数值积分，[a,b]上子区间划分为n个'''
    d = (b-a)/n
    x = [a+d*k for k in range(n)]  # 产生序列 xk
    return sum(map(f, x))*d

from math import sin, pi
f = lambda x: x*sin(x)
s = quad(f, 0, pi, 10000)
s
```

Out[14]: 3.141592627751224

```
本题也可以在numpy中运算，numpy的使用见下次课。
```

8. 编写一个排序函数 mysort(x)，用于对数组x的元素从小到大排序，返回排序后的数组。x是数值列表或元组。不得使用现成的排序函数。以 x=[9, 0, 3, 2, 1, 6, -7, 4, 8, 5] 检验你的排序函数。

In [2]:
```
def mySort(x):
    '''冒泡排序法对数组x从小到大排序'''
    x = list(x)  # 复制列表，或将元组转换为列表
    for i in range(1,len(x)):
        for j in range(len(x)-i):
            if x[j]>x[j+1]:
                x[j],x[j+1] = x[j+1],x[j]
    return x

x=[9, 0, 3, 2, 1, 6, -7, 4, 8, 5]
mySort(x)
```

Out[2]: [-7, 0, 1, 2, 3, 4, 5, 6, 8, 9]

In [3]:
```python
def mySort(x):
    '''插入排序法对数组x从小到大排序'''
    x = list(x)  # 复制列表，或将元组转换为列表
    for i in range(len(x)):
        pre = i-1  # 已排序的索引号
        cur = x[i]  # 当前元素
        while pre >= 0 and x[pre] > cur: # 当前元素与已排序的从后向前逐一比较
            x[pre+1] = x[pre]  # 依次后移
            pre-=1
        x[pre+1] = cur
    return x

x=9, 0, 3, 2, 1, 6, -7, 4, 8, 5  # 输入为元组
mySort(x)
```

Out[3]: [-7, 0, 1, 2, 3, 4, 5, 6, 8, 9]

9. （选做题）求一元二次方程的根。编写一个函数 sqroot(a,b,c)，对输入的一元二次方程 $ax^2+bx+c=0$的各项系数(降幂排列)，按不同情形输出方程的根。无实根的情况下，要能求出虚根（注：math.sqrt()不能求负数的平方根）。并分别验证这3种情况。

本问题的关键是如何正确地表示虚根，返回的虚根应是能正常运算的复数。可以使用以下几种方法产生复数

In [49]:
```python
a,b = 3,4
a + b*1j
```

Out[49]: (3+4j)

In [19]:
```python
a,b = 3,4
complex(a, b)
```

Out[19]: (3+4j)

In [15]:
```python
# cmath模块的sqrt能对负数开平方
from cmath import sqrt

print(sqrt(-4))
sqrt(4)  # 但cmath.sqrt对正数求平方根也会输出虚数形式
```

```
2j
```

Out[15]: (2+0j)

In [13]:
```python
# 以下方法也可以求负数的平方根，但误差稍大
(-4)**0.5  # pow(-4,0.5)
```

Out[13]: (1.2246467991473532e-16+2j)

以下用第1种方法表示虚根。其他方法请自行尝试。

In [16]:

```python
# 定义一个函数，求一元二次方程的根。将3个系数作为3个输入参数。
def sqroot(a,b,c):
    '''求一元二次方程 ax^2 + bx + c = 0 的根'''
    from math import sqrt
    d = b*b - 4*a*c
    sqrt_d = sqrt(d) if d>=0 else sqrt(-d)*1j
    x1 = (-b + sqrt_d)/2/a
    x2 = (-b - sqrt_d)/2/a
    return x1, x2

print(sqroot(1,-2,1))
print(sqroot(1,2,2))
print(sqroot(2,3,-5))
```

```
(1.0, 1.0)
((-1+1j), (-1-1j))
(1.0, -2.5)
```

10. （选做题）对上面的排序函数，增加一个输入参数order，用于表示升序'ascend'或降序'descend'，默认为'ascend'。并增加一个输出项，用于表示排序后的数在原数组中的索引号。如列表 [7,2,9,0,5,11,16,13] 降序排序后的输出为[16, 13, 11, 9, 7, 5, 2, 0]和[6, 7, 5, 2, 0, 4, 1, 3].

In [20]:

```python
def mySort(x, order='ascend'):
    '''冒泡排序法对数组x从小到大排序'''
    x = list(x)  # 复制列表，或将元组转换为列表
    ind = list(range(len(x)))  # 保存数组元素原序号
    for i in range(1,len(x)):
        for j in range(len(x)-i):
            if x[j]>x[j+1]:
                x[j],x[j+1] = x[j+1],x[j]
                ind[j],ind[j+1] = ind[j+1],ind[j]
    if order=='descend':  # 降序，是升序的逆序
        x=x[::-1]  # x.reverse()
        ind=ind[::-1]  # ind.reverse()
    elif order!='ascend': # 如果order不是ascend或descend，则返回异常
        raise ValueError('"order" should be "ascend" or "descend"!')
    return x, ind

x = [7,2,9,0,5,11,16,13]
print(mySort(x, 'ascend'))
mySort(x, 'descend')
```

```
([0, 2, 5, 7, 9, 11, 13, 16], [3, 1, 4, 0, 2, 5, 7, 6])
```

Out[20]:

```
([16, 13, 11, 9, 7, 5, 2, 0], [6, 7, 5, 2, 0, 4, 1, 3])
```

In [21]:

```
def mySort(x, order='ascend'):
    '''插入排序法对x1从小到大排序'''
    x = list(x)  # 保留原数组
    ind = list(range(len(x)))  # 保存数组元素原序号
    for i in range(len(x)):
        pre = i-1  # 已排序的索引号
        cur = x[i]  # 当前元素
        while pre >= 0 and x[pre] > cur: # 当前元素与已排序的从后向前逐一比较
            x[pre+1] = x[pre]  # 元素向后移
            ind[pre+1] = ind[pre]
            pre-=1
        x[pre+1] = cur  # 当前元素插入适当位置
        ind[pre+1] = i
    if order=='descend':  # 降序，是升序的逆序
        x=x[::-1]  # x.reverse()
        ind=ind[::-1]  # ind.reverse()
    return x, ind

x = [7,2,9,0,5,11,16,13]
print(mySort(x, 'ascend'))
mySort(x, 'descend')
```

```
([0, 2, 5, 7, 9, 11, 13, 16], [3, 1, 4, 0, 2, 5, 7, 6])
```

Out[21]:
```
([16, 13, 11, 9, 7, 5, 2, 0], [6, 7, 5, 2, 0, 4, 1, 3])
```

11. （选做题）设计一个求最大值的函数 mymax，适用于3种情形，mymax(x) 用于求x中元素的最大值；mymax(x,n) 求x中元素与数n相比的最大值，返回一个与x同样大小的对象；mymax(x,y) 求x与y中对应元素的最大值，返回一个与x同样大小的对象。这里x,y是同样大小的数值列表或元组。不使用现成的求最大值的函数。

In [50]:

```
def mymax(*p):
    '''mymax(x)用于求x中元素的最大值;
    mymax(x,n)求x中元素与数n相比的最大值;
    mymax(x,y)求x与y中对应元素的最大值'''
    if len(p)==1:
        x = p[0]
        xmax = x[0]
        for t in x[1:]:
            if t>xmax:
                xmax=t
        return xmax
    elif len(p)==2:
        x, y = p
        # 以下将 mymax(x,n)转换成与 mymax(x,y)一样的任务
        if type(y) in {int, float, range}:
            y = [y for i in range(x)]  # 产生与x等长的y序列
        if type(y) in {list, tuple} and len(y)==len(x):
            xmax = [m+(n-m)*(n>m) for n,m in zip(x,y)]
            return xmax
        else:
            raise ValueError('y的长度应为1或与x相同')
    else:
        raise SyntaxError('输入参数为1个或2个')

print(mymax([7,2,9,0,5]))
print(mymax([7,2,9,0,5], 6))
print(mymax([7,2,9,0,5], [4,5,2,3,0]))
```

```
9
[7, 6, 9, 6, 6]
[7, 5, 9, 3, 5]
```

In [25]:

```python
# 以下代码来自4班吴湘帆同学
def mymax(x, y=None):
    if y is None:
        maximum = x[0]
        for element in x:
            if element > maximum:
                maximum = element
        return maximum
    elif isinstance(y, (list, tuple)):
        result = []
        for i in range(len(x)):
            if x[i] > y[i]:
                result.append(x[i])
            else:
                result.append(y[i])
        return type(x)(result)
    else:
        result = []
        for element in x:
            if element > y:
                result.append(element)
            else:
                result.append(y)
        return type(x)(result)

print(mymax([1, 2, 3, 4]))
print(mymax([1, 2, 3, 4], 5))
print(mymax([1, 2, 3, 4], [4, 3, 2, 1]))
```

```
4
[5, 5, 5, 5]
[4, 3, 3, 4]
```

In [ ]:

```python

```