Python期末考试总复习资料

原创：林大夕可

👉阅读原文👈

2020.05.21

前言

先来听一个恐怖故事😲

> 在一个月黑风高的夜晚🌒，林大夕可同学埋头于昏黄的台灯下，为不久之后的Python期末考试做着复习。时间不多了，他把重点放在输入输出，字符串、列表、字典的操作上，至于函数与类，只是匆匆扫了一眼，他认为凭借着在字符串、列表、字典上较为熟练的操作，应该就勉强能及格了。
> 可万万没想到的是，不久之后的考场上，在浏览考题的那半分钟里，仿佛是有数道惊雷在耳边炸响——他复习的内容全都没考... 题目净是些诸如带有可变参数的函数、用类实现发牌等等。
> 这样一来，林大夕可同学的补考也就随之提上了日程。

这两个月以来，我断断续续地为Python补考做了很多准备👧，包括

1. 阅读*Python Cookbook*约1/3的内容
2. 阅读Python官方文档中的**入门教程**与部分**标准库参考**
3. 完成PTA上的**浙大版《Python 程序设计》题目集**与学校老师出的**5套题目集**

在经历了这个过程后，很多Python相关的知识都能够灵活运用了~ 🥳

然而到目前为止，学校还没通知补考时间和补考地点。再加上对`翻书马冬梅，考试孙红雷`的墙裂担忧，我便把各种知识点与题型进行了系统性的总结，等到补考（很可能安排在下学期）前也不用到时候再四处找零散的复习资料了，将这份总结看上一看、练上一练，应该会有事半功倍的效果。

同时，希望这份Python总结也能够帮助到很多正在为学校里的`Python考试`**（尤其是试题以编程题居多的考试）**而做着准备👨的小可爱们~

一、知识储备

1. 输入输出

1.1 输入

1.1.1 拆分输入数据

使用`map()`与`split()`将输入数据进行拆分

注：`map()`与`split()`的组合会自动忽略空格

```
a, b, c = map(int, input().split())     # split()中为将元素拆分的标志，默认为空格

value = b*b - 4*a*c

print(value)
```

1.1.2 设定截止位

只要`#`之前的输入内容

```
a = input()
s = ''
for i in a:
  if i == '#':
    break
  s = s + i     # 将'#'前的输入数据储存到s中
```

1.1.3 去调输入前后的空格

`strip()`：去掉元素前后方的空格，但中间的空格不去掉

```
s1 = input().strip()
s2 = input().strip()

print(s1)
print(s2)
```

1.1.4 所输即所得

`eval()`：直接将字符串当作有效的表达式（可以暂且理解为直接去掉字符串的双引号）

```
test = eval(input())
# 尝试输入内容1：[1, 2, 3, 4, 5]
# 尝试输入内容2：{'John': 89, 'Dutton': 95}

print(test)
print(type(test))     # 查看经过eval转化后的类型
```

```
result_dict = {"+": "x+y", "-": "x-y", "*": "x*y", "/": "x/y if y!=0  else
'divided by zero'"}

x = int(input())
op = input()
y = int(input())

result = eval(result_dict[op])
print(result)
```

1.2 输出

1.2.1 格式化输出 `format`

:	<填充>	<对齐>	<宽度>	,	<.精度>	<类型>
引导符号	用于填充的单个字符	< 左对齐 > 右对齐 ^居中对齐	输出宽度	千位分隔符	小数部分的精度	整数类型或浮点数类型

{index : space . decimal_precision conversion_type}

```
a, b = map(int, input().split(","))

for i in range(a, b):
  print("{:^6d} | {:^6d} | {:^6d} | {:^6d}".format(i, i**2, i**3, i**4))
```

1.2.2 end

```
# python默认输出会换行
# 使用end=''可以将前后输出内容连接在一行
print("Linda", end='_')
print("Silk")
```

---

2. 列表、元组、集合、字典

	列表 list	元组 tuple	集合 set	字典 dict
是否有序	是		否	
可否重复	是		否	否（键）
切片/索引	支持		不支持	
是否可变	可变	不可变	可变	可变
添加	append	/	add	d['k'] = 'v'

```
# 创建空列表
empty_list = []
empty_list = list()

# 创建空元组
empty_tuple = ()
empty_tuple = tuple()

# 创建空集合
empty_set = {}     # ❌错误！！！此方法用于创建字典
empty_set = set()

# 创建空字典
empty_dict = {}
empty_dict = dict()
```

2.1 列表

⚠️注意：像 `insert`，`remove` 或者 `sort` 能够修改列表中元素的方法，没有打印出返回值（它们返回默认值 `None`，这是Python中所有可变数据结构的设计原则）。

举个栗子🌰

```
hosts = ['Dolores', 'Angela', 'Charlotte']

# 1. insert修改了原来的列表
hosts.insert(1, 'William')
print(hosts)                  # 直接调用原列表名，会得到修改后的列表

new_hosts = hosts.insert(2, 'Teddy')     # 若创建一个变量存放返回结果，变量的值会变为None
print(new_hosts)


# 2. index并未修改原来的列表
position = hosts.index('Dolores')      # 可创建一个变量存放返回结果
print(position)
```

2.1.1 增、删、改、查

①. 增加/插入

1. `append()`：在list的最后加入一个`元素`

```
courses = ['History', 'Math', 'Physics']

courses.append('Art')
```

2. `extend()`：在list的最后加入一个`列表`

```
courses = ['History', 'Math', 'Physics']
courses_2 = ['Art', 'Education']

courses.extend(courses2)
```

3. `insert()`：在list的某个位置加入一个`元素`

```
courses = ['History', 'Math', 'Physics']

courses.insert(0, 'Art')
```

②. 删除

1. `remove()`：删除list中的特定元素

```
courses = ['History', 'Math', 'Physics']

courses.remove('Math')
```

2. `pop()`：移除list最后一个元素

```
courses = ['History', 'Math', 'Physics']

courses.pop()
```

③. 修改

```
courses = ['History', 'Math', 'Physics']

courses[0] = 'Art'
```

④. 查询

1. `index()`：查询某元素的在列表中的位置（索引）

```
courses = ['History', 'Math', 'Physics', 'CompSci']

print(courses.index('CompSci'))
```

2. `in`：查询某元素是否在列表中

```
courses = ['History', 'Math', 'Physics', 'CompSci']

print('Art' in courses)
```

列表中用到：`index()`，`in`

字符串中用到：`find()`，`count()`，`in`

- 使用`find()`时，若查询的元素存在，返回首次出现的**位置索引**；若不存在，返回 **-1**
- 使用`count()`时， 若查询元素出现，返回出现**次数**；若未出现，返回 **0**
- 使用`in`时，返回值为**True**或**False**

2.1.2 切片

语法：`list_name [start_index : stop_index : step]`

```
courses = ['History', 'Math', 'Physics', 'CompSci']

print(courses[0:2])

# step
print(courses[0:3:2])

# 若step为负，则代表反向
print(courses[-1:0:-1])


# 理解切片
# 将索引视作指向字符之间 ， 第一个字符的左侧标为0，最后一个字符的右侧标为n，其中n是字符串长度。
# +---+---+---+---+---+---+
```

```
# | P | y | t | h | o | n |
# +---+---+---+---+---+---+
# 0   1   2   3   4   5   6

# 注意：切片的开始总是被包括在结果中，而结束不被包括 <range()与之相同>。这使得s[:i] + s[i:]总是等于s
# word = 'Python'
# print(word[:2] + word[2:])
```

2.1.3 排序

1. `sort()` / `sorted()` 顺序

```
courses = ['Math', 'History', 'Physics', 'CompSci']


# 1. sort()
# sort()会改变原列表顺序
courses.sort()
print(courses)


# 2. sorted()
# sorted()不会改变原列表顺序，它会返回一个排序后的版本
# 需创建一个变量来接收返回值
sorted_courses = sorted(courses)
print(sorted_courses)
```

2. `reverse()` / `reversed()` / `sort()` / `[::-1]` 反转 / 倒序

```
courses = ['Math', 'History', 'Physics', 'CompSci']


# 1. reverse()
# reverse()会改变原列表顺序
courses.reverse()
print(courses)


# 2. reversed()
# sorted()不会改变原列表顺序，它会返回一个排序后的版本
# 需创建一个变量来接收返回值
reversed_courses = reversed(courses)
print(reversed_courses)
```

```
# 3. sort()
# 用包含申明的sort方法
courses.sort(reverse = True)
print(courses)


# 4. [::-1]
print(courses[::-1])
```

2.1.4 去重

使用`set()`，因`集合`中没有重复的元素

```
num_list = [1, 2, 3, 1, 2, 4, 2, 5, 3, 2, 6, 7]

print(set(num_list))

# 若要将去重的结果以列表表示
print(list(set(num_list)))
```

2.1.5 列表与字符串间的转化

列表 --> 字符串：`join()`

字符串 --> 列表：`split()`

```
courses = ['History', 'Math', 'Physics', 'CompSci']

# 以逗号和空格结合列表中的元素，将其变为一长串字符串
course_str = ', '.join(courses)

# 使用split()将字符串重新变换为列表
new_list = course_str.split(', ')
```

2.1.6 列表推导式

列表推导式的结构是由一对方括号所包含的以下内容：一个表达式，后面跟一个 `for` 子句，然后是零个或多个 `for` 或 `if` 子句。

其结果将是一个新列表，由对表达式依据后面的 `for` 和 `if` 子句的内容进行求值计算而得出。

举个栗子🌰

```python
# 为了创建一个求平方的列表
# 1. 普通方法
squares = []

for x in range(10):
    squares.append(x**2)

print(squares)


# 2. 列表推导式
squares = [x**2 for x in range(10)]     # for前方的内容为返回值

print(squares)
```

```python
# 用列表推导式来将输入值处理为矩阵
m, n = map(int, input().split())    # m为行，n为列，其间用空格隔开
matrix = []

for i in range(m):                  # 重复m次（m行）
    s = input()                     # 输入n个数，每个数之间用空格隔开
    matrix.append([int(n) for n in s.split()])    # ⚠️注意：将每个元素转化为整型

print(matrix)
```

2.1.7 最值 / 求和

对于`可迭代对象`，可以直接用`sum()`，`max()`，`min()`方法来进行快速计算

```python
nums = [1, 5, 2, 6, 3, 4]

print(min(nums))
print(max(nums))
print(sum(nums))
```

2.2 元组

```python
courses = ('History', 'Math', 'Physics')

print(courses)

courses[0] = 'Art'      # ❌错误：元组内元素为只读，不可更改

print(courses)
```

2.3 集合

```python
courses = {'History', 'Math', 'Physics', 'CompSci', 'Math'}

# 重复执行多次，输出结果不同
# 因set不注重内部元素顺序
# set会自动消除元素重复部分
print(courses)

# 当判断某元素是否在一群元素中时，set是首选项
print('Math' in courses)
```

集合间的运算

```python
cs_courses = {'History', 'Math', 'Physics', 'CompSci'}
art_courses = {'History', 'Math', 'Art', 'Design'}

# 交集（共有的元素）
print(cs_courses.intersection(art_courses))

# 不同（独有的元素）
print(cs_courses.difference(art_courses))

# 并集（所有的元素）
print(cs_courses.union(art_courses))
```

2.4 字典

从某种程度上来说，可以将字典看做一个自定义`索引（index）`的列表

- 列表中的某个元素可以表示为：`list[index]`
- 字典中的某个值可以表示为：`dict[key]`

```python
student = {'name': 'John', 'age': 25, 'courses': ['Math', 'CompSci']}
```

```python
# get方法可以使得在字典中缺少要查询的值时显示出默认或自定义的结果，而不是抛出异常
# 括号内第二个参数为自定义默认输出内容
print(student.get('name'))
print(student.get('phone', 'Sorry. Not Found~'))


# ------------------------
# 添加
student['phone'] = '199-8011-8011'
# 更新
student['name'] = 'Jane'

print(student)


# ------------------------
# 使用update可以一次更新多个值！！！
student.update({'name': 'Silk', 'age': 100, 'phone': '188-0525-7633'})

print(student)


# ------------------------
# 删除 方法一：使用del
del student['age']
# 删除 方法二：使用pop
# pop会删除该值，但也会同时返回删除的值，需用变量来接收
age = student.pop('age')

print(student)
print(age)


# ------------------------
# 查看长度
print(len(student))
# 查看键
print(student.keys())
# 查看值
print(student.values())
# 成对展现键与值
print(student.items())
```

重点

```
# 以键循环
for key in student:
    print(key)

# 键、值循环
for key, value in student.items():
    print(key, value)
```

---

3. 函数、类、异常

3.1 函数

函数参数中`arg`表示：传入的一个参数

函数参数中有单星号的`*args`表示：将参数以`元组`的形式导入（不限个数）

函数参数中有双星号的`**kwargs`表示：将参数以`字典`的形式导入（不限个数）

```
from datetime import date


def mission(arg, *args, **kwargs):
    information = []
    for i in args:
        information.append(i)

    print('New Mission')
    print('------------------------------')
    print('System date: {}'.format(arg))
    print('Location: {}, {}'.format(information[0], information[1]))
    print('Mission ID: {}'.format(information[2]))
    print('------------------------------')

    for k, v in kwargs.items():
        print('{:8}........ {}'.format(k, v))


mission(date.today(), 'Berlin', 'German', 293821, AGENT='Jason Bourne',
GENDER='Male', PROJECT='Treadstone')
```

3.2 类

举个栗子🌰

```
class MyDog:
    kind = 'Labrador'

    def __init__(self, name):
        self.name = name
        self.tricks = []      # 实例变量（每个实例的变量都不尽相同）

    def add_tricks(self, *trick):
        for i in trick:
            self.tricks.append(i)


dog1 = MyDog('Tommy')
dog2 = MyDog('Tom Me')
dog1.add_tricks('Bark', 'Slaughter')
dog2.add_tricks('Embark', 'Laughter')

print('My Dogs Info:')
print('-------------')
print('Dog1: {}--{}--{}'.format(dog1.kind, dog1.name, dog1.tricks))
print('Dog2: {}--{}--{}'.format(dog2.kind, dog2.name, dog2.tricks))
```

3.3 异常

```
a = input('Enter a number: ')

try:
    b = 5/int(a)
    print(b)
except ZeroDivisionError as err:        # 输入为0时执行
    print("There's a problem:", err)
except ValueError:                      # 输入不为数字时执行
    print('You entered a wrong value!')
else:                           # 输入为非零数字时执行
    print('try语句能顺利执行时，打印此句~')
```

4. 一些常用的方法

4.1 判断数字、字母、大小写、空格

判断是否为数字：`isdigit()`；判断是否为字母：`isalpha()`

是否为大写字母：`isupper()`；是否为小写字母：`islower()`

判断是否为空格：`isspace()`

```
# ~ 原方法 ~
# 使用列表储存数字，随后再转化为字符串，最后转化为整形
a = input()
b = []              # 列表存放数据
for i in a :
   if i.isdigit() :
      b.append(n)
print(int("".join(b)))


# ~ 改进版方法 ~
# 字符串可以通过+来连接。因此，可以用字符串来存储数字。之后就不用再进行转化了
a = input()
b = ''              # 字符串存放数据
for i in a:
  if i.isdigit():
    b += i           # 直接通过'+'将字符连接
print(b)
```

4.2 编码与转化

4.2.1 UTF-8编码

使用`encode()`方法

注意：编码格式需要被引号包含

```
s = '人生苦短，我用Python'

print(s.encode('UTF-8'))
```

4.2.2 ASCII码与字符的互换

使用`ord()`与`chr()`

```
# 字符 --> ASCII
character = 'z'
print(ord(character))

# ASCII --> 字符
ascii = 126
print(chr(ascii))
```

有时还需根据ASCII码的相加关系来做出一些限定，如'A' + 'Z' = 155

4.2.3 进制转化

①. 其它进制转化为十进制

使用`int(x, y)`将其他进制转化为十进制。其中x为原值，y为原进制

```
# 将8进制的56转化为十进制
print(int(56, 8))    # 56需要加上''，即转化为字符串形式
```

②. 十进制转化为其它进制

```
# 转化为二进制：bin
print(bin(1024))

# 转化为八进制：oct
print(oct(8732))

# 转化为十六进制：hex
print(hex(2918))
```

4.3 判断某个字符（串）是否属于另一个字符串

```
string_1 = input('1st string: ')
string_2 = input('2nd string: ')

# 方法一：使用find()
# 如果字符串1包含字符串2，返回字符串2在1中的起始索引
# 如果不包含，则返回❀'-1'❀
print(string_1.find(string_2))


# 方法二：使用count()
```

```
# 若出现，返回出现次数；若未出现，返回0
print(string_1.count(string_2))


# 方法三：使用in
# 返回值为'True'或'False'
print(string_2 in string_1)
```

4.4 替换

```
message = "I like you!"

trim = message.replace("l", "n")

print('Then:', message)     # 由此可知，replace并不改变原字符串。故需要创建新变量来接收
print('Now:', trim)
```

4.5 获取帮助

使用`dir()`或`help()`来查询方法该如何使用

```
greeting = 'Hello'
name  = 'Silk'

# dir()能显示括号内变量所有操作的函数、方法
print(dir(name))


# help()能显示某函数/方法的具体用法
# 括号内不能是变量名
print(help(str))

# 查询具体的某函数/方法
print(help(str.lower))
```

---

5. 一些常用的库

5.1 math

①. 平方根：`sqrt()`

```
# 先导入math
import math

a, b, c = map(int, input().split())

s = (a + b + c)/2
# 使用sqrt()求平方根
area = math.sqrt(s*(s - a)*(s - b)*(s - c))
```

②. 幂：`pow(x, y)`

```
import math

# 使用pow()计算3**4
d = math.pow(3, 4)
```

③. π：`pi`

```
# 使用pi获取π值
num = math.pi
```

5.2 random

①. `random()`：返回一个在区间`[0.0, 1.0)`的随机浮点数

```
import random

for i in range(10):
    print(random.random())
```

②. `randint(a, b)`：返回一个在区间`[a, b]`的随机整数，相当于`randrange(a, b+1)`

```python
import random

for i in range(10):
    print(random.randint(1, 15))
```

③. **`randrange(start, stop[, step])`**：从 `range(start, stop, step)` 返回一个随机选择的元素

```python
import random

for i in range(10):
    print(random.randrange(5, 120, 10))
```

④. **`seed()`**：获取某种特定生成的随机数
当`seed()`无参数时，每次生成的随机数不同，
当`seed()`有参数时，若参数相同，则每次生成的随机数相同；若参数不同，则生成的随机数也不同

```python
import random

ls = [1, 2, 3, 4, 5, 6, 7, 8]

for i in range(8):
    random.seed(4)              # 试着改变seed()中的数或者注释掉这一行，看看输出的变化
    print(random.randint(1, 10))  # randint(a, b)随机生成[a, b]间的整数

# 注意⚠️：randint与range取值范围的区别。前者能取右侧的值，后者不能
```

⑤. **`shuffle()`**：将一个序列（列表、字符串或元组）的顺序打乱

```python
import random

mylist = ["apple", "banana", "cherry"]
random.shuffle(mylist)

print(mylist)
```

5.3 datetime

获取今天的日期

```
from datetime import date

day = date.today()
print(day)
```

二、题目练习

1. 语言相关

1.1 打印一些形状

1.1.1 四种直角三角型

①. 直角在左上方

```
# 打印形状 #
*****
****
***
**
*

# 实现代码 #
# 输入层数
a = int(input('请输入层数：'))

for i in range(a):
    print('*'*(a - i))       # 若要使打印出的星号✳之间有空格，将'*'改为'* '即可
```

②. 直角在左下方

```
# 打印形状 #
*
**
***
****
*****

# 实现代码 #
# 输入层数
a = int(input('请输入层数: '))

for i in range(a):
    print('*'*(i + 1))
```

③. 直角在右上方

```
# 打印形状 #
*****
 ****
  ***
   **
    *

# 实现代码 #
# 输入层数
a = int(input('请输入层数: '))

# 可以将第三第四种三角形看作一个由' '和'*'组成的矩形,
# >>> 因此, 每行的' '和'*'数量相加必等于a <<<
for i in range(a):
    print(' '*i + '*'*(a - i))
```

④. 直角在右下方

```
# 打印形状 #
    *
   **
  ***
 ****
*****

# 实现代码 #
# 输入层数
a = int(input('请输入层数: '))
```

```
# 可以将第三第四种三角形看作一个由' '和'*'组成的矩形，
# >>> 因此，每行的' '和'*'数量相加必等于a <<<
for i in range(a):
    print(' '*(a - 1 - i) + '*'*(i + 1))
```

1.1.2 两种九九乘法表

①. 常规

九九乘法表重点在于由两个for循环控制输出
核心是第二个循环的范围由第一个循环的参数决定

```
1*1=1
1*2=2   2*2=4
1*3=3   2*3=6   3*3=9
1*4=4   2*4=8   3*4=12  4*4=16
1*5=5   2*5=10  3*5=15  4*5=20  5*5=25
1*6=6   2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7   2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8   2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9   2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```



```
# 输入层数
a = int(input('请输入层数: '))

for i in range(1, 10):
    for j in range(1, i+1):
        print('{}*{}={:<5d}'.format(j, i, i*j), end='')    # 使用format格式化输出
    print()
```

②. 非常规

```
1*1=1   1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
2*2=4   2*3=6   2*4=8   2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*3=9   3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*6=36  6*7=42  6*8=48  6*9=54
7*7=49  7*8=56  7*9=63
8*8=64  8*9=72
9*9=81
```

```
# 输入层数
a = int(input('请输入层数：'))

for i in range(1, 10):
    for j in range(i, 10):
        print('{}*{}={:<5d}'.format(i, j, i*j), end='')    # 使用format格式化输出
    print()
```

1.1.3 三种金字塔

①. 正金字塔

```
# 打印形状 #
    *
   ***
  *****
 *******
*********

# 实现代码 #
# 将金字塔看作一个由左边矩形、右边三角形组成的图形(拆分如下)
# for i in range(a):
#     print('~'*(a - 1 - i) + '*'*(i + 1) + '  ' + '*'*i)

# 输入层数
a = int(input('请输入层数：'))

for i in range(a):
    print(' '*(a - 1 - i) + '*'*(i + 1) + '*'*i)
```

②. 倒金字塔

```
# 打印形状 #
*********
 *******
  *****
   ***
    *

# 实现代码 #
# 输入层数
a = int(input('请输入层数：'))

for i in range(a):
    print(' '*i + '*'*(a - i) + '*'*(a - 1 - i))
```

③. 数字金字塔

```
# 打印形状 #
          0
        1 0 1
      2 1 0 1 2
    3 2 1 0 1 2 3
  4 3 2 1 0 1 2 3 4
5 4 3 2 1 0 1 2 3 4 5

# 实现代码 #
# 输入层数
a = int(input('请输入层数：'))

for i in range(a+1):
    # 调节数字前方空格数量
    # 可针对层数为一位数、两位数的金字塔打印
    if i < 10:
        print(' '*(a - 8) + '  '*(a + 1 - i), end='')
    else:
        print('   ' * (a + 1 - i), end='')

    # 输出金字塔左半部分（包括0）
    j = i
    while j != -1:
        print(j, end=' ')
        j = j - 1

    # 输出金字塔右半部分（不包括0）
    j = 1
    while j != i + 1:
        print(j, end=' ')
```

```
        j = j + 1

    print()
```

1.2 猜词

①. 题目🔍

随机给出一个1~100之间的数字，然后让玩家猜这个数字。如果没有猜出正确答案，给出“`大了`”还是“`小了`”的提示；如果猜出正确答案，打印出“`恭喜，你猜了x次猜对了答案！`”。
每个玩家有5次猜数字的机会，如果5次都没有猜对，给出“`是否继续游戏`”的提示。

输出样例

```
猜数字游戏！（请给出1-100之间的数字）
请输入你猜的数字：69
猜的数字大了...
请输入你猜的数字：28
猜的数字小了...
请输入你猜的数字：60
猜的数字小了...
请输入你猜的数字：65
猜的数字小了...
请输入你猜的数字：68
猜的数字大了...
次数用完啦！是否重新开始？(y/n):y
猜数字游戏！（请给出1-100之间的数字）
请输入你猜的数字：60
猜的数字大了...
请输入你猜的数字：50
猜的数字小了...
请输入你猜的数字：55
恭喜，你猜了3次猜对了答案
是否重新开始猜数字游戏？(y/n)：n

Process finished with exit code 0
```

②. 题解✨

思路

把用于存放输入是否继续（`y`或者`n`）的变量设定为`y`，然后将其当做是否进行while循环的标准

代码

```
import random

# 将输入赋值为'y'
inp = 'y'

while inp == 'y':
    print('猜数字游戏！（请给出1-100之间的数字）')
    generate = random.randint(1, 100)
    # print(generate)                                                    # 显示生成数，方便
测试

    # 5次机会
    for flag in range(6):                                                # 范围[0, 5]是为
了将次数用完后的输出部分放入循环来判断
        if flag == 5:
            inp = input('次数用完啦！是否重新开始？(y/n):')
            break
        else:
            num = int(input('请输入你猜的数字：'))
            if num < generate:
                print('猜的数字小了...')
            elif num > generate:
                print('猜的数字大了...')
            else:
                print('恭喜，你猜了{}次猜对了答案'.format(flag + 1))
                inp = input('是否重新开始猜数字游戏？(y/n)：')
                break
```

1.3 字符串、列表操作

1.3.1 删除重复字符串

①. 题目

本题目要求找出输入字符串中重复的字符，并输出去掉重复字符后的字符串以及重复的字符是哪些，重复了几次。

1.1 输入格式

输入一个字符串。

1.2 输出格式

输出去掉重复字符的字符串，每个重复的字符重复了几次。

1.3 输入样例

> ajlrihvnvgugnmkh

1.4 输出样例

```
    ajlrihvngumk
v 2
g 2
n 2
h 2
```

②. 题解✨

2.1 思路

关键 创建两个空字符串和一个空字典📔

- `字符串1`用来存储输入字符串中没有重复的字符
- `字符串2`用来存储剩下的字符（重复的字符）
- 将`字符串2`的字符当作key🔑、字符的重复次数当作value依次存入`字典`

2.2 代码

```
string = input()
result_str = ''
redundant_str = ''
dic = {}

for i in string:
    if i not in result_str:
        result_str += i      # 将非重复的字符依次存入result_str
    else:
        redundant_str += i    # 将重复的字符依次存入redundant_str

for i in redundant_str:      # 将redundant_str中的元素存入字典
    if i not in dic:
        dic[i] = 2         # 元素已经在result_str中存在，故redundant_str中出现的应该记为第二个
    else:
        dic[i] += 1        # 若元素已在dic中存在，键值 +1

print(result_str)

for k, v in dic.items():     # 遍历字典，输出键值对
    print('{} {}'.format(k, v))
```

1.3.2 求单词长度并排序

①. 题目🔍

输入n个单词，计算每个单词长度。对单词长度排序，分行输出单词长度及其单词。

1.1 输入格式

行1：单词个数n
分行输入n个单词

1.2 输出格式

分行输出单词长度及其单词。（单词长度，单词）用元组表示

1.3 输入样例

```
5
python
list
set
996
tuple
```

1.4 输出样例

```
(3, '996')
(3, 'set')
(4, 'list')
(5, 'tuple')
(6, 'python')
```

②. 题解✨

2.1 思路

关键 创建一个空字符串和一个空字典📔

- `字符串` 用来存储依次输入的字符串
- 将`字符串`排序后（将形如'996'的数字字符串排到字母字符串'set'前面）的元素当作key🔑、元素（每个字符）的长度当作value依次存入`字典`
- 将`字典`以值的大小排序（参考：lambda表达式的用法）

⚠️注意： `字典`按值排序后返回值是元素为`集合`的`列表`，每个集合即为原字典中的一个键值对。

2.2 代码

```
a = int(input())
ls = []
dic = {}
```

```
for i in range(a):
    temp = input()
    ls.append(temp)    # 将输入字符串依次存入ls

for i in sorted(ls):  # 对ls排序后进行迭代
    dic[i] = len(i)    # 将ls中的元素与对应的长度存入字典


new_dic = sorted(dic.items(), key=lambda item: item[1])    # 对字典按值排序

for i in new_dic:
    print("({}, '{}')".format(i[1], i[0]))
```

1.3.3 学生成绩处理

①. 题目🔍

小明在帮助老师统计成绩，老师给他的是一组数据。数据的第1行代表学生数n，后面的n行代表每个学生的成绩。成绩是整数类型。
小明编写了一个程序，该程序可以批量处理数据，统计所有学生的平均分。当数据没有任何错误时，提示'All OK'，当数据有一些错误(某行是浮点数、某行是非整数字符)时，可以提示哪些数据出错，并最后提示第几行出错，出错的原因，共出错多少行。对于另一些小错误，如某行虽然是整数，但是左右有多余的空格，可以将空格去除掉进行正常统计。

在最后输出：
共处理多少行数据，几行正确，几行错误，平均成绩(保留两位小数)。

进阶要求：
有可能碰到要求输入n行，后面的数据却小于n行。要求处理这种情况。碰到这种情况。
输出end of files，并统计现有行数。见样例3

1.1 输入样例1

```
3
1
    2
3
```

1.2 输出样例1

```
Total: 3
OK: 3
Error: 0
avg grade = 2.00
```

1.3 输入样例2

```
5
1
  2
   a
  b  5
3
```

1.4 输出样例2

```
line 3 error for input "    a   "
line 4 error for input "  b  5"
Total: 5
OK: 3
Error: 2
avg grade = 2.00
```

1.5 输入样例3

```
5
 a
 2
3
```

1.6 输出样例3

```
line 1 error for input " a"
end of files
Total: 3
OK: 2
Error: 1
avg grade = 2.50
```

提示：对于样例3，如果是在IDLE中进行测试，可在输入最后一行并回车后，按`Ctrl+D`结束输入。

②. 题解✨

2.1 思路

- 通过第一行的输入数字来作为循环，确定随后输入的行数
- 通过`strip()`来去掉输入部分多余的空格
- 通过`isdecimal()`来判断是否为整数
- 用`字典`将格式错误的行数与对应的内容存入，之后再格式化输出

注：暂未完成**进阶要求**中的内容

2.2 代码

```
total_num = int(input())
ok = 0
error = 0
total = 0
dic = {}

for i in range(total_num):
    temp = input()
    trim = temp.strip()
    if trim.isdecimal():
        ok += 1
        total += int(trim)
    else:
        dic[i + 1] = temp
        error += 1

for k, v in dic.items():
    print('line {} error for input "{}"'.format(k, v))
print('Total:', total_num)
print('OK:', ok)
print('Error:', error)
print('avg grade = {:.2f}'.format(total/ok))
```

1.4 可变参数传递

本题含有计算年龄的方法

①. 题目🔍

本题要求实现一个函数，可输出姓名、出生日期、性别、国籍和电话，并根据出生日期算出岁数（需要判断是否足岁）。函数可以对输入中的错误信息进行捕获。

1.1 函数接口定义

```
def student(name,*birth,**information)
```

1.2 裁判测试程序样例

```
name = input()
birth = input()
student(name,birth,sex='Female', nation='China', phone='123456789')
```

1.3 输入样例1

> zhangsan
> 1999 2 3

1.4 输出样例1

> name:zhangsan
> birth:1999-2-3
> age is 20
> sex:Female
> nation:China
> phone:123456789

1.5 输入样例2

> zhangsan
> 1999-2-3

1.6 输出样例2

> name:zhangsan
> The interval in the input 'birth' is a space

②. 题解✨

2.1 知识储备

函数参数前的单星号`*args`表示：将参数以元组的形式导入
函数参数前的双星号`**kwargs`表示：将参数以字典的形式导入

2.2 代码

```
from datetime import date


def student(name, *birth, **information):
    string = birth[0]       # 将出生日期转化为字符串
    if ' ' not in string:     # 这里仅判断出生日期不以空格为间隔的情况
        print('name:{}'.format(name))
        print("The interval in the input 'birth' is a space")
    else:
        ls = string.split()     # 将字符串转化为列表
        year = ls[0]        # 切片储存
        month = ls[1]
        day = ls[2]
        today = date.today()        # 获取今天的日期
        age = today.year - int(year) - ((today.month, today.day) < (int(month),
int(day)))      # ⚠️重要！！！计算年龄
```

```
        print('name:{}'.format(name))
        print('{}-{}-{}'.format(year, month, day))
        print('age is', age)
        for k, v in information.items():
            print('{}:{}'.format(k, v))


name = input()
birth = input()
student(name, birth, sex='Female', nation='China', phone='123456789')
```

1.5 圆球类、学生类

1.5.1 圆球类计算圆周长面积、球表面积体积

①. 题目🔍

编写程序，创建类MyMath，计算圆的周长、面积和球的表面积和体积，结果均保留两位小数。若输入的是非数字，则输出：请输入数字！
提示：要引入math包

1.1 输入样例1

输入半径

> 5

1.2 输出样例1

> 圆的周长 = 31.42
> 圆的面积 = 78.54
> 球的表面积 = 314.16
> 球的体积 = 523.60

1.3 输入样例2

> s

1.4 输出样例2

> 请输入数字！

②. 题解✨

2.1 知识储备

球的体积：V = 4πR³/3
球的面积：S = 4πR^2
表面积：Surface Area / 体积：Volume / 周长：Perimeter / 面积：Area

2.2 代码

```
import math


class MyMath:
    def __init__(self, r):
        self.r = r

    def perimeter(self):
        print('圆的周长 = {:.2f}'.format(2*math.pi*r))

    def area(self):
        print('圆的面积 = {:.2f}'.format(math.pi*math.pow(r, 2)))

    def surface(self):
        print('球的表面积 = {:.2f}'.format(4*math.pi*math.pow(r, 2)))

    def volume(self):
        print('球的体积 = {:.2f}'.format(4/3*math.pi*math.pow(r, 3)))


r = input()

if r.isdigit():
    r = int(r)
    p1 = MyMath(r)
    p1.perimeter()
    p1.area()
    p1.surface()
    p1.volume()
else:
    print('请输入数字！')
```

1.5.2 学生类计算年龄、成绩等级

①. 题目🔍

编写个学生类`Student`：

- 包含姓名、出生日期和成绩`属性`（数据成员）；
- 包含一个用于给定数据成员初始值的`构造函数`；

- 包含一个可计算学生年龄的`方法`（学生年龄判断根据日期是否超过生日的年、月、日）；
- 包含一个将成绩对应成等级的`方法`；
- 包含一个输出“**姓名+年龄+成绩等级**”的`方法`。

1.1 输入样例1

```
Name: LindaSilk
Birth: 2016-08-24
Grade(0~100): 99
```

1.2 输出样例1

```
姓名：LindaSilk  年龄：3  成绩：优
```

1.3 输入样例2

```
Name: 林大夕可
Birth: 2000-02-29
Grade(0~100): 85
```

1.4 输出样例2

```
姓名：林大夕可  年龄：20  成绩：良
```

②. 题解✨

2.1 代码

```
from datetime import date


class Student:
    def __init__(self, name, birth, grade):
        self.name = name
        self.birth = birth
        self.grade = grade

    def calculate_age(self):
        today = date.today()                    # 获取今天的日期
        year, month, day = map(int, self.birth.split('-'))    # 将birth以'-'拆分储存为年、月、日
        age = (today.year - year - ((today.month, today.day) < (month, day)))
        return age
```

```
    def classify(self):
        dic = {
            '优': [90, 100],
            '良': [80, 89],
            '中': [70, 79],
            '及格': [60, 69],
            '不及格': [0, 59]
        }                               # 用字典方法判断等级
        for k, v in dic.items():
            if v[0] <= self.grade <= v[1]:
                return k

    def out(self):
        print('姓名: {}  年龄: {}  成绩: {}'.format(self.name,
self.calculate_age(), self.classify()))


name = input('Name: ')
birth = input('Birth: ')
grade = input('Grade(0~100): ')

s1 = Student(name, birth, int(grade))
s1.out()                                # 调用Student类中的out方法
```

---

2. 算法相关

2.1 素数

1. 知识储备

质数（Prime Number），又称素数，是指大于1的自然数中，除了1和它本身，没有其他因数的数

2. 代码

```
number = int(input('请输入一个数: '))


def is_prime(num):
    if num > 1:
        for i in range(2, num):
            if num % i == 0:
                return False
        else:
            return True
    else:
        return False
```

```
if is_prime(number) is True:
    print('{}是素数'.format(number))
else:
    print('{}不是素数'.format(number))
```

2.2 斐波那契数

①. 知识储备📫

斐波那契数列（Fibonacci sequence），又称黄金分割数列。因意大利数学家Leonardo Fibonacci以兔子繁殖为例子而引入，故又称为“兔子数列”
指的是这样一个数列：1、1、2、3、5、8、13、21、34、……

在数学上，斐波那契数列被以递推的方法定义：
$F(1) = 1, F(2) = 1, F(n) = F(n - 1)+F(n - 2) (n \geq 3, n \in N^*)$

②. 几种情形🏞️

2.2.1 求第n个斐波那契数（核心）

运行效果✨

```
请输入数字：10
第10个斐波那契数：55
```

方法一：使用for循环

```
num = int(input('请输入数字：'))


def fib1(n):
    a, b = 1, 1
    for j in range(n - 1):      # ♻️
        a, b = b, a + b
    return a


print('第{}个斐波那契数：{}'.format(num, fib1(num)))
```

方法二：使用递归

```
num = int(input('请输入数字：'))


def fib2(n):
    if n == 1 or n == 2:
        return 1
    else:
        return fib2(n - 1) + fib2(n - 2)


print('第{}个斐波那契数：{}'.format(num, fib2(num)))
```

2.2.2 求前n个斐波那契数

思路：在上方函数的基础上加上一个for循环即可

运行效果✨（格式化输出）：

```
请输入数字：14
前14个斐波那契数为：
       1       1       2       3       5
       8      13      21      34      55
      89     144     233     377
```

实现代码👨🏻‍💻：

```
num = int(input('请输入数字：'))


# 直接使用上面提到的fibonacci函数
def fib1(n):
    a, b = 1, 1
    for j in range(n - 1):
        a, b = b, a + b
    return a


if num <= 0:
    print('无效的输入！')
else:
    print('前{}个斐波那契数为：'.format(num))
    for i in range(1, num + 1):
        print('{:8}'.format(fib1(i)), end='')     # 格式化输出，每个结果占8位
        if i % 5 == 0:                  # 每输出5个，换一行
            print()
```

2.2.3 求大于某个数的最小斐波那契数

运行效果✨：

```
请输入数字：128
大于128的最小Fibonacci数是：144
它是数列中的第12个数
```

实现代码👨‍💻：

```
num = int(input('请输入数字：'))


# 直接使用上面提到的fibonacci函数
def fib2(n):
    if n == 1 or n == 2:
        return 1
    else:
        return fib2(n - 1) + fib2(n - 2)


if num <= 0:
    print('无效的输入！')
else:
    for i in range(1, num + 1):
        if fib2(i) > num:
            print('大于{}的最小斐波那契数是：{}\n它是数列中的第{}个数'.format(num,
fib2(i), i))
            break
```

2.3 最大公约数、最小公倍数

2.3.1 最大公约数

最大公约数（Greatest Common Divisor）缩写为GCD
这里求最大公约数的方法为`辗转相除法`

```
a, b = map(int, input('请输入两个数字，其间用空格隔开：').split())
big = max(a, b)                    # 比较两个数，使得big > small
small = min(a, b)
c = big % small                    # 将big作被除数，small做除数，相除后余数为c

while c != 0:
    big = small
    small = c
    c = big % small

print('{}和{}的最大公约数是：{}'.format(a, b, small))
```

2.3.2 最小公倍数

最小公倍数（Least Common Multiple）缩写为LCM
求最小公倍数的方法：两数之积除以最大公约数

```
lcm = int(a*b/small)
print('{}和{}的最小公倍数是：{}'.format(a, b, lcm))
```

2.4 折半查找（二分查找）

①. 题目🔍

本题要求采用折半查找的思想，每次搜索原来数据的一半，直到搜索成功或待搜索数据为空。

1.1 输入格式

输入一个列表A和查找的值B。

1.2 输出格式

如果查找成功输出数B在列表A中的位置，否则输出查找不成功。

1.3 输入样例1

> [19,23,46,49,65,78,98,101,125]
> 46

1.4 输出样例1

> 46 2

1.5 输入样例2

> [19,23,46,49,65,78,98,101,125]
> 82

1.6 输出样例2

> not find

②. 题解✨

2.1 思路

使用`eval()`将输入的值转化为列表，使用`sorted()`将列表排序

关键 创建一个折半查找的函数，传入值为列表A与数字B

折半查找判断方法：

> 将列表中间位置的值与查找值比较
>
> - 如果两者**相等**，则查找成功；否则，利用中间位置的值将表分成前、后两个子列表，
> - 如果**中间位置**记录的值**大于**查找值，则进一步查找**前**一子列表；
> - 如果**中间位置**记录的值**小于**查找值，则进一步查找**后**一子列表。
>
> ---
>
> 重复以上过程，直到找到满足条件的记录，使查找成功，或直到子表不存在为止，此时查找不成功。

2.2 代码

```
a = sorted(eval(input()))    # 将输入转化为列表，并排序
b = int(input())


def find(ls, item):
    low = 0
    high = len(ls)
    while low < high:
        mid = int((low + high)/2)
        temp = ls[mid]
        if temp == item:
            return mid
        elif temp > item:
            high = mid - 1
        else:
            low = mid + 1
    return False


if find(a, b) is False:
    print('not find')
else:
    print('{} {}'.format(b, find(a, b)))
```

2.5 闰年

2.5.1 判断闰年（核心）

判断闰年相关知识：

- 四年一闰；百年不闰，四百年再闰

```
# 接收输入
a = int(input('请输入年份：'))


# 创建一个判断闰年的函数
def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                print('{}年是闰年'.format(year))     # format格式化输出   | 四百年再闰
            else:
                print('{}年不是闰年'.format(year)) # 百年不闰
        else:
            print('{}年是闰年'.format(year))       # 四年一闰
    else:
        print('{}年不是闰年'.format(year))


# 调用函数
is_leap_year(a)
```

2.5.2 判断某天是该年的第几天

①. 题目🔍

输入一个日期，打印出这一天是该年的第几天

1.1 输入样例

> 请输入日期，格式YYYY MM DD：2020 04 17

1.2 输出样例

> 2020 04 17是2020年的第108天

②. 题解✨

2.1 思路

- 首先，此题的❄核心❄在于判断闰年（判断闰年方法：四年一闰；百年不闰，四百年再闰）
- 其次，我们根据该年是否为闰年来创建两个分别适用于平年和闰年的每月天数列表
- 最后，将该月前面月份的天数与该月的天数相加即可“判断一某天是该年的第几天”

2.2 代码

```
# 模块一：接受输入的日期，切片存储
date = input('请输入日期，格式YYYY MM DD：')
year = int(date[0:4])          # 切片后存储
month = int(date[5:7])
day = int(date[8:])
# 上方代码可简化为
# year, month, day = map(int, input().split())


# 模块二：判断闰年
def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True        # 四百年再闰
            return False         # 百年不闰
        return True            # 四年一闰
    return False


# 模块三：闰年/平年每月天数列表
if is_leap_year(year) is True:
    month_list = [31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
else:
    month_list = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]


# 模块四：判断输入的月份，加上之前的天数
days = 0                  # 某月之前月份包含的天数
for i in range(month - 1):
    days += month_list[i]
total = days + day             # 总天数


# 模块五：格式化输出
print('{}是{}年的第{}天'.format(date, year, total))
```

2.6 钱💰

2.6.1 发红包

①. 题目🔍

使用`random()`模拟10次发红包的情况。

1.1 输入样例

```
请输入红包总金额：100
请输入红包总数量：8
```

1.2 输出样例

```
[11, 34, 16, 14, 2, 16, 2, 5]
[65, 26, 2, 1, 1, 2, 1, 2]
[30, 61, 3, 2, 1, 1, 1, 1]
[70, 10, 12, 1, 3, 1, 1, 2]
[14, 43, 31, 2, 7, 1, 1, 1]
[64, 11, 18, 3, 1, 1, 1, 1]
[91, 3, 1, 1, 1, 1, 1, 1]
[27, 24, 14, 2, 21, 4, 2, 6]
[64, 11, 6, 2, 14, 1, 1, 1]
[36, 19, 15, 12, 1, 8, 2, 7]
```

②. 题解✨

2.1 代码

```
import random

a = int(input('请输入红包总金额：'))
b = int(input('请输入红包总数量：'))


def dis_lucky_money(total, num):
    pack = []                                                    # 以列表
形式存储分配的红包金额
    already = 0                                                  # 已经分
配的金额

    # 前num-1次的分配情况
    for j in range(1, num):
        least_remain = num - j                                   # 后几个
红包中至少含有的总金额
```

```
        money = random.randint(1, (total - already) - least_remain)      #
randint取值范围包含首尾 <区别于range()>
        pack.append(money)
        already += money

    pack.append(total - already)                                          # 将剩余
的钱作为最后一个红包，添加至红包分配列表
    print(pack)


# 随机生成10次
for i in range(10):
    dis_lucky_money(a, b)
```

2.6.2 换硬币

①. 题目🔍

将一笔零钱💰换成5分、2分和1分的硬币，要求每种硬币至少有一枚，有几种不同的换法?

1.1 输入格式

输入在一行中给出待换的零钱数额x∈(8,100)。

1.2 输出格式

要求按5分、2分和1分硬币的数量依次从大到小的顺序，输出各种换法。
每行输出一种换法，格式为：“fen5:5分硬币数量, fen2:2分硬币数量, fen1:1分硬币数量, total:硬币总数量”。
最后一行输出“count = 换法个数”。

1.3 输入样例

```
13
```

1.4 输出样例

```
fen5:2, fen2:1, fen1:1, total:4
fen5:1, fen2:3, fen1:2, total:6
fen5:1, fen2:2, fen1:4, total:7
fen5:1, fen2:1, fen1:6, total:8
count = 4
```

②. 题解✨

2.1 思路

换钱💱时，先确定大面值的数量，再以此确定较小面值的数量

2.2 代码

```
x = int(input())

fen1 = 1
fen2 = 2
fen5 = 5
total = 0
count = 0

for i in range(x//5, 0, -1):          # '//'为整除运算
    for j in range(x//2, 0, -1):
        for k in range(x, 0, -1):
            if fen5*i + fen2*j + fen1*k == x:
                total = i + j + k
                count += 1
                print('fen5:{}, fen2:{}, fen1:{}, total:{}'.format(i, j, k,
total))

print('count =', count)
```

2.7 扑克牌🃏

2.7.1 普通发牌

①. 题目🔍

从键盘输入一个整数作为`随机种子`，随机生成一副扑克牌（去掉大小王），循环分给4位牌手，每人5张牌（第1、5、9、13、17张牌给第一个玩家，第2、6、10、14、18给第二个玩家...以此类推）并输出。

1.1 输出样例

```
4
♦Q   ♠10  ♣A   ♠9   ♠4   ♣K   ♦3   ♣Q   ♣4   ♣5   ♥8   ♠A   ♦J
♦K   ♣2   ♥K   ♥J   ♥3   ♦7   ♦2   ♣7   ♣6   ♦4   ♠Q   ♣10  ♣3
♠8   ♦A   ♥A   ♦6   ♥9   ♣J   ♥Q   ♠5   ♠J   ♦9   ♦8   ♠2   ♥4
♠6   ♦10  ♣9   ♥2   ♥5   ♥6   ♥10  ♦5   ♠K   ♣8   ♥7   ♠7   ♠3
第1个玩家的牌是: ♦Q, ♠4, ♣4, ♦J, ♥J
第2个玩家的牌是: ♠10, ♣K, ♣5, ♦K, ♥3
第3个玩家的牌是: ♣A, ♦3, ♥8, ♣2, ♦7
第4个玩家的牌是: ♠9, ♣Q, ♠A, ♥K, ♦2
```

②. 题解✨

2.1 知识储备

1. `random.seed()`：获取某种特定生成的随机数
   当`seed()`无参数时，每次生成的随机数不同，
   当`seed()`有参数时，若参数相同，则每次生成的随机数相同；若参数不同，则生成的随机数也不同

```
import random

ls = [1, 2, 3, 4, 5, 6, 7, 8]

for i in range(8):
    random.seed(4)              # 试着改变seed()中的数或者注释掉这一行，看看输出的变化
    print(random.randint(1, 10))  # randint(a, b)随机生成[a, b]间的整数

# 注意⚠️：randint与range取值范围的区别。前者能取右侧的值，后者不能
```

2. `random.shuffle()`：将一个序列（列表、字符串或元组）的顺序打乱

```
# 示例
import random

mylist = ["apple", "banana", "cherry"]
random.shuffle(mylist)

print(mylist)
```

2.2 代码

```
import random


# 生成一副不含大小王的扑克牌序列
def create(suit, d):
    ls = []
    for i in range(4):
        for j in range(13):
            temp = suit[i] + d[j]
            ls.append(temp)
    return ls


# 随机洗牌
def shufflecard(pokers):
    random.shuffle(pokers)
    return pokers
```

```
# 发5张牌给一个玩家并将发给该玩家的牌输出
def deal(pokers, n):
    ls = []
    for i in range(n - 1, n + 16, 4):
        ls.append(pokers[i])
    print('第{}个玩家的牌是：{}'.format(n, ', '.join(ls)))


n = int(input())

suit = ['♥', '♠', '♦', '♣']
d = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
random.seed(n)
poker = create(suit, d)
poker = shufflecard(poker)

for i in range(52):
    print('{:4}'.format(poker[i]), end=' ')    # 格式化输出洗牌后的牌
    if i % 13 == 12:
        print()                 # 换行
for i in range(1, 5):
    deal(poker, i)
```

2.7.2 用类发牌

①. 题目🔍

编写程序， 4名牌手打牌，计算机随机将52张牌（不含大小鬼）发给4名牌手，在屏幕上显示每位牌手的牌。

提示：

设计出3个类：Card类、Hand类和Poke类。

`Card类`代表一张牌，其中FaceNum字段指出是牌面数字1~13，Suit字段指出的是花色，值“梅”为梅花，“方”为方块，“红”为红心，“黑”为黑桃。

`Hand类`代表一手牌，可以认为是一位牌手手里的牌，其中cards列表变量存储牌手手里的牌。可以增加牌、清空手里的牌、把一张牌给别的牌手。

`Poke类`代表一副牌，可以把一副牌看作是有52张牌的牌手，所以继承Hand类。

1.1 输出样例

```
This is a module with classes for playing cards.
牌手 1:红桃6  黑桃6 梅花A 方片6 黑桃2 梅花10  红桃3 方片4 方片10  黑桃J 红桃Q 红桃10  红桃8
牌手 2:梅花J  梅花9 红桃7 红桃2 方片K 黑桃K 梅花3 方片7 黑桃Q 黑桃10  梅花Q 梅花8 黑桃7
牌手 3:梅花2  方片A 黑桃3 方片9 黑桃4 红桃K 红桃J 梅花7 红桃4 方片2 梅花4 梅花6 红桃5
牌手 4:黑桃5  红桃9 方片8 梅花5 方片J 黑桃A 梅花K 方片5 黑桃9 方片3 黑桃8 方片Q 红桃A
```

②. 题解✨

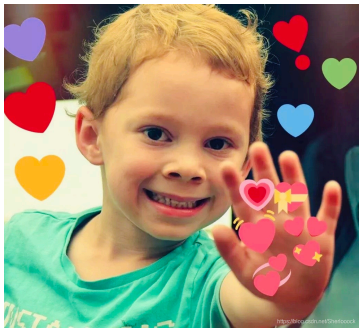
2.1 代码

```
# Card类：一张牌
class Card:
    """A playing card.card"""
    RANKS = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
    SUITS = ['梅花', '方片', '红桃', '黑桃']

    def __init__(self, rank, suit, face_up=True):
        self.rank = rank                # 牌面数字1~13
        self.suit = suit                # 花色
        self.is_face_up = face_up       # 是否显示牌的正面，True为正面，False为反面

    def __str__(self):                  # 重写print()方法，打印一张牌的信息
        if self.is_face_up:
            rep = self.suit + self.rank
        else:
            rep = 'XX'
        return rep

    def pic_order(self):                # 牌的顺序号
        if self.rank == 'A':
            FaceNum = 1
        elif self.rank == 'J':
            FaceNum = 11
        elif self.rank == 'Q':
            FaceNum = 12
        elif self.rank == 'K':
            FaceNum = 13
        else:
            FaceNum = int(self.rank)
        if self.suit == '梅花':
            Suit = 1
        elif self.suit == '方片':
            Suit = 2
        elif self.suit == '红桃':
            Suit = 3
        else:
            Suit = 4
        return (Suit - 1) * 13 + FaceNum

    def flip(self):                     # 翻牌方法
        self.is_face_up = not self.is_face_up
```

```
# Hand类：一手牌
class Hand:
    """A hand of playing cards  Hand"""

    def __init__(self):
        self.cards = []                 # cards列表变量存储牌手手里的牌

    def __str__(self):                  # 重写print()方法，打印出牌手的所有牌
        if self.cards:
            rep = ''
            for card in self.cards:
                rep += str(card) + '\t'
        else:
            rep = '无牌'
        return rep

    def clear(self):                    # 清空手里的牌
        self.cards = []

    def add(self, card):                # 增加手里的牌
        self.cards.append(card)

    def give(self, card, other_hand):   # 把一张牌给其他选手
        self.cards.remove(card)
        other_hand.add(card)
        # other_hand.append(card)       # 上面两行可以用这一行代替


# Poke类：一副牌
# 继承Hand类
class Poke(Hand):
    """Poke类代表一副牌，可以看做是有52张牌的牌手，所以继承Hand类。由于其中cards列表变量要存储52张牌
    而且要发牌，洗牌，所以增加方法如下方法："""

    def populate(self):                 # 生成一副牌
        for suit in Card.SUITS:
            for rank in Card.RANKS:
                self.add(Card(rank, suit))

    def shuffle(self):                  # 洗牌
        import random
        random.shuffle(self.cards)      # 打乱牌的顺序

    def deal(self, hands, per_hand=13):  # 将牌发给玩家，每人默认13张牌
        for rounds in range(per_hand):
            for hand in hands:
                if self.cards:
                    top_card = self.cards[0]
```

```
                    self.cards.remove(top_card)
                    hand.add(top_card)
                    # self.give(top_card,hand)  #上两句可以用此句替换
                else:
                    print('不能继续发牌了，牌已经发完了!')


if __name__ == "__main__":
    print('This is a module with classes for playing cards.')
    players = [Hand(), Hand(), Hand(), Hand()]
    poke1 = Poke()
    poke1.populate()                    # 生成一副牌
    poke1.shuffle()                     # 洗牌
    poke1.deal(players, 13)             # 发给每人13张牌
    n = 1
    for hand in players:
        print('牌手', n, end=':')
        print(hand)
        n = n + 1
```

如果这篇文章对你有所帮助的话，请点个赞👍哟~



一路看到这里，相信你的Python考试应该已经增加了几分胜算💪

如果喜欢本文请不吝点赞👍，如果爱上本文请留下评论~

如果既不想点赞又不想评论...

那么去听首我[原创的歌曲]也好鸭🥺

参考资料

- Python官方文档
- Python Cookbook

我的其它相关文章

- Linux服务器(Ubuntu 18.04)安装JDK、Hadoop、Hbase以及图形界面的全过程
- 软件测试与质量保证期末复习重点
- 嵌入式期末复习重点
- Oracle期末考试总复习资料
- 离散数学期末复习笔记【精华版】