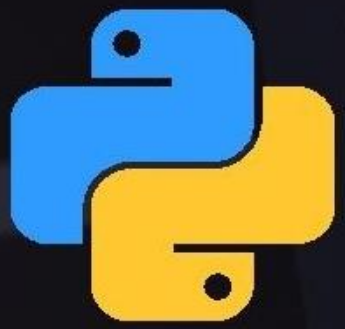
Python程序语言入门与应用



python

Life is short, use Python

人生苦短，我用Python

新乡医学院

Python程序语言入门与应用

第八章 Python 程序设计方法

王海蛟

新乡医学院

课程回顾

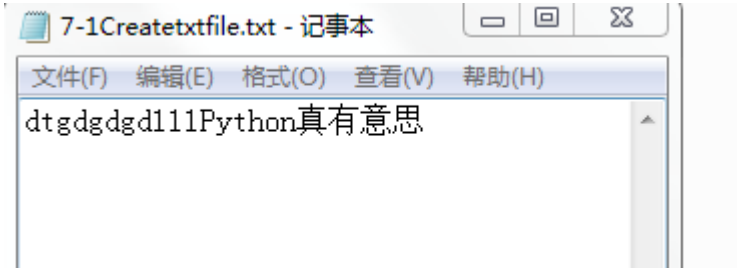
- 第7章 Python程序控制结构
  - 7.1 文件的使用
  - 7.2模块5：PIL库的使用
  - 7.3实例12：图像的字符画绘制
  - 7.4一二维数据的格式化和处理
  - 7.5实例13：CSV格式的HTML
  - 7.6高维数据的格式化
  - 7.7模块6：JSON库的使用
  - 7.8实例14：CSV和JSON格式的相互转化

上周练习题

7.1 ： 7-1Createtxtfile

使用python新建一个txt文件，并写入一句话。

```
1    str1=input('请输入您要写入文件的话：')
2    fs=open('7-1Createtxtfile.txt','at')
3    fs.write(str1)
4    fs.close()
```



上周练习题

7.27-2ReadFile

从一个建立好的txt文件中读取数据，并把数据逐行输出到控制台。

E: > 课程 > python > 第8次课 > 上周作业 > 7-2ReadFile.py >

```
1  str1=input('请输入您要打开的文件：')
2  fs=open(str1,'rt')
3  i=0
4  for line in fs:
5      i=i+1
6      print('{}行：{}'.format(i,line))
7  fs.close()
8
```

Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: E:\课程\python\第8次课\上周作业\7-2ReadFile.py ==========
请输入您要打开的文件：7-2.txt
1行：  原标题：开辟“中国之治”新境界——写在党的十九届四中全会召开之际

2行：

3行：  新华社北京10月27日电 题：开辟“中国之治”新境界——写在党的十九届四中全会召开之际

4行：

5行：  新华社记者邹伟、罗争光、刘奕湛、林晖

6行：

7行：  天下大治，古往今来无数治国者孜孜以求的理想，也是中国共产党带领中国人民矢志不渝的追寻。

8行：

9行：  10月28日至31日，中国共产党第十九届中央委员会第四次全体会议将在京召开。研究坚持和完善中国特色社会主义制度、推进国家治理体系和治理能力现代化若干重大问题，是会议的鲜明主题。

10行：

11行：  蓄积七十年磅礴之力，奋进新时代筑梦之途。

12行：

13行：  时代正见证，历史将铭记。以习近平同志为核心的党中央引领全党全国各族人民，在推动中国特色社会主义制度更加完善、国家治理体系和治理能力迈向现代化的进程中改革创新、开拓前行，不断迈向“中国之治”更高境界，必将实现党和国家事业蓬勃发展、长治久安，为人类制度文明作出新的中国贡献。

14行：

15行：  积厚以成势——我们党领导人民不断探索、不断实践，形成了具有强大生命力和巨大优越性的制度和治理体系，创造了世界罕见的经济发展奇迹和政治稳定奇迹

16行：
```

上周练习题

7-3PILIm

使用PIL库对图像进行压缩，使得图片的大小不超过10KB 。

E: > 课程 > python > 第8次课 > 上周作业 > 7-3PILIm.py > ...

```
1  from PIL import Image
2  Im=Image.open('7-3.jpeg')
3  Im.thumbnail((100,100))
4  Im.save('7-3th.jpeg')
```

新乡医学院

Python程序语言入门与应用

第八章 Python 程序设计方法

王海蛟

新乡医学院

学习目标

外文名	Bruce Eckel	主要成就	ANSI / ISO C + +标准委员会的创始成员
主要成就	MindView公司的CEO	代表作品	《Thinking in C++》
	C++标准委员会拥有表决权的成员之一	学　位	计算机工程硕士学位

学习目标

基本要求

- 掌握
  - 自顶向下的设计方法
  - 自底向上的设计过程
  - Python第三方库的安装方法
  - Python源文件的打包方法
  - python面向对象（类）
- 理解&了解
  - 计算机思维
  - 计算机生态和模块化编程思想

本课概要

- 第8章 程序设计方法论
  - 8.1 计算思维
  - 8.2实例15：体育竞技分析
  - 8.3自顶向下和自底向上
  - 8.4 模块7：pyinstaller库的使用
  - 8.5计算机生态和模块编程
  - 8.6第三方库的安装
  - 8.7 python面向对象（类）

8.1计算思维

- 8.1计算思维概述
  - 计算思维是人类科学思维活动的重要组成部分，与逻辑思维和实证思维同等重要。
  - 计算思维是人类科学思维活动的重要组成部分，人类在认识世界、改造世界过程中表现出三种基本的思维特征：
    - 以实验和验证为特征的实证思维，以物理学科为代表。
    - 以推理和演绎为特征的逻辑思维，以数学学科为代表。
    - 以设计和构造为特征的计算思维，以计算机学科为代表。
  - 计算机思维是计算机科学发展到一定程度而提出的，它是人类逐渐意识到计算机解决问题的强大能力后而产生的思维模式，具有显著的时代特性。

8.1计算思维

- 程序设计是实践计算思维的重要手段
- 抽象实际问题的计算特性，利用计算机去求解
- 计算思维的本质是抽象和自动化。
- 在程序设计范畴，计思维的主要反映在理解问题的计算特性，将计算特性抽象为计算问题，通过程序设计语言实现问题的自动求解等几个方面。

实例15：体育竞技分析

- 模拟体育竞技并进行竞技分析：

- 模拟是用来解决现实世界问题的重要手段和技术。

  计算机可以通过模拟现实世界的运行过程提供一般情况下无法获得的信息。

实例15：体育竞技分析

- 模拟体育竞技并进行竞技分析：

- 使用计算机模拟解决问题的实例包括：天气预测、飞机设计、电影特效、核试验甚至军事对抗等。如果不采用计算机模拟，这些应用则需要极其复杂的实施过程，往往代价巨大。即使很简单的模拟也可以揭示一些困难问题的本质规律。

实例15：体育竞技分析

模拟体育竞技规则：

- 从各种球类比赛中抽象一般规则，规则定义如下：
- 两个球员在一个有四面边界的场地上用球拍击球。

开始比赛时，其中一个球员首先发球。接下来球员交替击球，直到可以判定得分为止，这个过程称为回合。当一名球员未能进行一次合法击打时，回合结束。未能打中球的球员输掉这个回合。

实例15：体育竞技分析

模拟体育竞技规则：

- 如果输掉这个回合的是发球方，那么发球权交给另一方；如果输掉的是接球方，则仍然由这个回合的发球方继续发球。总之，每回合结束，由赢得该回合的一方发球。球员只能在他们自己的发球局中得分。首先达到15 分的球员赢得一局比赛。

实例15：体育竞技分析

模拟体育竞技规则：

- 在计算机模拟中，运动员的能力级别将通过发球方赢得本回合的概率来表示。因此，一个0.6 概率的球员可以在他的发球局有百分之六十的可能性赢得1 分。程序首先接收两个球员的水平值，然后通过利用这个值采用概率方法模拟多场比赛。程序最后会输出比赛运行结果。

实例15：体育竞技分析

- 该问题的IPO如下：
  - 输入：两个球员（球员A和B）的能力概率，模拟比赛的场次；
  - 处理：模拟比赛结果
  - 输出：球员A和球员B分别赢得球赛的概率

实例15：体育竞技分析

该问题的IPO如下：

- 抽象这个问题时，将球员失误、犯规等可能性一并考虑在能力概率中，在每局比赛中，球员A 先发球。

一个期望的输出结果如下

模拟比赛数量：500

球员A 获胜场次：268（53.6%）

球员B 获胜场次：232（46.4%）

实例15：体育竞技分析

该问题的IPO如下：

- 解决体育竞技分析问题似乎与之前所解决的问题有所不同，因为其处理过程并不是仅靠一个算法完成，需要稍微复杂的程序结构。
- 将结合这个例子介绍自顶向下的设计方法和自底向上的执行方法。

实例15：体育竞技分析

自顶向下的设计方法：

- 以一个总问题开始，试图把它表达为很多小问题组成的解决方案。再用同样的技术依次攻破每个小问题，最终问题变得非常小，以至于可以很容易解决。然后只需把所有的碎片组合起来，就可以得到一个程序

实例15：体育竞技分析

顶层设计：

- 自顶向下设计中最重要的是顶层设计。
- 以体育竞技分析为例，可以从问题的IPO描述开始。大多数程序都可以简单将IPO 描述直接用到程序结构设计中，体育竞技分析从用户得到模拟参数模拟比赛，最后输出结果。

实例15：体育竞技分析

- 顶层设计：
  - 步骤1：打印程序的介绍性信息
  - 步骤2：获得程序运行需要的参数：probA, probB，n；
  - 步骤3：利用球员A和B的能力值probA和prboB模拟n次比赛；
  - 步骤4：输出球员A和B获胜比赛的场次和概率。

实例15：体育竞技分析

顶层设计：

- 步骤1：输出一些介绍信息，针对提升用户体验十分有益。

```
1  def main():
2      printIntro()
```

- 顶层设计一般不写出具体代码，仅给出函数定义，其中，printIntro()函数打印一些必要的说明

实例15：体育竞技分析

顶层设计：

- 步骤2：获得用户输入。

```
1  def main():
2      printIntro()
3      probA, probB, n = getInputs()
```

- 通过函数将输入语句及输入格式等细节封装或隐藏，只需要假设程序如果调用了getInputs()函数即可获取变量 probA， probB 和 n 的值。这个函数必须为主程序返回这些值，截止第2步。

实例15：体育竞技分析

顶层设计：

- 步骤3：使用pronA和pronB获取n次比赛。

此时，可以采用解决步骤2的类似方法，设计一个simNGames()函数来模拟n场比赛，并返回结果。按照体育竞技问题的要求，该函数需要模拟比赛，并获得球员A 和球员B赢得比赛的结果。

```
1  def main():
2      printIntro()
3      probA, probB, n = getInputs()
4      winsA, winsB = simNGames(n, probA, probB)
```

实例15：体育竞技分析

顶层设计：

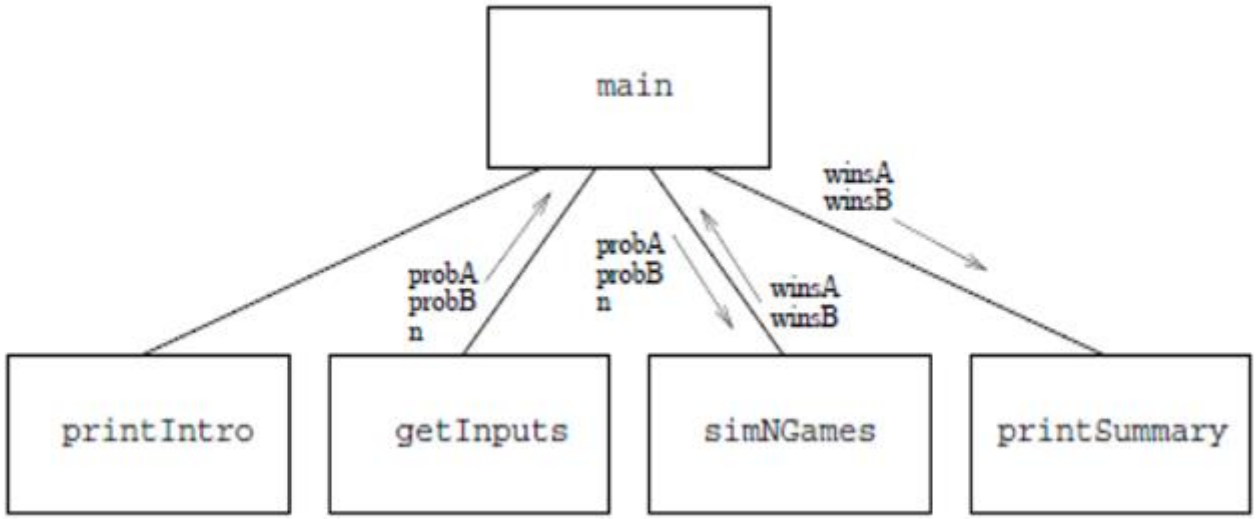
- 步骤4：输出结果，设计思想类似，仍然只规划功能和函数。

```
1  def main():
2      printIntro()
3      probA, probB, n = getInputs()
4      winsA, winsB = simNGames(n, probA, probB)
5      printSummary(winsA, winsB)
```

实例15：体育竞技分析

顶层设计：

- 原问题被划分为了4个独立的函数：printIntro()，getInputs(),simNGames()和printSummary()。分解过程让程序员在这一步不必关心具体细节而专心考虑程序的结构设计。



实例15：体育竞技分析

第n层设计：

- 每层设计中，参数和返回值如何设计是重点，其他细节部分可以暂时忽略。确定事件的重要特征而忽略其它细节过程称为抽象
- 自顶向下设计的第二阶段是实现或进一步抽象第2层函数

实例15：体育竞技分析

- 第n层设计：
  - printIntro()应该输出一个程序介绍：

```
1  def printIntro():
2      print("这个程序模拟两个选手 A 和 B 的某种竞技比赛")
3      print("程序运行需要 A 和 B 的能力值（以 0 到 1 之间的小数表示）")
```

实例15：体育竞技分析

第n层设计：

- getInputs()函数根据提示得到三个需要返回主程序的值：

```
1  def getInputs():
2      a = eval(input("请输入选手A的能力值(0-1): "))
3      b = eval(input("请输入选手B的能力值(0-1): "))
4      n = eval(input("模拟比赛的场次: "))
5      return a, b, n
```

实例15：体育竞技分析

第n层设计：

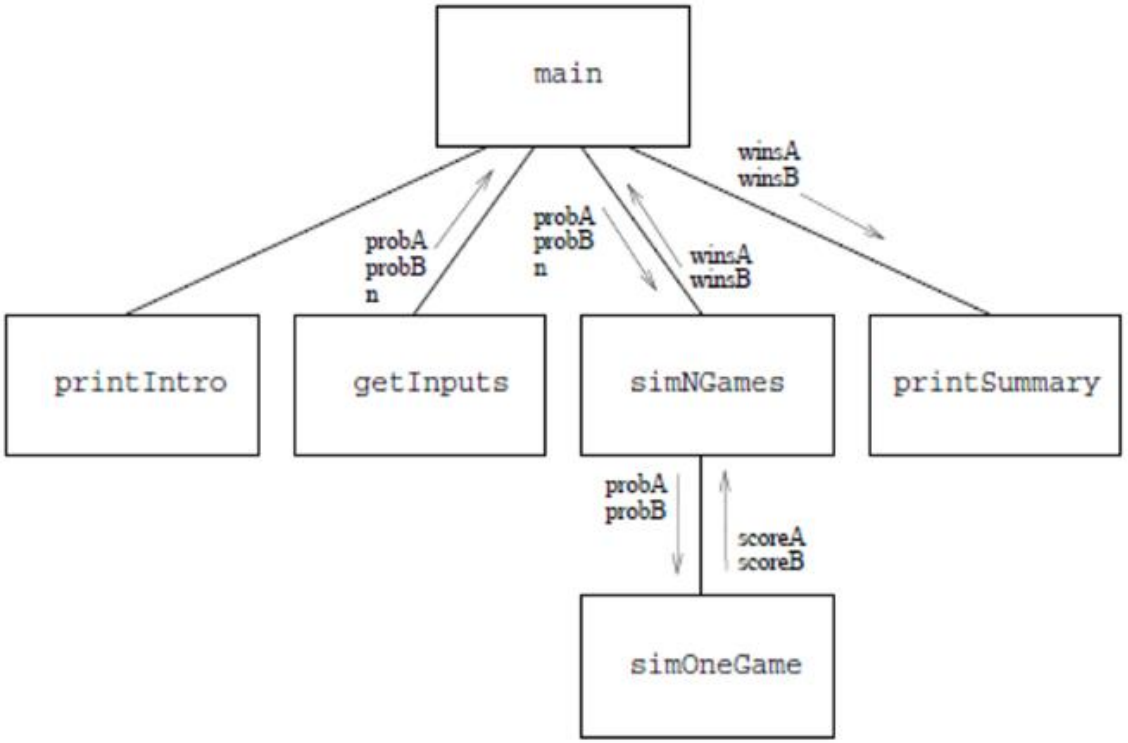
- simNGames()函数是整个程序的核心，其基本思路是模拟n场比赛，并跟踪记录每个球员赢得了多少比赛。

```
1  def simNGames(n, probA, probB):
2      winsA, winsB = 0, 0
3      for i in range(n):
4          scoreA, scoreB = simOneGame(probA, probB)
5          if scoreA > scoreB:
6              winsA += 1
7          else:
8              winsB += 1
9       return winsA, winsB
10
```

实例15：体育竞技分析

第n层设计：

- simNGames()函数是整个程序的核心，其基本思路是模拟n场比赛，并跟踪记录每个球员赢得了多少比赛。



实例15：体育竞技分析

第n层设计：

simOneGames()函数：

- 接下来需要实现simOneGame()函数。在模拟比赛的循环中，需要考虑单一的发球权和比分问题，通过随机数和概率，可以确定发球方是否赢得了比分（random() < proB）。如果球员A发球，那么需要使用A 的概率，接着根据发球结果，更新是否球员A 得分还是将球权交给球员B

实例15：体育竞技分析

第n层设计：

simOneGames()函数：

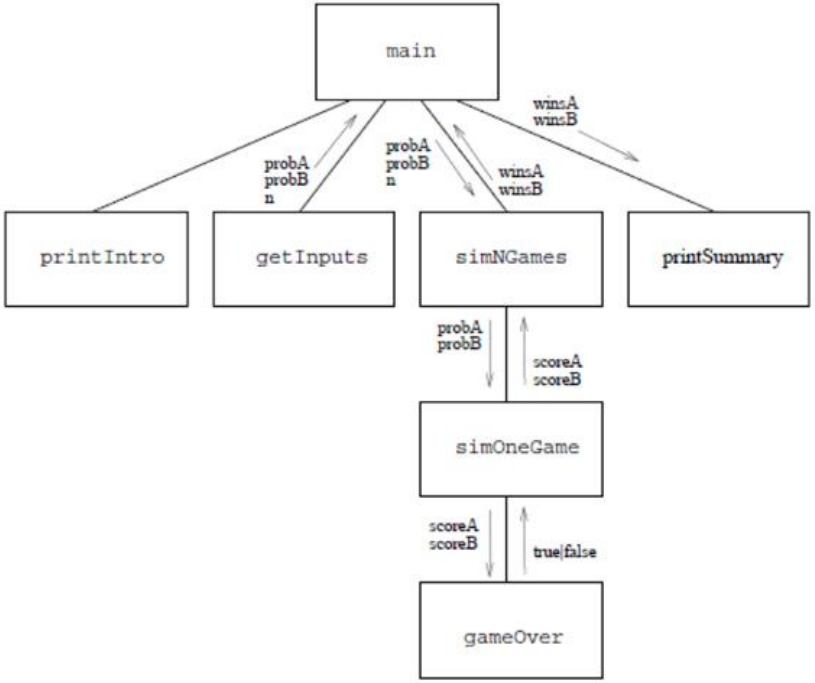
- 这里进一步设计了gameOver()函数，用来表示一场比赛结束的条件，对于不同体育比赛结束条件可能不同，封装该函数有助于简化根据不同规则修改函数的代价，提高代码可维护性。gameOver()函数跟踪分数变化并在比赛结束时返回True，未结束则返回False。然后继续循环的其余部分。

```
1  def simOneGame(probA, probB):
2      scoreA, scoreB = 0, 0
3      serving = "A"
4      while not gameOver(scoreA, scoreB):
5          if serving == "A":
6              if random() < probA:
7                  scoreA += 1
8              else:
9                  serving="B"
10         else:
11             if random() < probB:
12                 scoreB += 1
13             else:
14                 serving="A"
15     return scoreA, scoreB
```

实例15：体育竞技分析

第n层设计：

simOneGames()函数：




```
1  def simOneGame(probA, probB):
2      scoreA, scoreB = 0, 0
3      serving = "A"
4      while not gameOver(scoreA, scoreB):
5          if serving == "A":
6              if random() < probA:
7                  scoreA += 1
8              else:
9                  serving="B"
10         else:
11             if random() < probB:
12                 scoreB += 1
13             else:
14                 serving="A"
15     return scoreA, scoreB
```

实例15：体育竞技分析

第n层设计：

- 根据比赛规则，当任意一个球员分数达到15分时比赛结束。gameOver()函数实现代码如下。

```
1  def gameOver(a,b):
2      return a==15 or b==15
```

- printSummary()函数：

```
def printSummary(winsA, winsB):
1       n = winsA + winsB
2       print("竞技分析开始，共模拟{}场比赛".format(n))
3       print("选手 A 获胜{}场比赛，占比{:0.1%}".format(winsA,
4   winsA/n))
5       print("选手 B 获胜{}场比赛，占比{:0.1%}".format(winsB,
    winsB/n))
```

实例15：体育竞技分析

总代码：

```
==========
这个程序模拟两个选手A和B的某种竞技比赛
程序运行需要A和B的能力值（以0到1之间的小数表示）
请输入选手A的能力值(0-1): 0.6
请输入选手B的能力值(0-1): 0.1
模拟比赛的场次: 50
竞技分析开始，共模拟50场比赛
选手A获胜50场比赛，占比100.0%
选手B获胜0场比赛，占比0.0%
```

- 结合体育竞技实例介绍了自顶向下的设计过程。从问题输入输出确定开始，整体设计逐渐向下进行。每一层以一个大体算法描述，然后逐步细化成代码，细节被函数封装

```python
#e15.1MatchAnalysis.py
from random import random
def printIntro():
    print("这个程序模拟两个选手A和B的某种竞技比赛")
    print("程序运行需要A和B的能力值（以0到1之间的小数表示）")
def getInputs():
    a = eval(input("请输入选手A的能力值(0-1): "))
    b = eval(input("请输入选手B的能力值(0-1): "))
    n = eval(input("模拟比赛的场次: "))
    return a, b, n
def simNGames(n, probA, probB):
    winsA, winsB = 0, 0
    for i in range(n):
        scoreA, scoreB = simOneGame(probA, probB)
        if scoreA > scoreB:
            winsA += 1
        else:
            winsB += 1
    return winsA, winsB
def gameOver(a,b):
    return a==15 or b==15
def simOneGame(probA, probB):
    scoreA, scoreB = 0, 0
    serving = "A"
    while not gameOver(scoreA, scoreB):
        if serving == "A":
            if random() < probA:
                scoreA += 1
            else:
                serving="B"
        else:
            if random() < probB:
                scoreB += 1
            else:
                serving="A"
    return scoreA, scoreB
def printSummary(winsA, winsB):
    n = winsA + winsB
    print("竞技分析开始，共模拟{}场比赛".format(n))
    print("选手A获胜{}场比赛，占比{:0.1%}".format(winsA, winsA/n))
    print("选手B获胜{}场比赛，占比{:0.1%}".format(winsB, winsB/n))
def main():
    printIntro()
    probA, probB, n = getInputs()
    winsA, winsB = simNGames(n, probA, probB)
    printSummary(winsA, winsB)
main()
```

实例15：体育竞技分析

自顶向下设计过程总结：

- 步骤1：将算法表达为一系列小问题；
- 步骤2：为每个小问题设计接口；
- 步骤3：通过将算法表达为接口关联的多个小问题来细化算法；
- 步骤4：为每个小问题重复上述过程。

自顶向下过程是一种开发复杂程序最有价值的设计理念和工具，设计过程自然而且简单。自顶向下过程通过封装实现了抽象，利用了模块化的设计方法。

实例15：体育竞技分析

自底向上设计思路：

- 开展测试的更好办法也是将程序分成小部分逐个测试
- 执行中等规模程序的最好方法是从结构图最底层开始，而不是从顶部开始，然后逐步上升。或者说，先运行和测试每一个基本函数，再测试由基础函数组成的整体函数，这样有助于定位错误

实例15：体育竞技分析

测试：

- 可以从gameOver()函数开始测试。Python解释器提供import保留字辅助开展单元测试，语法格式如下：

  Import<源文件名>

- 通过输入比赛分数可以测试gameOver()函数的执行结果，初步分析gameOver() 函数是正确的。

```
>>>import e151MatchAnalysis
>>>e151MatchAnalysis.gameOver(15, 10)
True
>>>e151MatchAnalysis.gameOver(10, 1)
False
```

实例15：体育竞技分析

测试：

- 可以进一步测试simOneGame()函数：

```
>>>import e151MatchAnalysis
>>>e151MatchAnalysis.simOneGame(.45, .5)
(9, 15)
>>>e151MatchAnalysis.simOneGame(.45, .5)
(15, 13)
```

当概率相等时，比分也十分接近。但当概率相差很远时，比赛呈现压倒性优势。这和概率的预期也相符合。

实例15：体育竞技分析

测试：

- 通过继续进行这样的单元测试可以检测程序中的每个函数。独立检验每个函数更容易发现错误。通过模块化设计可以分解问题使编写复杂程序成为可能，通过单元测试方法分解问题使运行和调试复杂程序成为可能。自顶向下和自底向上贯穿程序设计和执行的整个过程。

实例15：体育竞技分析

拓展：软件开发模型

- 软件开发模型是指软件开发全部过程、活动和任务的结构框架。软件开发包括需求、设计、编码和测试等阶段，有时也包括维护阶段。软件开发模型能清晰、直观地表达软件开发全过程，明确规定了要完成软件的主要活动和任务，用来作为软件项目工作的基础。对于不同的软件系统，可以采用不同的开发方法、使用不同的编程语言、组织不同技能的人员、运用不同的管理方法等。

模块7：pyinstaller库的使用

概述

- Pyinstaller是将Python语言脚本（.py文件）打包成Windows、Linux、Mac OS X等操作系统下可执行文件的第三方库。
- Pyinstaller是一个十分有用的第三方库，它能够在Windows、Linux、Mac OS X等操作系统下将Python源文件打包，通过对源文件打包，Python程序可以在没有安装Python的环境中运行，也可以作为一个独立文件方便传递和管理。

模块7：pyinstaller库的使用

Pyinstaller安装：

- Pyinstaller在命令行下使用pip安装：
  - :\>pip install pyinstaller 或者
  - :\>pip3 install pyinstaller
- Pyinstaller的官方网址:http://www.pyinstaller.org。

模块7：pyinstaller库的使用

Pyinstaller的使用：

- PyInstaller库会自动将pyinstall 命令安装到Python解释器目录中，与pip 或pip3命令路径相同，因此可以直接使用。
- 使用PyInstaller 库十分简单，在Windows 平台的命令行中输入Python 源文件名称，可以使用相对路径或绝对路径

模块7：pyinstaller库的使用

- Pyinstaller的使用：

  - :\>pyinstaller dpython.py
  - 或
  - :\>pyinstaller D:\codes\dpython.py

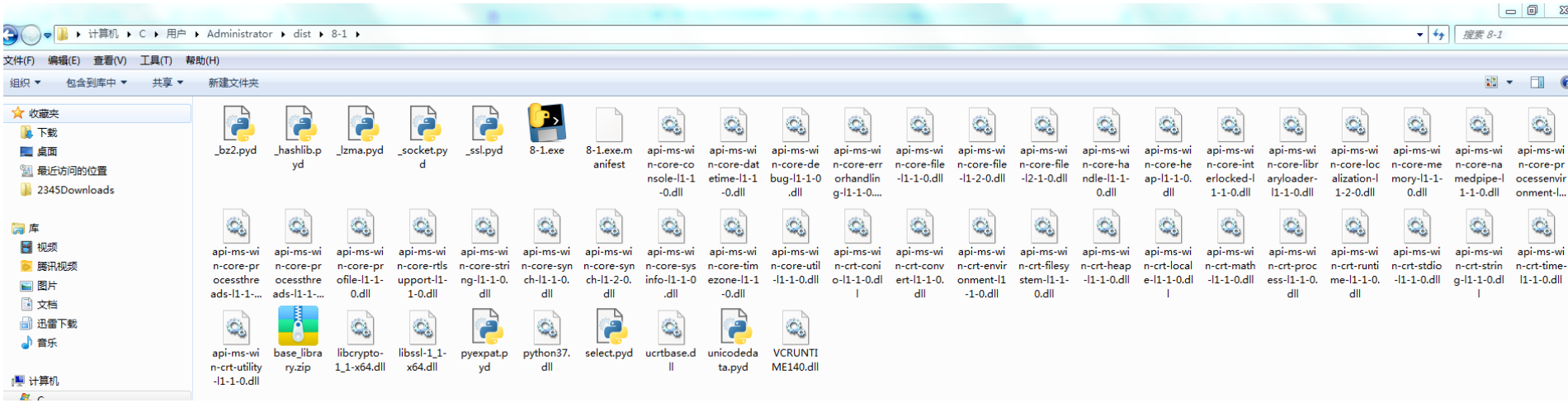
模块7：pyinstaller库的使用

Pyinstaller的使用：

- 执行完毕后，源文件所在目录将生成dist和build 两个文件夹。其中，build 目录是pyinstaller存储临时文件的目录，可以安全删除。最终的打包程序在dist内部的dpython目录中。目录中其他文件是可执行文件dpython.exe 的动态链接库

模块7：pyinstaller库的使用

- Pyinstaller的使用：



模块7：pyinstaller库的使用

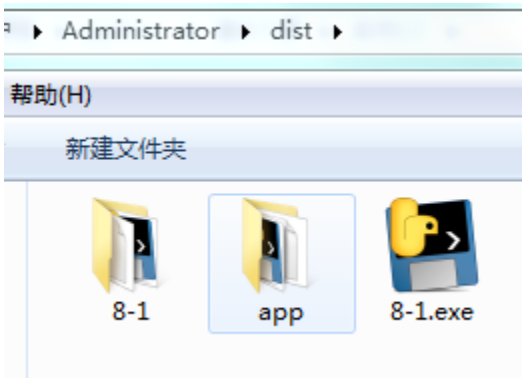
Pyinstaller的使用：

- 可以通过-F 参数对Python 源文件生成一个独立的可执行文件，如下：
- **:\>pyinstaller -F dpython.py**
- 执行后在dist 目录中出现了dpython.exe 文件，没有任何依赖库，执行它即可。

模块7：pyinstaller库的使用

Pyinstaller的使用：

```
C:\Users\Administrator>pyinstaller -F E:\课程\python\第8次课\实例15\8-1.py
66 INFO: PyInstaller: 3.5
66 INFO: Python: 3.7.3
66 INFO: Platform: Windows-7-6.1.7601-SP1
67 INFO: wrote C:\Users\Administrator\8-1.spec
```



模块7：pyinstaller库的使用

Pyinstaller的使用的注意事项：

- 使用PyInstaller 库需要注意以下问题：
  - 文件路径中不能出现空格和英文句号（.）；
  - 源文件必须是UTF-8 编码，暂不支持其他编码类型。采用IDLE 编写的源文
  - 件都保存为UTF-8 编码形式，可直接使用。

模块7：pyinstaller库的使用

Pyinstaller的使用：

- 可以通过-F 参数对Python 源文件生成一个独立的可执行文件，如下：
- **:\>pyinstaller -F dpython.py**
- 执行后在dist 目录中出现了dpython.exe 文件，没有任何依赖库，执行它即可。

模块7：pyinstaller库的使用

Pyinstaller命令常用参数：

参数	功能
-h, --help	查看帮助
-v, --version	查看 PyInstaller 版本
--clean	清理打包过程中的临时文件
-D, --onedir	默认值，生成 dist 目录
-F, --onefile	在 dist 文件夹中只生成独立的打包文件
-p *DIR*, --paths *DIR*	添加 Python 文件使用的第三方库路径
-i <*.ico or .exe,ID or .icns*>, --icon <*.ico or .exe,ID or .icns* >	指定打包程序使用的图标（icon）文件

模块7：pyinstaller库的使用

动态链接库：

- 动态链接提供了一种方法，能够使进程在运行时实际调用不属于其程序的代码。如果其他代码由操作系统提供，则应用程序由于不包含这些代码而变得十分精简。Windows 平台提供大量的动态链接库，一般使用dll 或ocx 为扩展名。静态链接与动态链接相对，指程序中自包含其所调用的所有代码，这使程序可以在系统间移动而无需考虑库函数是否一致

8.5计算机生态和模块化编程

- 概述：
  - Python语言有9万多个第三方库，形成庞大的计算机生态，模块化编程思想是python语言最大的价值。
  - 近20年的开源运动产生了深植于各信息技术领域的大量可重用资源，直接且有力的支撑了信息技术超越其他技术领域的发展速度。形成了“计算生态”，产生了业界广泛利用可重用资源快速构建应用已经是主流产品开发模式。
  - Python从诞生之初就致力于开源，建立了全球最大的编程计算生态。
    - https://pypi.python.org/pypi

8.5计算机生态和模块化编程

胶水语言

- 由于Python 有非常简单灵活的编程方式，很多采用C、C++等语言编写的专业库可以经过简单的接口封装供Python 语言程序调用。这样的粘性功能使得Python 语言成为了各类编程语言之间的接口，俗称Python 语言为“胶水语言”。

8.5计算机生态和模块化编程

计算生态

- 30 年前，编写程序仅能调用官方提供的API功能。
- 20 年前，开源运动的兴起和蓬勃发展，一批开源项目诞生
- 10 年前，开源运动深入开展，专业人士开始大量贡献各领域最优秀的研究和开发成果，并通过开源库形式发布出来。
- 那今天呢？编程领域形成了庞大的计算生态，需要一种编程语言或方式能够将不同语言、不同特点、不同使用方式的代码统一起来。

第三方库

- Python第三方程序包括库（library）、模块（module）、类（class）和程序包（Package）等多种命名
- 统一将这些可重用代码统称为“库”。

第三方库

- Python 内置的库称为标准库，其他库称为第三方库
- 在计算生态思想指导下，编写程序的起点不再是探究每个具体算法的逻辑功能和设计，而是尽可能利用第三方库进行代码复用，探究运用库的系统方法。

8.5计算机生态和模块化编程

模块化编程

- 这种像搭积木一样的编程方式，称为“模块编程”
- 每个模块可能是标准库、第三方库、用户编写的其他程序或对程序运行有帮助的资源等。

8.5计算机生态和模块化编程

模块化编程

- 模块编程与模块化编程不同，模块化编程主张采用自顶向下设计思想，主要开展耦合度低的单一程序设计与开发，而模块编程主张利用开源代码和第三方库作为程序的部分或全部模块，搭积木一样编写程序。
- 10 余个各类函数库 + 20 余个实例

8.5计算机生态和模块化编程

模块化编程

- 因此，需要专业程序员经过漫长学习才能够掌握并开发有价值程序。Python 语言却不相同，它不是其他语言的替代，而是一个真正面向计算生态的语言。
- ——AlphaGo 很炫，它打败了世界上最厉害的人类围棋选手。
- ——AlphaGo 开源了，采用Python 语言，快去用用看看吧。

8.6 python第三方库的安装

PIP安装工具

- pip 是Python 内置命令，需要通过命令行执行，执行pip -h 命令将列出pip 常用的子命令，注意，不要在IDLE 环境下运行pip 程序。
- :\>pip -h

8.6 python第三方库的安装

PIP安装工具

```
Commands:
  install      Install packages.
  download     Download packages.
  uninstall    Uninstall packages.
  freeze       Output installed packages in requirements format.
  list         List installed packages.
  show         Show information about installed packages.
  search       Search PyPI for packages.
  wheel        Build wheels from your requirements.
  hash         Compute hashes of package archives.
  completion   A helper command used for command completion
  help         Show help for commands.
```

8.6 python第三方库的安装

- PIP安装常用的第三方库：

库名	用途	pip 安装指令
NumPy	矩阵运算	pip install numpy
Matplotlib	产品级 2D 图形绘制	pip install matplotlib
PIL	图像处理	pip install pillow
sklearn	机器学习和数据挖掘	pip install sklearn
Requests	HTTP 协议访问	pip install requests
Jieba	中文分词	pip install jieba
Beautiful Soup 或 bs4	HTML 和 XML 解析	pip install beautifulsoup4

8.6 python第三方库的安装

- PIP安装常用的第三方库：

Wheel	Python 文件打包	pip install wheel
PyInstaller	打包 python 源文件为可执行文件	pip install pyinstaller
Django	Python 最流行的 Web 开发框架	pip install django
Flask	轻量级 Web 开发框架	pip install flask
WeRoBot	微信机器人开发框架	pip install werobot
Networkx	复杂网络和图结构的建模和分析	pip install networkx
SymPy	数学符号计算	pip install sympy
pandas	高效数据分析	pip install pandas

8.6 python第三方库的安装

- PIP安装常用的第三方库：

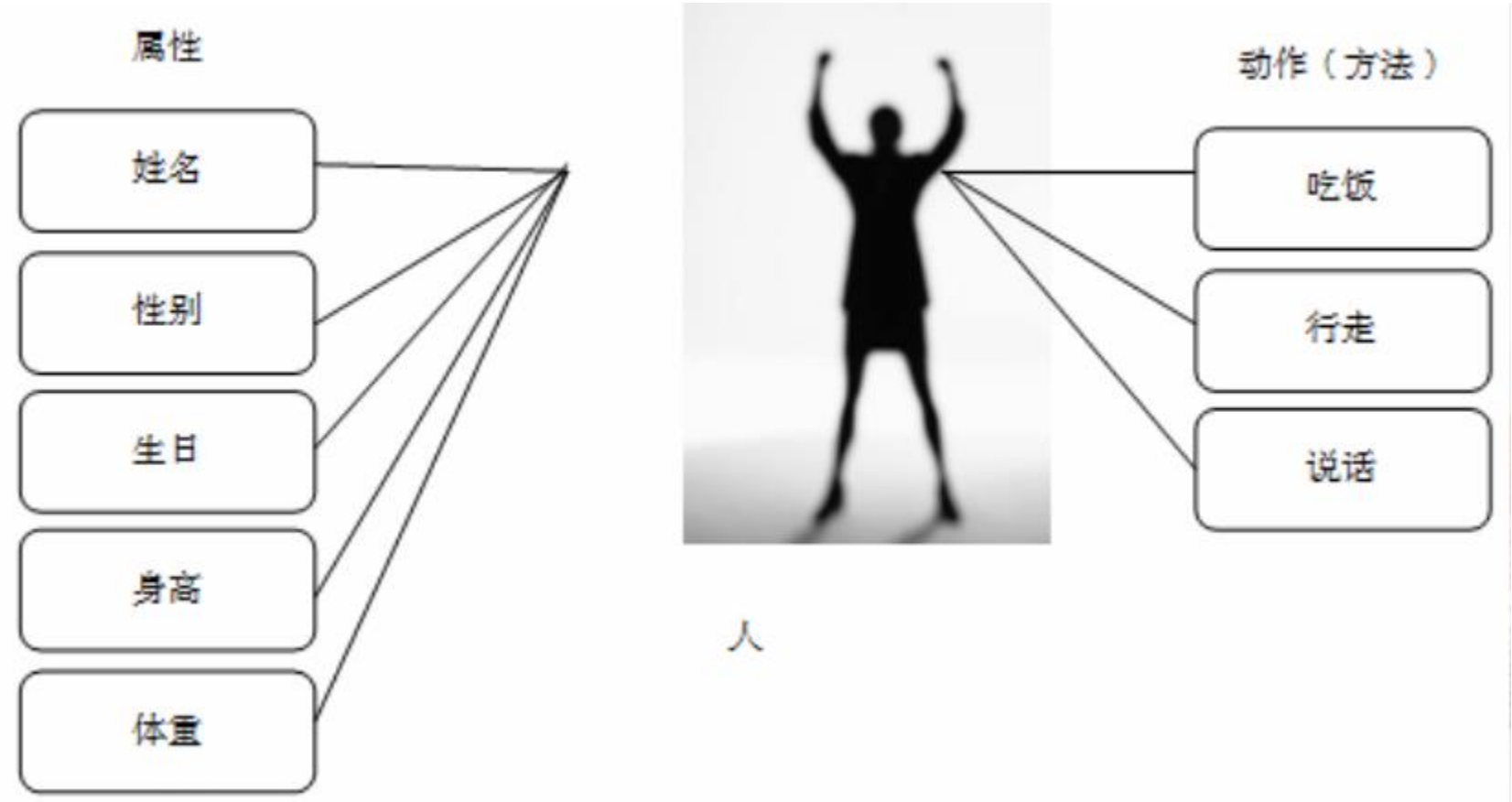
PyQt5	基于 Qt 的专业级 GUI 开发框架	pip install pyqt5
PyOpenGL	多平台 OpenGL 开发接口	pip install pyopengl
PyPDF2	PDF 文件内容提取及处理	pip install pypdf2
docopt	Python 命令行解析	pip install docopt
PyGame	简单小游戏开发框架	pip install pygame

8.7 python面向对象（类）

- 面向对象（Object Oriented Programming, OOP）是python采用的基本编程思想，它可以将属性和代码集成在一起，定义为类，从而使得程序设计更加简单、规范、有条理。
- OPP是以一种程序设计思想。OPP把对象作为程序的基本单元，一个对象包含了数据和操作数据的函数。
- 在Python中，所有的数据类型都可以视为对象，当然也可以自定义对象。自定义的对象数据类型就是面向对象中类的概念。
- 当一个类定义完之后，就产生了一个类对象。类对象支持两种操作：引用和实例化。引用是通过类对象去调用类中的属性或方法，而实例化则是产出一个类的具体实例，称作实例对象。

8.7 python面向对象（类）

- 类是抽象出来某种事物的集合。



8.7 python面向对象（类）

- 类的定义：
  - 使用class关键词定义类，语法如下：

```
Class <类名>:
    语句
```

  - Python中规定类名首字母大写。

8.7 python面向对象（类）

- 类的定义：

```
class People:
    '这是一个类'
    sex='男'
    def name(self,str):
        self.name=str
        print('实例的名字是：'+self.name)
        return None
    def __init__ (self,birthday,height,weight):
        self.birthday=birthday
        self.height=height
        self.weight=weight
    def talk(self):
        print(self.name+'在说话')
        return None
p1=People(1120,178,130)
print(p1.name('刘明'))
print(p1.sex)
print(p1.talk())
print(p1.weight)
```

```
=====
实例的名字是：刘明
None
男
刘明在说话
None
130
>>>
```

```
<class '__main__.People'>
```

```
>>> ls=(1,2)
>>> type(ls)
<class 'tuple'>
>>>
```

8.7 python面向对象（类）

类的继承

class ClassName1（ClassName2）：

```
File  Edit  Format  Run  Options  Window  Help
class People:
    '这是一个类'
    sex='男'
    def name(self,str):
        self.name=str
        print('实例的名字是：'+self.name)
        return None
    def __init__ (self,birthday,height,weight):
        self.birthday=birthday
        self.height=height
        self.weight=weight
    def talk(self):
        print(self.name+'在说话')
        return None
class Woman(People):
    sex='女'
p1=Woman(1120,178,130)
print(p1.name('刘明'))
print(p1.sex)
print(p1.talk())
print(p1.weight)
print(type(p1))
```

```
Python 3.7.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: E:\课程\python\第8次课\实例15\class1-woman.py ==========
====
实例的名字是：刘明
None
女
刘明在说话
None
130
<class '__main__.Woman'>
>>>
```

下周课程&课后作业

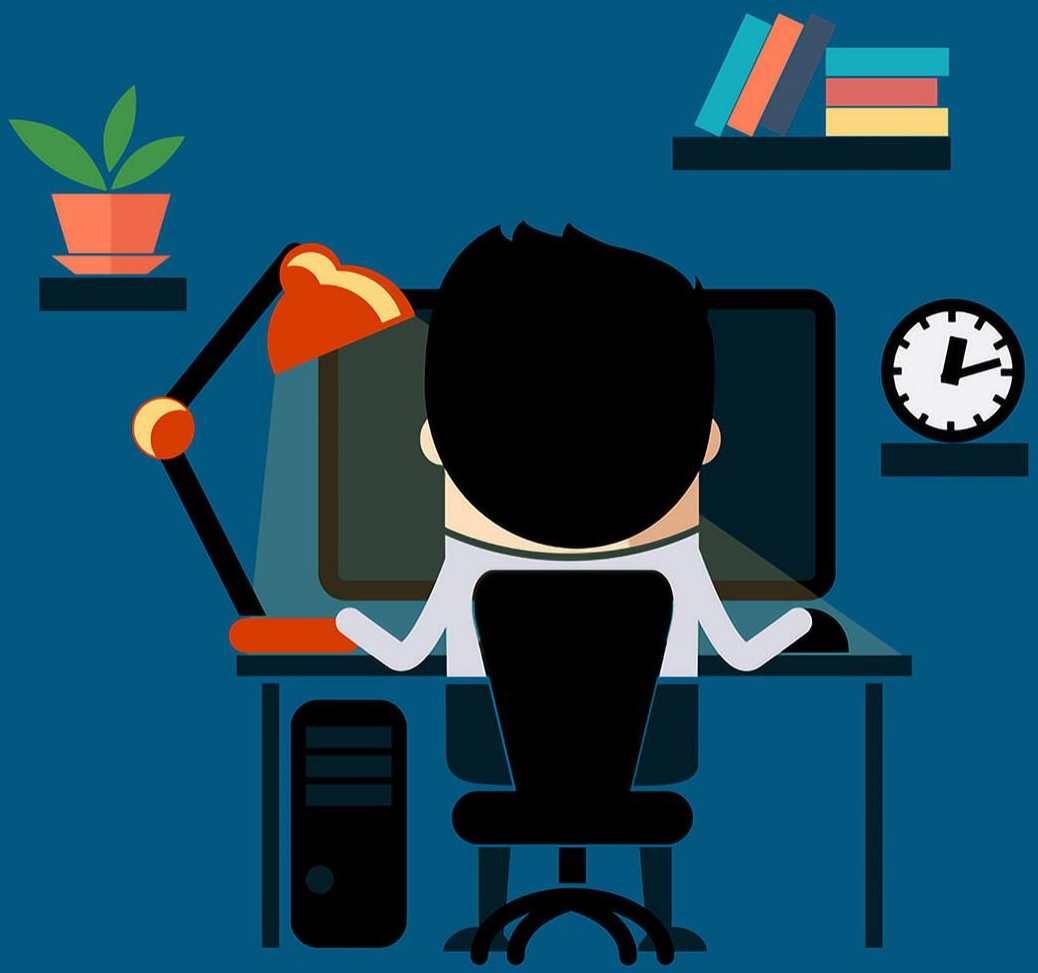
- 第七章 文件和数据格式化
- 课后练习
  - 借用实例15，采用乒乓球规则，分析体育竞技规律。

    代码文件命名：8-1pangbangqiu
  - 定义一个类并实例化。

    代码文件命名：8-2Cls

.py代码文件打包(8.学号+姓名)发送到python_xxmu@163.com

编程辣么好，还等什么？开始学习吧！



Programing is an Art