

一、

- (1) 扩展库管理工具 pip 的使用
- (2) 标准库与扩展库对象的导入与使用

```
import random
n = random.random()
n = random.randint(1,100)

import math                      # from math import *
math.sin(0.5)

from math import sin
sin(3)
```

- (3) 通过案例认识 Python 程序和编程规范

```
for i in range(1, 10):
    for j in range(1, i+1):
        print('{0}*{1}={2}'.format(i,j,i*j), end=' ')
    print()
```

二、

1. 运算符:

- (1) ‘+’ 运算符除了用于算术加法以外，还可以用于列表、元组、字符串的连接，如：“12” + “34” 结果是：“1234”
- (2) ‘*’ 除了表示算术乘法，还可用于列表、元组、字符串与整数的乘法，表示序列元素的重复，如：’abc’ * 3 结果是 ‘ abcabcabc ’
- (3) ‘ / ’ 和 ‘ // ’ 分别表示算术除法和算术求整商。
- (4) ‘ ** ’ 表示幂乘，如：3**2 结果是 9
- (5) 关系运算符可以连用，如：1 < 3 < 5 结果是 True
- (6) ‘ in ’ 用于成员测试，即测试一个对象是否为另一个对象的元素，如：’abc’ in ’abcdefg’ 的结果是 True；’ac’ in ’abcdefg’ 的结果是 False
- (7) ‘ | ’ 用于集合并运算，如：{1, 2} | {2, 3} 的结果是{ 1, 2, 3 }
- ‘ & ’ 用于集合的交运算，如：{1, 2, 3} & {3, 4, 5} 的结果是{ 3 }
- ‘ - ’ 用于集合的差运算，如：{1, 2, 3} - {3, 4, 5} 结果是{1, 2, 4, 5}

2. 内置函数：不需要额外导入任何模块即可直接使用，如：sum()、max()、min()、sorted()等。

(1) 排序函数 sorted()对列表、元组、字典、集合或其他可迭代对象进行排序并返回新列表。

(2) reversed()对可迭代对象进行翻转并返回可迭代的 **reversed** 对象。

(3) **map()**把一个函数 func 依次映射到序列或迭代器对象的每个元素上，并返回一个可迭代的 map 对象。

```
>>> list(map(str, range(5)))  
['0', '1', '2', '3', '4']
```

(4) **filter()**将一个单参数函数作用到一个序列上，返回该序列中使得该函数返回值为 True 的那些元素组成的 filter 对象。

```
>>> x = list(range(10))  
>>> list(filter(lambda value : value % 2 == 0, x))  
[0, 2, 4, 6, 8]
```

(5) range()返回指定范围的整数序列，包含左闭右开区间[start,end)内以 step 为步长的整数，语法格式为 range([start,] end [, step])，参数 start 默认为 0，step 默认为 1。

```
>>>list(range(9, 0, -2))  
[9, 7, 5, 3, 1]
```

(6) zip()把多个可迭代对象中的元素压缩到一起，返回一个可迭代的 zip 对象，其中每个元素都是包含原来的多个可迭代对象对应位置上元素的**元组**。

```
>>> list(zip('abcd', [1, 2, 3]))  
[('a', 1), ('b', 2), ('c', 3)]
```

(7) 输入：input()，**接收的任何输入都作为字符串对象**，可以使用内置函数 int()、float()或 eval()对用户输入的内容进行类型转换。

```
>>> x = input('Please input: ')
```

Please input: 345 (输入 345)

```
>>> x
```

```
'345'
```

```
>>> int(x)                      #转换为整数
```

```
345
```

3. 列表：若干元素的有序连续内存空间，用于存储任意数目、任意类型的数据集合。

(1) 列表的所有元素放在一对方括号[]中，相邻元素之间使用逗号分隔。如：

```
a=[10,'20','abc',True]
```

(2) 列表的创建：

```
>>> a_list = ['a', 'b', 'mpilgrim', 'z', 'example']      # 直接赋值，使用[]
```

```
>>> list(range(1, 10, 2))                      # 使用 list() 函数
```

```
>>> list((3,5,7,9,11))
```

```
>>> list({'a':3, 'b':9, 'c':78}.items())
```

(3) 列表的删除：

```
>>> del x                      # 删除列
```

'P'	'y'	't'	'h'	'o'	'n'	
0	1	2	3	4	5	6
-6	-5	-4	-3	-2	-1	

表对象 x

(4) 列表元素的访问：

```
>>> x[0]                      # 'p'
```

```
>>> x[-1]                      # 'n'
```

(5) 列表常用方法：

① 添加元素：append()、insert()、extend() (原地操作)

② 删减元素：pop()、remove()、clear() (原地操作)

③ 统计指定元素：count()

④ 定位指定元素：index() (返回指定元素在列表中首次出现的位置)

⑤ 元素排序：sort()、reverse()

(6) 列表对象支持的运算符

- ① ‘+’ “拼接”两个列表，返回**新列表**对象。
- ② ‘+=’ “拼接”两个列表，原地操作，操作效率等同 `append()`
- ③ ‘*’ 用于列表和整数相乘，表示序列重复，返回新列表。
- ④ ‘*=’ 用于列表和整数相乘，表示序列重复，原地操作。

(7) 内置**函数**对列表的操作

`max()`、`min()`、`sum()`、`len()`

`zip()`: 把函数映射到列表上的每个元素

```
>>> list(zip('abcd', '1234'))
```

`filter()`: 根据指定函数的返回值对列表元素进行过滤:

```
>>> x = list(range(10))
```

```
>>> list(filter(lambda v : v % 2 != 0, x))
```

`map()`: 把函数映射到列表上的每个元素:

```
>>> k = 12345
```

```
>>> list(map(int, str(k)))
```

```
>>> list(map(list, zip(*[[1, 2, 3], [4, 5, 6]])))
```

`all()`: 测试列表中是否所有元素都等价于 `True`;

`any()`: 测试列表中是否有等价于 `True` 的元素。

```
>>> all([0,1,2,3,4])
```

```
>>> any([0,1,2,3,4])
```

(8) **列表推导式**: 使用非常简洁的方式来快速生成满足特定需求的列表。

```
>>> aList = [x*x for x in range(10)]
```

```
>>> vec = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
>>> [num for elem in vec for num in elem]
```

```
>>> vec = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
>>> [num for elem in vec for num in elem]
```

```
>>> [v**2 if v%2 == 0 else v+1 for v in [2, 3, 4, -1] if v>0]
```

```
>>>[v*2 if v> 0 else v**2 for v in [17,-2,9,-4,32,0] if v%2==0]
```

```
>>> [i for i, v in enumerate([3,5,7]) if v == max([3,5,7])]
```

(9) 切片操作: [start : end : step]

```
>>> aList = [3, 4, 5, 6, 7, 9, 11, 13, 15, 17]
```

```
>>> aList[::-1]
```

```
>>> aList[::2]
```

```
>>> aList[1::2]
```

```
>>> aList[3:6]
```

```
>>> aList[-15:3]
```

```
>>> aList[:0] = [1, 2]
```

```
>>> aList[0::5] = map(lambda x: x!=5, range(3))
```

4. **元组**：轻量级列表。元组属于**不可变**序列，不可以直接修改元组中元素的值，也无法为元组增加或删除元素。

(1) 元组的创建：

```
>>> x = tuple()
>>> tuple(range(5))
>>> list(zip(range(3), 'abcdefg'))
[(0, 'a'), (1, 'b'), (2, 'c')]
```

(2) 生成器推导式，返回一个生成器对象，只能访问一次

```
>>> g = ((i+2)**2 for i in range(10))
>>> tuple(g)

>>> g = ((i+2)**2 for i in range(10))
>>> for item in g:
    print(item, end=' ')
```

5. **字典**：反映对应关系的映射类型。元素放在一对大括号“{ }”中，每个元素包含用冒号分隔开的“**键**”和“**值**”两部分，“键”必须为不可变数据类型且不允许重复

```
>>> aDict = {'server': 'db.diveintopython3.org', 'database': 'mysql'}
```

(1) 字典创建与删除

```
>>> keys = ['a', 'b', 'c', 'd']  
  
>>> values = [1, 2, 3, 4]  
  
>>> dictionary = dict(zip(keys, values))  
  
>>> d = dict(name='Dong', age=39)
```

(2) 字典元素的访问：“键”作为下标就可以访问对应的“值”。

```
>>> aDict = {'age': 39, 'score': [98, 97], 'name': 'Dong', 'sex': 'male'}  
>>> aDict['age']  
>>> aDict.get('age')  
  
>>> aDict['address']      #指定的“键”不存在，抛出异常  
  
>>> aDict.get('address', 'Not Exists.') #指定的“键”不存在时返回指定的默认值
```

(3) 字典应用实例：生成包含 1000 个随机字符的字符串，然后统计每个字符的出现次数。

```
>>> import string  
>>> import random  
>>> x = string.ascii_letters + string.digits + string.punctuation  
>>> y = [random.choice(x) for i in range(1000)]  
>>> z = ''.join(y)  
>>> d = dict()      #使用字典保存每个字符出现次数  
  
>>> for ch in z:  
    d[ch] = d.get(ch, 0) + 1
```

(4) 序列解包：对多个变量同时进行赋值

```
>>> x, y, z = sorted([1, 3, 2])  
>>> s = {'a':1, 'b':2, 'c':3}  
>>> b, c, d = s.items()  
  
>>> x = ['a', 'b', 'c']  
>>> for i, v in enumerate(x):  
    print('The value on position {0} is {1}'.format(i,v))
```

```
>>> keys = ['a', 'b', 'c', 'd']
>>> values = [1, 2, 3, 4]
>>> for k, v in zip(keys, values):
    print(k, v)
```

6. 集合：属于 Python 无序可变序列，使用一对大括号作为定界符，元素之间使用逗号分隔，同一个集合内的每个元素都是唯一的，元素之间不允许重复。

```
>>> s = set([0, 1, 2, 3, 0, 1, 3, 6])
>>> x = { 1, 2, 3, 5, 6 }
>>> s & x
>>> nums = set([1, 2, 2, 3, 3, 3, 4])
>>> print(len(nums))
```

三、

1. 字符串常用操作

```
>>> s="apple,peach,banana,peach,pear"
>>> s.count('p')
>>> s = "apple,peach,banana,pear"
>>> s.split(",")
>>> s = 'hello world \n\n My name is Dong '
>>> s.split()
>>> li = ["apple", "peach", "banana", "pear"]
>>> ','.join(li)
```

#测试用户输入中是否有敏感词，如果有的话就把敏感词替换为 3 个星号***

```
>>> words = ('测试', '非法', '暴力', '话')
>>> text = '这句话里含有非法内容'
>>> for word in words:
    if word in text:
        text = text.replace(word, '***')
```

```
>>> "aaaassddf".strip(" ")
```

```
>>> "aaaassdffaaa".rstrip("a")
```

```
>>> "aaaassdffaaa".lstrip("a")
```

```
>>> 'aabbccddeeffg'.strip('gaf')
```

2. 适用于字符串对象的内置函数、切片操作

```
>>> x = 'Hello world.'
```

```
>>> sorted(x)
```

```
>>> eval("3+4")
```

```
>>> 'Explicit is better than implicit.'[9:23]
```

```
>>> list(map(add, x, x))
```

3. 字符串常量

```
>>> import string
```

```
>>> string.digits
```

```
>>> string.ascii_letters
```

```
>>> string.ascii_lowercase
```

```
>>> string.ascii_uppercase
```

```
# 生成指定长度的随机密码
```

```
import string
```

```
characters = string.digits + string.ascii_letters
```

```
import random
```

```
".join([random.choice(characters) for i in range(8)])
```

```
# 生成字符映射表，加密字符串
```

```
text='abccd efgadh'
```

```
table=".maketrans('abcdefgh','helopytn')
```

```
print(text.translate(table))
```

四、函数与编程

1. lambda 表达式: 只包含一个表达式, 用来声明匿名函数, 适合需要一个函数作为另一个

函数参数的场合

```
>>> L = [1,2,3,4,5]
```

```
>>> print(list(map(lambda x: x+10, L)))
```

```
>>> sorted([123, 23, 223,12,33], key=lambda x: len(str(x)))
```


2. **函数编程案例 1:** 编写函数，接收任意多个实数，返回一个元组，其中第一个元素为所有参数的平均值，其他元素为所有参数中大于平均值的实数。

```
def demo(*para):
    avg = sum(para) / len(para)          # 平均值
    g = [i for i in para if i>avg]       # 列表推导式
    return (avg,) + tuple(g)

>>> a=(2,5,8,1,9)
>>> demo(*a)
```

3. **函数编程案例 2:** 编写函数，接收字符串参数，返回一个元组，其中第一个元素为大写字母个数，第二个元素为小写字母个数。

```
def demo(s):
    result = [0, 0]
    for ch in s:
        if ch.islower():
            result[1] += 1
        elif ch.isupper():
            result[0] += 1
    return tuple(result)
```

4. **函数编程案例 3:** 编写函数，实现生成指定大小范围内的、指定长度的、不重复的随机数集合，返回该集合。

```
import random
def randomNumbers(number, start, end):
    data = set()
    while len(data)<number:
        element = random.randint(start, end)
        data.add(element)
    return data
```

5. 函数编程案例 4:

一般来说，在一封正常的邮件中，标题文字中是不会出现太多类似于：**【**、**】**、*****、**/**、**?** 这样的符号，如果包含类似以上特殊字符的比例超过整个标题字数的 30%，该邮件被断定认为是垃圾邮件。

编写函数 `check (text)`，实现垃圾邮件的识别。如果是正常邮件，函数返回 `Ture`，

否则返回 False。

编写主程序实现：从键盘输入任意的邮件标题，调用函数 `check(text)` 判断是否为正常邮件，根据判断结果输出“正常邮件”或“垃圾邮件”

```
def check(text):
    characters = '【】*-/\\'
    num=sum(map(lambda ch:text.count(ch),characters))
    if num/len(text) > 0.3:
        return False
    return True
```

```
title = input()
if(check(title)):
    print('正常邮件')
else:
    print('垃圾邮件')
```

6. 函数编程案例 5：检查并判断密码字符串的安全强度

```
import string
def check(pwd):
    if not isinstance(pwd, str) or len(pwd)<6:
        return 'not suitable for password'
    d = {1:'weak', 2:'below middle', 3:'above middle', 4:'strong'}
    r = [False] * 4
    for ch in pwd:
        if not r[0] and ch in string.digits:      #是否包含数字
            r[0] = True
        elif not r[1] and ch in string.ascii_lowercase:  #是否包含小写字母
            r[1] = True
        elif not r[2] and ch in string.ascii_uppercase:  #是否包含大写字母
            r[2] = True
        elif not r[3] and ch in ',.!?<>':          #是否包含指定的标点符号
            r[3] = True
    return d.get(r.count(True), 'error')    #统计包含的字符种类，返回密码强度

print(check('a2Cd,'))
```

编程案例 7：判断任意两个字符串是否为**变位词**，如果是，则返回 True，否则返回 False。（如果一个字符串是 另一个字符串的重新排列组合，那么这两个字符串互为变位词。比如，” heart” 与” earth” 互为变位 词，” Mary” 与” arMy” 也互为变位词。）

```
def is_anagram(str1, str2):
    str1_list = list(str1)
    str2_list = list(str2)
    str1_list.sort()
    str2_list.sort()
    pos = 0
    Match=True
    while pos < len(str1) and match:
        if str1_list[pos] == str2_list[pos]:
            pos = pos + 1
        else:
            match=False
    return match
```

五、文本文件操作案例：假设文件 data.txt 中有若干整数，所有整数之间使用英文逗号分隔，编写程序读取所有整数，将其按升序排序后再写入文本文件 data_asc.txt

```
with open('data.txt', 'r') as fp:
    data = fp.readlines()
    data = [line.strip() for line in data]
    data = ','.join(data)
    data = data.split(',')
    data = [int(item) for item in data]
    data.sort()
    data = ','.join(map(str,data))
    with open('data_asc.txt', 'w') as fp:
        fp.write(data)
```

六、数据处理与可视化

1. pandas 模块

案例：对某超市销售数据（D:\我的桌面\超市营业额 2.xlsx）进行查看、预处理、分析。

#查看行下标为[3,5,10]，列下标为[0,1,4]的数据

```
>>> df.iloc[[3,5,10],[0,1,4]]
```

```

#查看行下标为 3, 5 的姓名、销售额列的值。
>>> df.loc[[3,5], ['姓名','交易额']]
#查看交易额高于 1700 元的销售记录
>>> df[df['交易额']>1700]
#查看下午班的交易总额
>>> df[df['时段']=='14: 00-21: 00']['交易额'].sum()
#查看交易额最大的 5 条记录
>>> df.nlargest(5, '交易额')
#按交易额和工号降序排序
>>> df.sort_values(by=['交易额','工号'], ascending=False)
#统计每个员工不同时段的交易额
>>> df.groupby(by=['姓名','时段'])['交易额'].sum()
#分别统计各柜台的销售总额
>>> df.groupby(by='柜台')['交易额'].sum()
#统计每个员工交易额平均值
>>> df.groupby(by='姓名')['交易额'].mean().sort_values()
#查看每位员工（以姓名分组）交易额的最高、最低、平均
>>> df.groupby(by='姓名')['交易额'].agg(['max','min', 'mean'])
#统计交易额低于 200 或高于 3000 的数量
>>> df[(df.交易额<200)|(df.交易额>3000)]['交易额'].count()
#使用整体均值的 80%填充缺失值
>>> df.fillna({'交易额': round(df['交易额'].mean()*0.8)}, inplace=True)
#删除重复的记录行
>>> df = df.drop_duplicates()
# 每天交易总额变化情况
>>> dff = df.groupby(by='日期').sum()['交易额'].diff()
#查看每人在各柜台的交易总额
>>> dff = df.groupby(by=['姓名','柜台'], as_index=False).sum()
>>> print(dff.pivot(index='姓名', columns='柜台', values='交易额'))

```

2. matplotlib 模块

(1) 假设学生的成绩信息存放于 Excel 文件 “成绩.xls” 中，包含的信息有：学号及各科目的课程名和成绩。编程实现：读取 “成绩.xls” 的成绩信息，并统计每位同学的总分，将总分在前 5 名的同学信息用柱形图表示，并保存为文件 top5.jpg

	A	B	C	D	E	F	G
1	学号	课程1	课程2	课程3	课程4	课程5	课程6
2	1	68	92	50	76	80	88
3	2	73	80	92	72	90	72
4	3	86	72	70	84	78	86
5	4	90	84	82	86	88	78
6	5	78	86	90	84	82	88
7	6	88	78	78	86	67	86
8	7	67	88	88	78	72	90
9	8	92	67	84	78	86	78
10	9	50	92	86	88	78	88

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_excel('d:\\我的桌面\\score.xlsx')
df['总分'] = df.iloc[:,1:5].sum(axis=1)
result=pd.DataFrame(df,columns=['学号','总分'])
top5=result.nlargest(5,'总分').sort_values('学号')
top5.index=list(range(len(top5)))
plt.figure()
top5.plot(x='学号',kind='bar')
plt.savefig('d:\\我的桌面\\top5.jpg')
```

(2) 读取文件“D:\我的桌面\超市营业额.xlsx”营业数据，绘制饼状图展示每月每个柜台营业额的在交易总额中的比例。

工号	姓名	日期	时段	交易额	柜台
1001	张三	2019-03-01	9: 00-14: 00	1, 664	化妆品
1002	李四	2019-03-01	14: 00-21: 00	954	化妆品
1003	王五	2019-03-01	9: 00-14: 00	1407	食品
1004	赵六	2019-03-01	14: 00-21: 00	1, 320	食品
1005	周七	2019-03-01	9: 00-14: 00	994	日用品
1006	钱八	2019-03-01	14: 00-21: 00	1421	日用品
1006	钱八	2019-03-01	9: 00-14: 00	1226	蔬菜水果
1001	张三	2019-03-01	14: 00-21: 00	1442	蔬菜水果
1001	张三	2019-03-02	9: 00-14: 00	1530	化妆品
1002	李四	2019-03-02	14: 00-21: 00	1395	化妆品
1003	王五	2019-03-02	9: 00-14: 00	936	食品
1004	赵六	2019-03-02	14: 00-21: 00	906	食品
1005	周七	2019-03-02	9: 00-14: 00	1444	日用品

```
import
pandas
as pd
import
```

```
matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['simhei']
df = pd.read_excel(r'D:\我的桌面\超市营业额 2.xlsx')
ddf = df.loc[:, ['柜台', '交易额']].groupby(by='柜台', as_index=False).sum()
ddf.plot(x='柜台', y='交易额', kind='pie', labels=df.柜台.values, autopct="%0.2f%%")
plt.show()
```

(3) 读取文件 “D:\我的桌面\score.xls” 中的成绩数据，接收任意输入的学号，将该学号对应的成绩，使用雷达图显示。

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_excel(r'D:\我的桌面\score.xlsx', index_col=0)
stu_id = input('请输入要查询的学号: ').strip()

scores = df.loc[int(stu_id)].values
scores = list(map(int,scores))

coures = df.columns
angles = np.linspace(0, 2*np.pi,len(coures), endpoint=False)
scores = np.append(scores,scores[0])
angles = np.append(angles, angles[0])

plt.polar(angles,scores,'rv--',linewidth=2)
plt.thetagrids(angles*180/np.pi,coures,fontproperties='simhei')
plt.fill(angles,scores,facecolor='r',alpha=0.3)
plt.show()
```