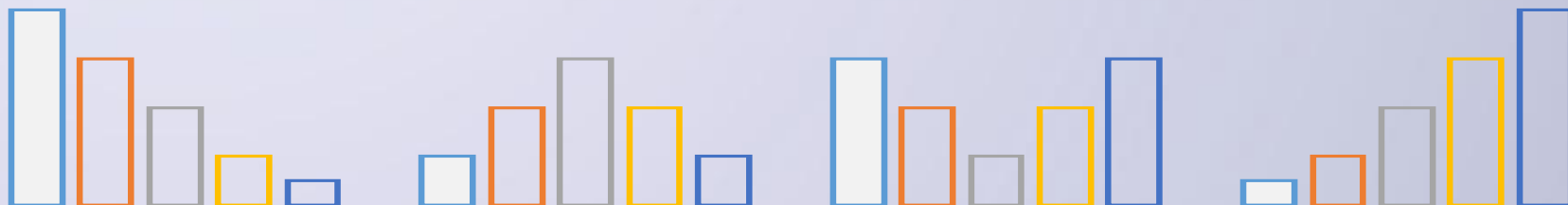
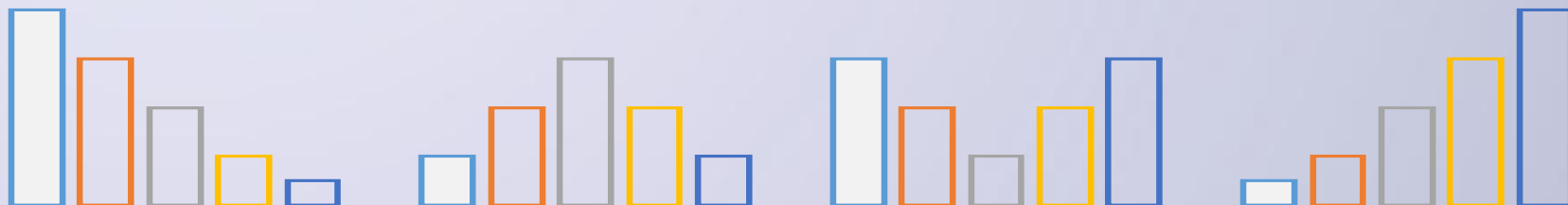


Python语言程序设计

北京理工大学 嵩天



第8章 程序设计方法论





人生苦短，请用 *Python*。

Life is short. You need Python.

——布鲁斯·埃克尔(Bruce Eckel)

ANSI/ISO C++ 标准委员会发起者之一



学习目标

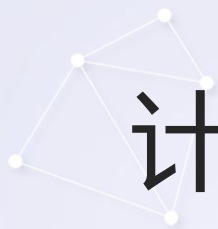
- 1. 了解计算思维的概念；
- 2. 掌握自顶向下的设计方法；
- 3. 掌握自底向上的执行过程；
- 4. 了解计算生态和模块编程思想；
- 5. 掌握Python 第三方库的安装方法；
- 6. 掌握Python 源文件的打包方法。



计算思维

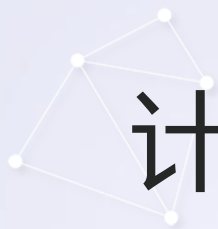


- 计算思维是人类科学思维活动的重要组成部分，与逻辑思维 and 实证思维同等重要。



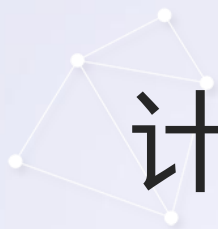
计算思维

- 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。



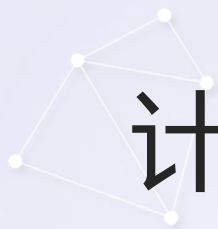
计算思维

- 计算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自然产生的思维模式，具有显著的时代特性。



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（ Abstraction ）和自动化（ Automation ）。



计算思维

- 在程序设计范畴，计算思维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。



拓展：ENIAC

- ENIAC (Electronic Numerical Integrator And Computer , 电子数字积分计算机) 是世界上第一台通用计算机 , 于1946 年诞生于宾夕法尼亚大学摩尔实验室。尽管 ENIAC 占地面积约170 平方米 , 重达30 吨 , 它比当时最快的计算设备速度高1000倍。这样惊人的计算性能在随后半个多世纪里改变了整个世界 , 原始创新极其重要。



实例15：体育竞技分析



- 这是一个模拟体育竞技并进行竞技分析的实例。



模拟分析

- 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。



模拟分析

- 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。



拓展：模拟和仿真

- 模拟（simulation）是抽象原系统某些行为特征并用另一系统来表示这些特征的过程，通常用于设计初期的模型验证。



拓展：模拟和仿真

- 仿真（emulation）则更进一步，需要模仿系统真实能做的事情，接收同样的数据，获得同样的结果，只不过实现的“过程”不同。仿真一般用于处理兼容性问题或在资源有限条件下实现系统原型。



实例15：体育竞技分析 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 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

- 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 模式如下：
 - 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：模拟比赛过程；
 - 输出：球员A 和B 分别赢得球赛的概率



实例15：体育竞技分析

- 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

- 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。



自顶向下设计



- 程序需要采用自顶向下设计方法，采用自底向上的执行过程。



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



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。



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 probB ，模拟 n 场比赛；
- 步骤4：输出球员A 和B 获胜比赛的场次及概率。



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的代码，仅给出函数定义，其中，`printIntro()`函数打印一些必要的说明



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。



顶层设计

- **步骤3** 需要使用probA、probB 模拟n场比赛。
- 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)
```



顶层设计

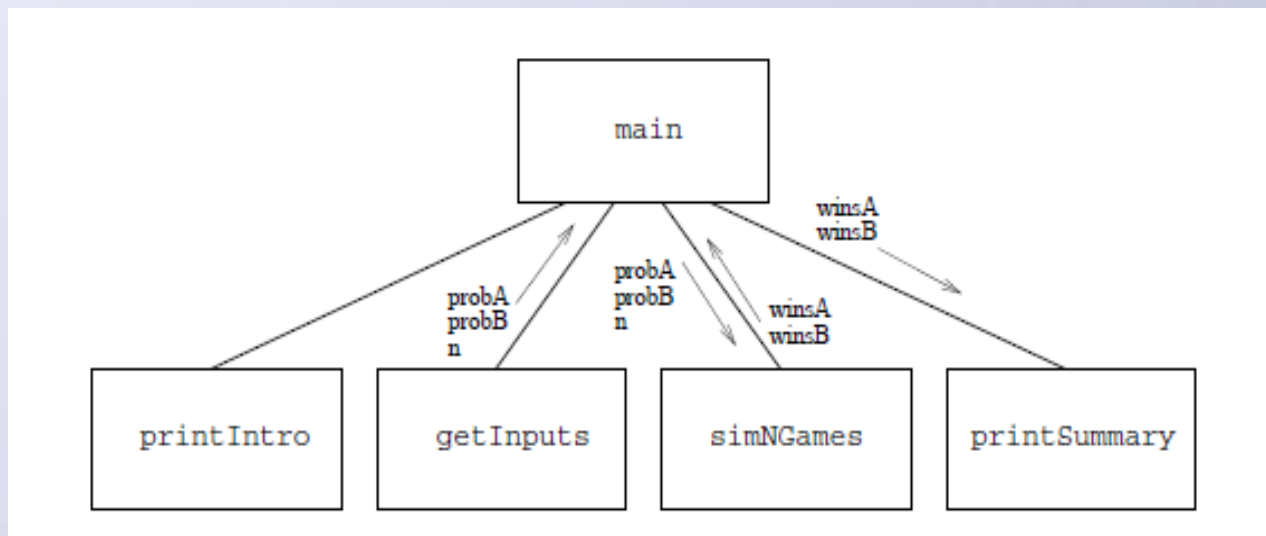
- **步骤4** 输出结果，设计思想类似，仍然只规划功能和函数

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




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





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



第 n 层设计

- printIntro()函数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）")
```



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
```



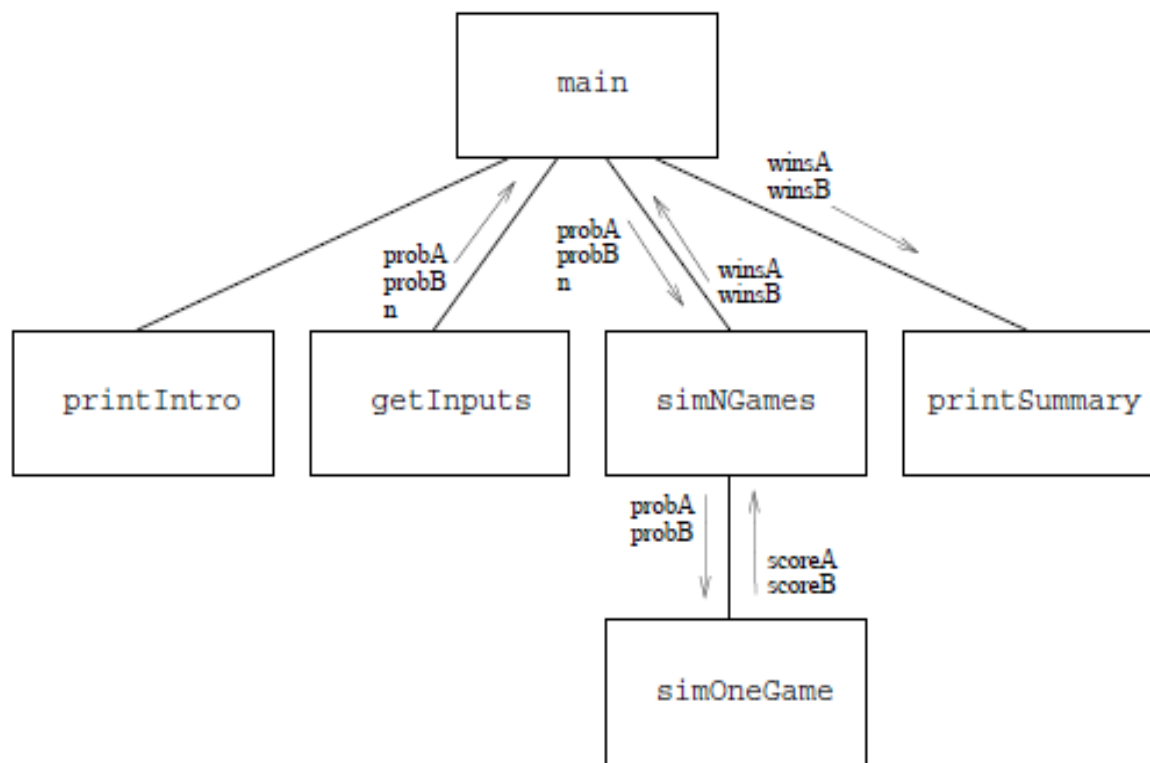
第 n 层设计

- `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
```



第 n 层设计





第 n 层设计

- 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（ $\text{random}() < \text{proB}$ ）。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



第 n 层设计

```
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
```

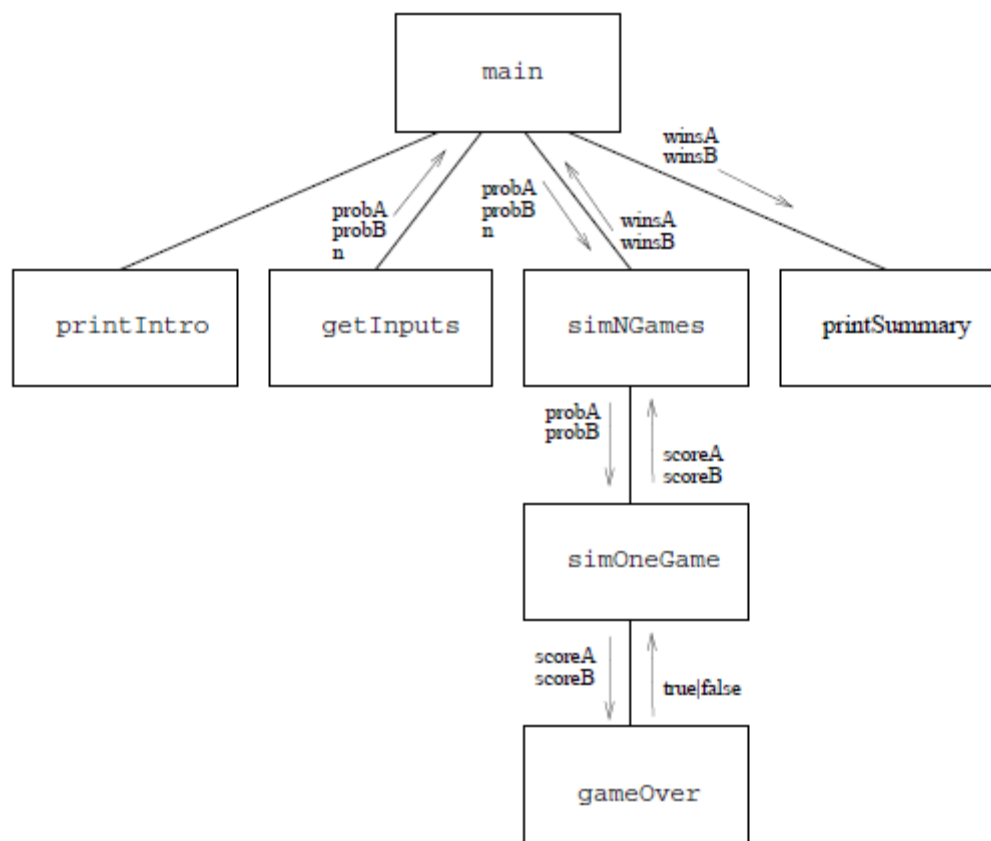



第 n 层设计

- 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。



第 n 层设计





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
```



第 n 层设计

■ 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 {:.1%}".format(winsA,  
4 winsA/n))  
5     print("选手 B 获胜 {} 场比赛, 占比 {:.1%}".format(winsB,  
winsB/n))
```



第 n 层设计

■ 输出结果

```
>>>
```

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) : **0.45**

请输入选手 B 的能力值 (0-1) : **0.5**

模拟比赛的场次: **1000**

竞技分析开始，共模拟 1000 场比赛

选手 A 获胜 371 场比赛，占比 37.1%

选手 B 获胜 629 场比赛，占比 62.9%



第 n 层设计

- 最后再回到体育竞技分析问题，通过模拟方法分析球员之间能力的微小差异带来的比赛结果不同，是否会产生能力差别小却导致比赛结果一边倒的现象？



设计过程总结

- 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



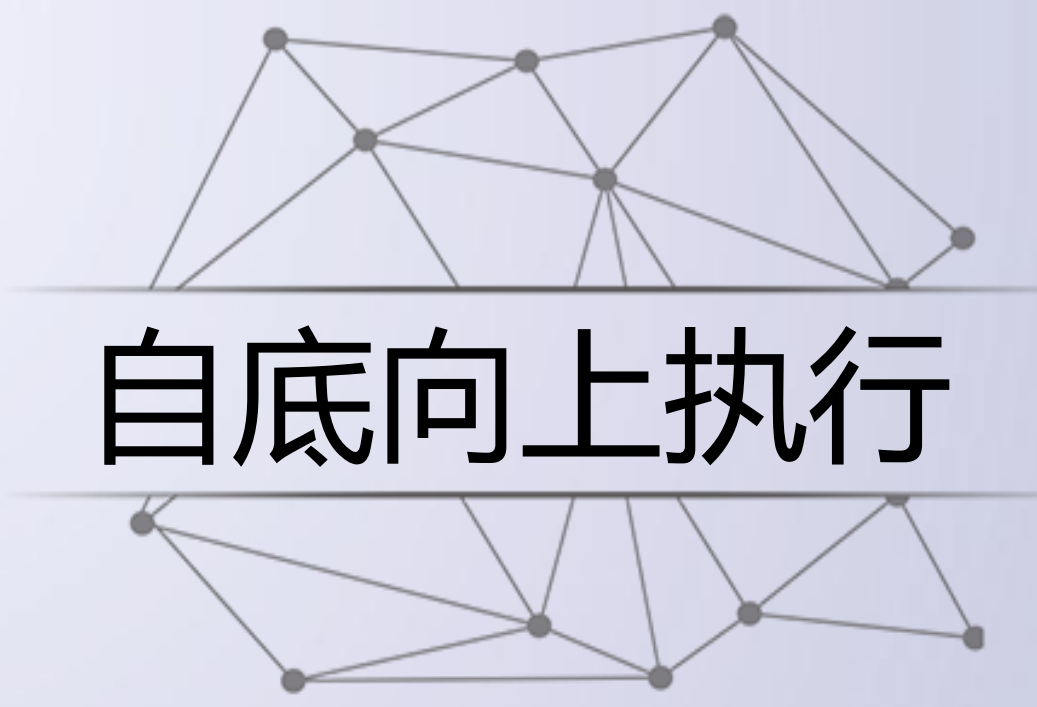
设计过程总结

- 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。



自顶向下设计过程

- 自顶向下设计是一种开发复杂程序最具价值的设计理念 and 工具，设计过程自然且简单，自顶向下设计通过封装实现抽象，利用了模块化方法的思想



自底向上执行



自底向上执行

- 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



- 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 <源文件名称>



测试

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

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



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)
```

- 注意到当概率相等时，比分也十分接近。当概率相差很远时，比赛则成压倒性趋势。这与对函数的预期是相符合的。



- 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。



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



- 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概述

- 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。



Pyinstaller 安装

- PyInstaller 需要在命令行（控制台）下用pip 工具安装：

- **:\>pip install pyinstaller 或者**

- **:\>pip3 install pyinstaller**

- PyInstaller 的官方网站网址为

<http://www.pyinstaller.org/>



PyInstaller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



PyInstaller使用

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



PyInstaller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



PyInstaller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。



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。



拓展：动态链接

- 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。



pyinstaller 命令的常用参数

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 <i>DIR</i> , --paths <i>DIR</i>	添加 Python 文件使用的第三方库路径
-i <i><.ico or .exe,ID or .icns></i> , icon <i><.ico or .exe,ID or .icns ></i> --	指定打包程序使用的图标 (icon) 文件



pyinstaller 解析

- `pyinstall` 命令不需要在Python 源文件中增加代码，只需要通过命令行进行打包即可。`-F` 参数最为常用，对于包含第三方库的源文件，可以使用`-p` 添加第三方库所在路径。如果第三方库由`pip` 安装且在Python 环境目录中，则不需要使用`-p` 参数。



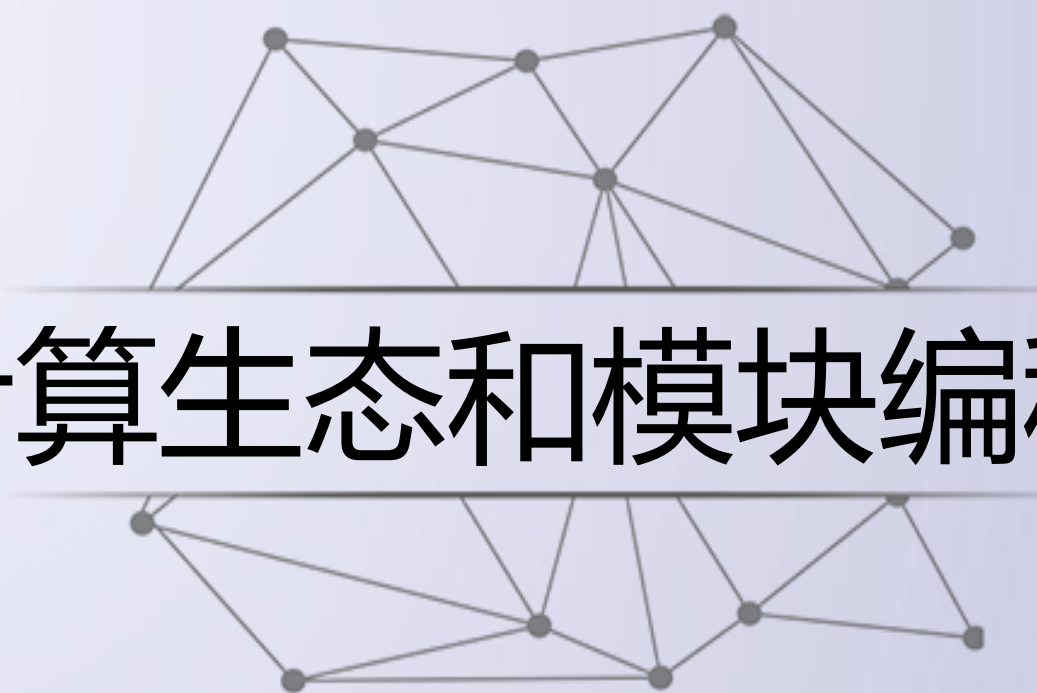
pyinstaller 解析

- 以实例10 的实例代码10.3 为例，该代码使用了jieba 库，将该文件改名为caltk.py，打包方法如下：
：
- **`:\>pyinstaller -F D:\codes\caltk.py`**



pyinstaller 解析

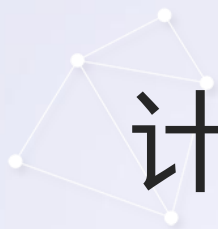
- 在dist 目录中将生成打包文件caltk.exe , 将三国演义.txt 文件拷贝到dist 目录中 , 执行该程序 :
- **:\ >D:\codes\dist\caltk.exe**



计算生态和模块编程



- Python 语言有9万多个第三方库，形成了庞大的计算生态，模块编程思想是Python 语言最大的价值



计算生态

- 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方式。



Python的计算生态

- Python 语言从诞生之初致力于开源开放，建立了全球最大的编程计算生态。
- Python 官方网站提供了第三方库索引功能
- <https://pypi.python.org/pypi>



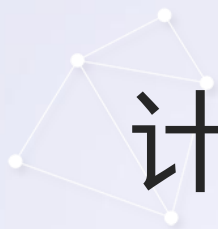
Python的计算生态

- 该页面列出了Python 语言9 万多个第三方库的基本信息，这些函数库覆盖信息领域技术所有技术方向。
这里需要说明，Python 语言的函数库并非都采用Python 语言编写。



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”。



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 第三方的程序包括库（ 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。



模块编程

- 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。



模块编程与模块化编程

- 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



拓展：Python 构建计算生态的 编程语言

- ——学了编程能做什么？
- ——你将走进信息时代的大门，能够运用信息时代的最新成果。
- 学了编程能做什么？似乎只能打印字符、挑战汉诺塔。这是大多数编程语言学习预期和学习效果的鸿沟。因为绝大多数编程语言设计用于开发专业功能，而不是用于构建计算生态

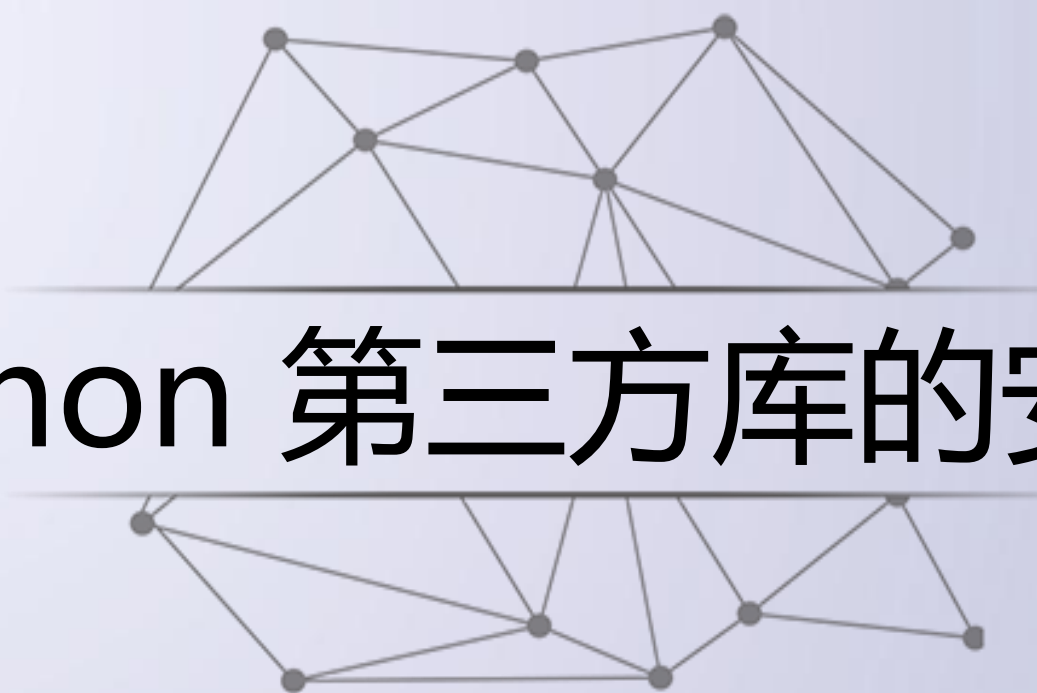


拓展：Python 构建计算生态的 编程语言

- 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。



Python 第三方库的安装




第三方库的安装

- 第三方库需要安装后才能使用。由于Python 语言经历了版本更迭过程，而且，第三方库由全球开发者分布式维护，缺少统一的集中管理，因此，Python的第三方库曾经一度制约了Python 语言的普及和发展。随着官方pip 工具的应用，Python 第三方库的安装变得十分容易。




第三方库的安装

- Python 第三方库依照安装方式灵活性和难易程度有三个方法，建议读者依次使用，能够将第三方库安装成功，
- 这三个方法是：pip 工具安装、自定义安装和文件安装。



pip 工具安装

- 最常用且最高效的Python第三方库安装方式是采用pip工具安装。
- pip是Python官方提供并维护的在线第三方库安装工具。
- 对于同时安装Python 2 和Python 3 环境的系统，建议采用pip3 命令专门为Python 3 版本安装第三方库。



pip 工具安装

- 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**



pip 工具安装

Commands:

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



pip 使用

- pip 支持安装 (install)、下载 (download)、卸载 (uninstall)、列表 (list)、查看 (list)、查找 (search) 等一系列安装和维护子命令。
- 安装一个库的命令格式如下，例如：

```
pip install <拟安装库名>
```



pip 使用

- 例如，安装pygame 库，pip 工具默认从网络上下载pygame 库安装文件并自动安装到系统中。

```
: \> pip install pygame  
  
...  
  
Installing collected packages: pygame  
Successfully installed pygame-1.9.2b1
```



pip 使用

- 使用-U 标签可以更新已安装库的版本，例如，用

pip 更新本身：

```
:\>pip install -U pip
```

```
Requirement already up-to-date: pip in d:\python35-32\lib\site-packages
```



pip 使用

- 卸载一个库的命令格式如下，例如：

`pip uninstall <拟卸载库名>`

```
:>pip install pygame
```

```
...
```

```
Successfully uninstalled pygame-1.9.2b1
```



pip 使用

- 可以通过list 子命令列出当前系统中已经安装的第三方库，例如：

- pip list

```
:>pip install sip
---
Metadata-Version: 1.1
Name: sip
Version: 4.18.1
Summary: Python extension module generator for C and C++ libraries
Home-page: https://www.riverbankcomputing.com/software/sip/
Author: Riverbank Computing Limited
Author-email: info@riverbankcomputing.com
Installer: pip
License: None
Location: d:\python35-32\lib\site-packages
Requires:
Classifiers:
```



pip 使用

- pip 的download 子命令可以下载第三方库的安装包，但并不安装，例如： pip download <拟下载库名>

```
:>pip download PyQt5
Collecting sip (from PyQt5)
  Downloading sip-4.18.1-cp35-none-win32.whl
  Saved c:\windows\system32\sip-4.18.1-cp35-none-win32.whl
Successfully downloaded PyQt5 sip
```




pip 使用

- pip 是Python 第三方库最主要的安装方式，可以安装超过90%以上的第三方库。然而，由于一些历史、技术和政策等原因，还有一些第三方库无法暂时用pip 安装，此时，需要其他的安装方法



pip 使用

- pip 工具与操作系统也有关系，在Mac OS X 和 Linux 等操作系统中，pip 工具几乎可以安装任何 Python 第三方库，在Windows 操作系统中，有一些第三方库仍然需要用其他方式尝试安装。



自定义安装

- 自定义安装指按照第三方库提供的步骤和方式安装。第三方库都有主页用于维护库的代码和文档。以科学计算用的numpy 为例，开发者维护的官方主页是：<http://www.numpy.org/>
- 浏览该网页找到下载链接，如下：
<http://www.scipy.org/scipylib/download.html>
- 进而根据指示步骤安装。自定义安装一般适合在pip 中尚无登记或安装失败的。



文件安装

- 由于Python 某些第三方库仅提供源代码，通过pip 下载文件后无法在Windows系统编译安装，会导致第三方库安装失败。在Windows 平台下遇到的无法安装第三方库的问题大多属于这类。



文件安装

- 为了解决这类第三方库安装问题，美国加州大学尔湾分校提供了一个页面，帮助Python 用户获得Windows 可直接安装的第三方库文件，链接地址如下：

- <http://www.lfd.uci.edu/~gohlke/pythonlibs/>



文件安装

- 该地址列出了一批在pip 安装中可能出现问题的第三方库。
- 这里以scipy 为例说明，首先在上述页面中找到scipy 库对应的内容



文件安装

- 选择其中的.whl 文件下载，这里选择适用于Python 3.5 版本解释器和32 位系统的对应文件：scipy-0.18.1-cp35-cp35m-win32.whl，下载该文件到D:\pycodes目录。

SciPy is software for mathematics, science, and engineering.
Requires `numpy+mk1`.
Install `numpy+mk1` before installing `scipy`.
[scipy-0.18.1-cp27-cp27m-win32.whl](#)
[scipy-0.18.1-cp27-cp27m-win_amd64.whl](#)
[scipy-0.18.1-cp34-cp34m-win32.whl](#)
[scipy-0.18.1-cp34-cp34m-win_amd64.whl](#)
[scipy-0.18.1-cp35-cp35m-win32.whl](#)
[scipy-0.18.1-cp35-cp35m-win_amd64.whl](#)



文件安装

- 然后，采用pip 命令安装该文件

```
: \> pip install D:\pycodes\scipy-0.18.1-cp35-cp35m-win32.whl
Processing d:\pycodes\scipy-0.18.1-cp35-cp35m-win32.whl
Installing collected packages: scipy
Successfully installed scipy-0.18.1
```




第三方库安装

- 对于上述三种安装方式，一般优先选择采用pip 工具安装，如果安装失败，则选择自定义安装或者文件安装（Windows 平台）。另外，如果需要在没有网络条件下安装Python 第三方库，请直接采用文件安装方式。其中，.whl 文件可以通过pip download 指令在有网络条件的情况下获得。



拓展：whl 格式

- .whl 是Python 库的一种打包格式，用于通过pip 进行安装，相当于Python 库的安装包文件。.whl 文件本质上是一个压缩格式文件，可以通过变化扩展名为.zip查看其中内容。.whl 格式用于替代Python 早期的eggs 格式，是Python 打包格式的事实标准。



实例16: pip 安装脚本



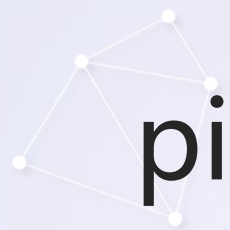
要点：

- 这是一个用pip 安装第三方库的例子



pip 安装脚本

- 实例16共需要安装20个第三方Python 库
- 需要注意，库名是第三方库常用的名字，pip 安装用的名字和库名不一定完全相同，建议采用小写字符。



pip 安装脚本

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




pip 安装脚本

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



pip 安装脚本

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



pip 安装脚本

- 安装过程请在系统命令行下进行，不在IDLE 中，部分库会依赖其他函数库，pip会自动安装，部分库下载后需要一个安装过程，pip 也会自动执行。成功安

```
:\>pip install pygame
```

```
...
```

```
Installing collected packages: pygame
```

```
Successfully installed pygame-1.9.2b1
```



pip 批量安装Python 库

- 如果读者希望自动安装这些库，可以使用Python 标准库os 的system()函数调用控制台。
- 下面给出采用pip 批量安装Python 库的方法。



pip 批量安装Python 库

```
1  #e16.1BatchInstall.py
2  import os
3  libs = {"numpy","matplotlib","pillow","sklearn","requests",\
          "jieba","beautifulsoup4","wheel","networkx","sympy",\
          "pyinstaller","django","flask","werobot","PyQt5",\
          "pandas","pyopengl","pypdf2","docopt","pygame"}
4  try:
5      for lib in libs:
6          os.system("pip install "+lib)
7          print("Successful")
8  except:
9      print("Failed Somehow")
```



拓展：PyPI 的权重值

- PyPI 提供了第三方库的索引，除了基本信息外，PyPI 还根据每个库被检索和下载情况计算了权重值（Weight）。由于第三方库的开发没有任何规划，对某个功能将有一批库可以支持，权重值较高的库往往质量更好。