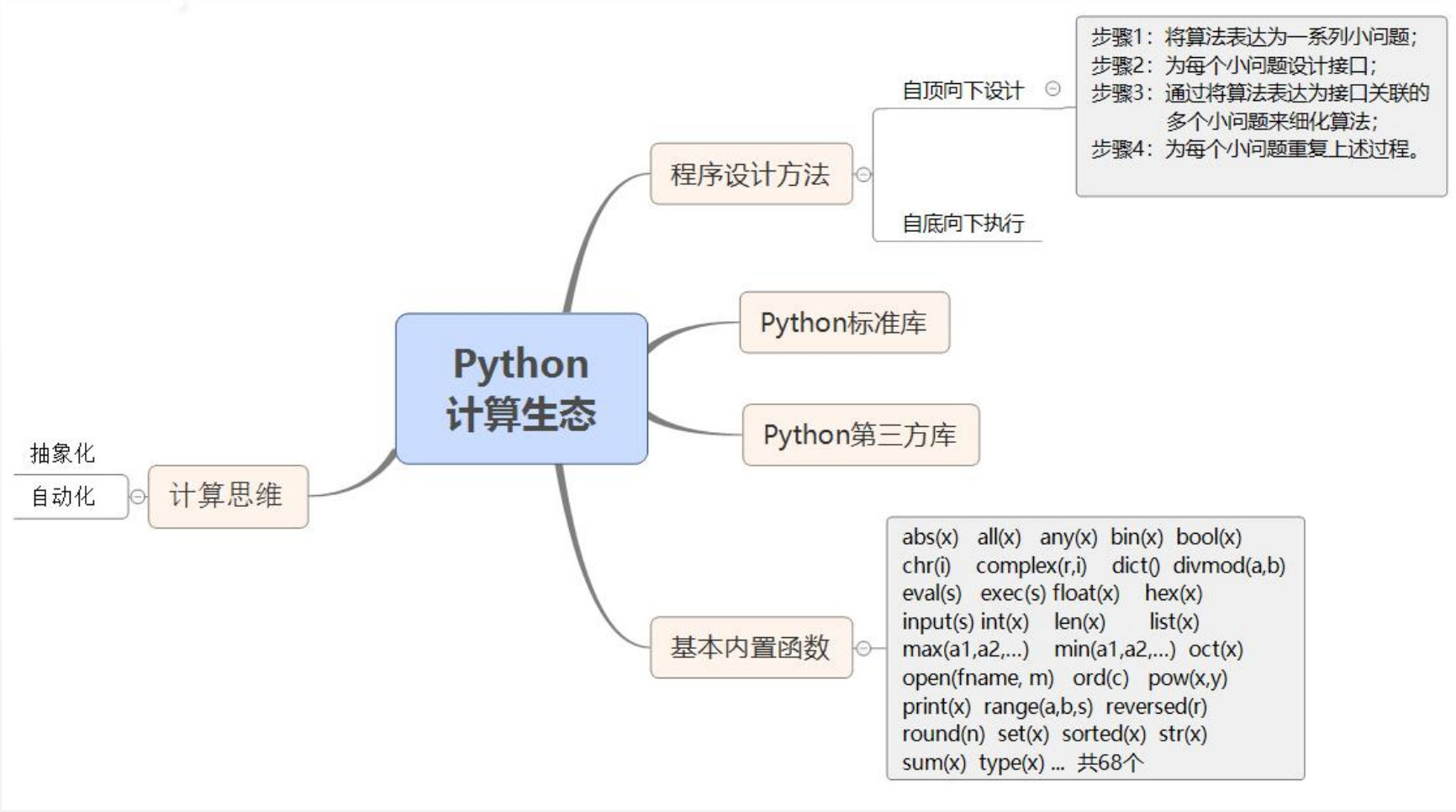
全国计算机等级考试二级教程
Python语言程序设计
(2018年版)

【第8章】
Python计算生态

考纲考点

- 基本的Python内置函数
- 了解Python计算生态

知识导图


Python
计算生态
程序设计方法
自顶向下设计
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。
自底向下执行
Python标准库
Python第三方库
基本内置函数
abs(x) all(x) any(x) bin(x) bool(x)
chr(i) complex(r,i) dict() divmod(a,b)
eval(s) exec(s) float(x) hex(x)
input(s) int(x) len(x) list(x)
max(a1,a2,...) min(a1,a2,...) oct(x)
open(fname, m) ord(c) pow(x,y)
print(x) range(a,b,s) reversed(r)
round(n) set(x) sorted(x) str(x)
sum(x) type(x) ... 共68个
计算思维
抽象化
自动化

计算思维

计算思维

- 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。
- 计算思维的本质是**抽象**（Abstraction）和**自动化**（Automation）

程序设计方法论

程序设计方法论

- 一个解决复杂问题行之有效的方法被称作**自顶而下的设计**方法，其基本思想是以一个总问题开始，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。

程序设计方法论

“体育竞技分析”实例

- 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。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。

自顶向下设计

- 自顶向下设计中最重要的是顶层设计。体育竞技分析从用户处得到模拟参数，最后输出结果。下面是一个基础设计：

- 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次比赛；
- 步骤4：输出球员A和B获胜比赛的场次及概率。

自顶向下设计

- 步骤1　输出一些介绍信息，针对提升用户体验十分有益。下面是这个步骤的Python代码，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

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

自顶向下设计

- 步骤2　获得用户输入。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，全部代码如下。

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

自顶向下设计

- 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

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

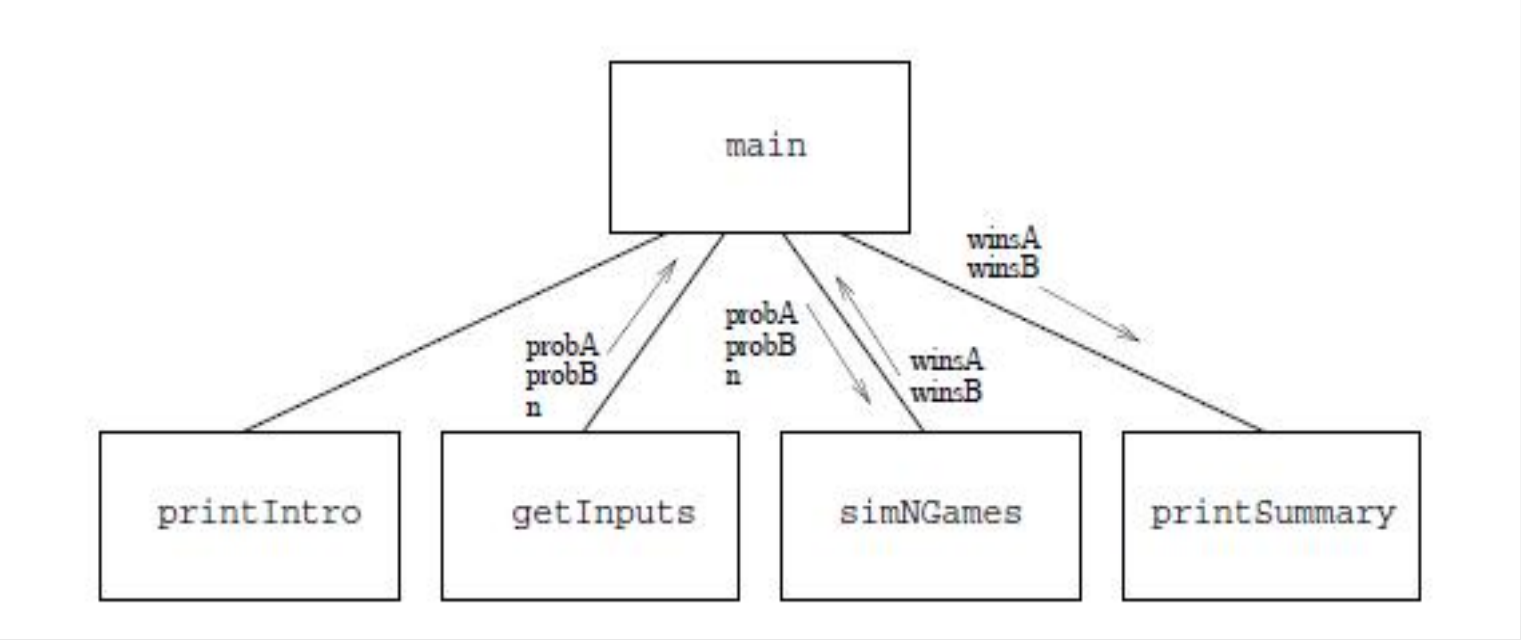
自顶向下设计

- 步骤4　输出结果，设计思想类似，仍然只规划功能和函数，代码如下。

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

自顶向下设计

- 原问题被划分为了4个独立的函数：printIntro()，getInputs()，simNGames()和printSummary()。



自顶向下设计

- 每层设计中，参数和返回值如何设计是重点，其他细节可以暂时忽略。确定事件的重要特征而忽略其它细节过程称为抽象。抽象是一种基本设计方法，自顶向下的设计过程可以看作是发现功能并抽象功能的过程。

自顶向下设计

- printIntro()函数应该输出一个程序介绍，这个功能的Python代码如下，这个函数由Python基本表达式组合，不增加或改变程序结构。

```
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）")
```

自顶向下设计

- getInputs()函数根据提示得到三个需要返回主程序的值，代码如下。

```
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
```

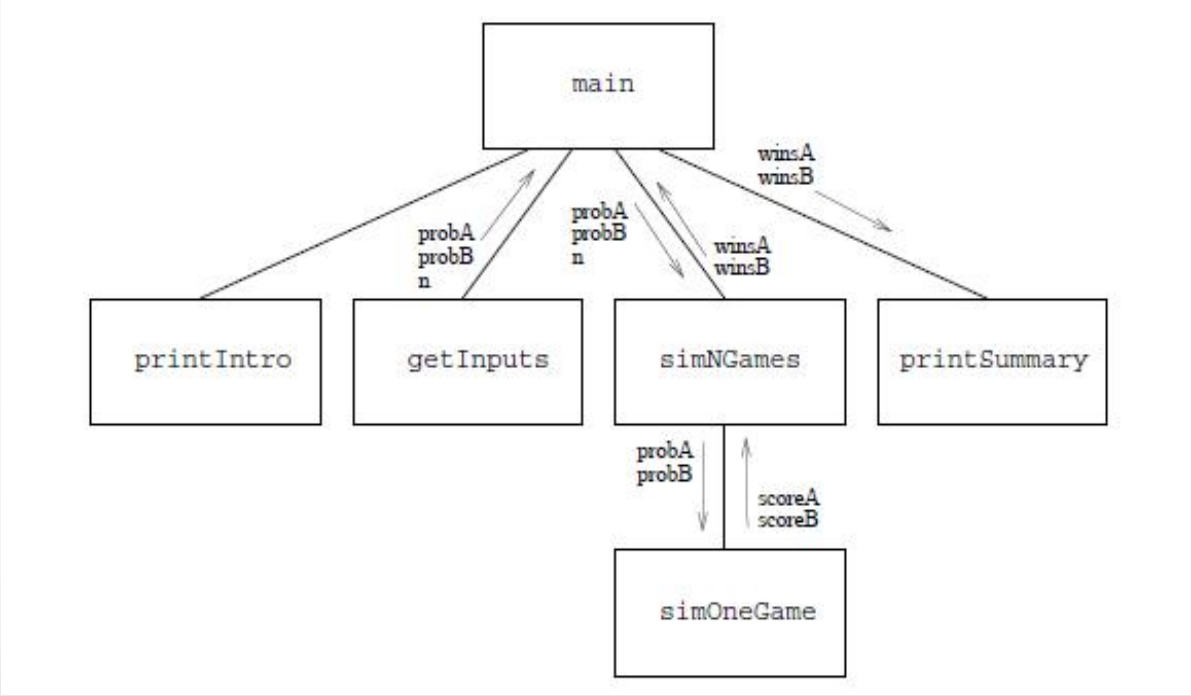
自顶向下设计

- 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。

```
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
```

自顶向下设计

- 代码中设计了simOneGame()函数，用于模拟一场比赛，这个函数需要知道每个球员的概率，返回两个球员的最终得分



自顶向下设计

- 接下来需要实现simOneGame()函数。
- 为了模拟一场比赛，需要根据比赛规则来编写代码，两个球员A和B持续对攻直至比赛结束。可以采用无限循环结构直到比赛结束条件成立。同时，需要跟踪记录比赛得分，保留发球局标记，

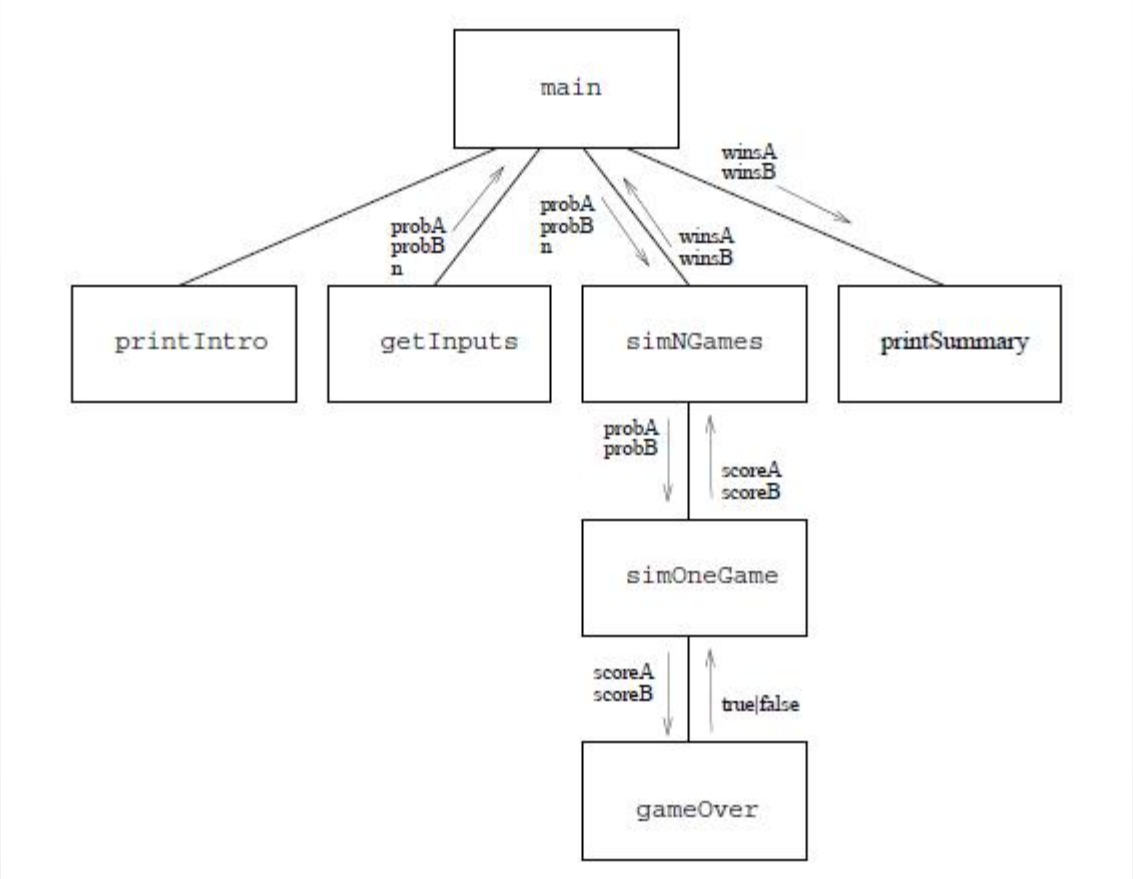
自顶向下设计

- 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，更新球员A得分或是将球权交给球员B。

自顶向下设计

```
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
```

自顶向下设计


main
printIntro
getInputs
simNGames
printSummary
simOneGame
gameOver
probA
probB
n
winsA
winsB
scoreA
scoreB
true|false

自顶向下设计

- 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
```

自顶向下设计

- 最后是printSummary()函数，其Python代码如下。

```
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))
```

自顶向下设计

- 最后是printSummary()函数，其Python代码如下。

```
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))
```

自顶向下设计

- 将上述所有代码放在一起，形成了实例全部代码。

行结果如下：

```
>>>
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%
```

自顶向下设计

- 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。每一层以大体算法描述开始，然后逐步细化成代码，细节被函数封装

自顶向下设计

整个过程可以概括为四个步骤：

- 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

自底向上执行

- 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>

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

自底向上执行

- 初步测试说明gameOver()函数是正确的。进一步测试simOneGame()函数，如下：

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

自底向上执行

- 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。

计算生态

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**”。
- 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。

计算生态

- Python官方网站提供了第三方库索引功能（PyPI，the Python Package Index），网址如下：

  https://pypi.python.org/pypi

- 该页面列出了Python语言超过12万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。

计算生态

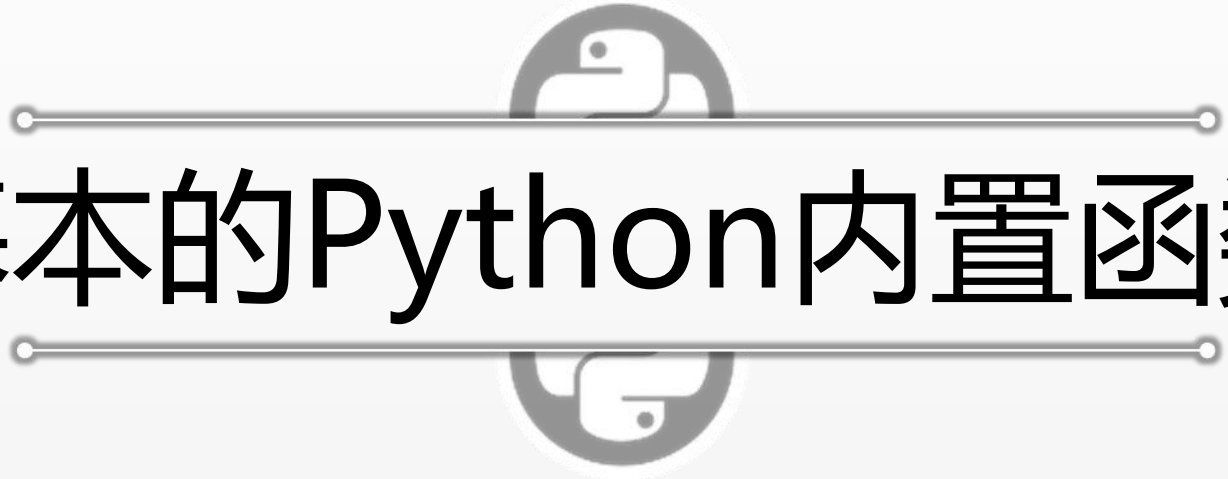
- 由于Python有非常简单灵活的编程方式，很多采用C、C++等语言编写的专业库可以经过简单的接口封装供Python语言程序调用。这样的**粘性功能**使得Python语言成为了各类编程语言之间的接口，Python语言也被称为“**胶水语言**”。

Python标准库

- 有一部分Python计算生态随Python安装包一起发布，用户可以随时使用，被称为**Python标准库**。
- 受限于Python安装包的设定大小，标准库数量270个左右。

Python第三方库

- 更广泛的Python计算生态采用额外安装方式服务用户，被称为Python第三方库。这些第三方库由全球各行业专家、工程师和爱好者开发，没有顶层设计，由开发者采用“尽力而为”的方式维护。Python通过新一代安装工具pip管理大部分Python第三方库的安装。



基本的Python内置函数

基本的Python内置函数

- Python解释器提供了68个内置函数（下面介绍32个）

函数名称	函数说明
abs(x)	x的绝对值 如果x是复数，返回复数的模
all(x)	组合类型变量x中所有元素都为真时返回True，否则返回False；若x为空，返回True
any(x)	组合类型变量x中任一元素都为真时返回True，否则返回False；若x为空，返回False
bin(x)	将整数x转换为等值的二进制字符串 bin(1010)的结果是'0b1111110010'
bool(x)	将x转换为Boolean类型，即True或False bool('') 的结果是False
chr(i)	返回Unicode为i的字符 chr(9996)的结果是'✌ '
complex(r,i)	创建一个复数 r + i*1j，其中i可以省略 complex(10,10)的结果是10+10j
dict()	创建字典类型 dict()的结果是一个空字典{}
divmod(a,b)	返回a和b的商及余数 divmod(10,3)结果是一个(3,1)

基本的Python内置函数

函数名称	函数说明
eval(s)	计算字符串s作为Python表达式的值 eval('1+99')的结果是100
exec(s)	计算字符串s作为Python语句的值 exec('a = 1+999')运行后，变量a的值为1000
float(x)	将x转换成浮点数 float(1010)的结果是1010.0
hex(x)	将整数转换为16进制字符串 hex(1010)的结果是'0x3f2'
input(s)	获取用户输入，其中s是字符串，作为提示信息 s可选
int(x)	将x转换成整数 int(9.9)的结果是9
list(x)	创建或将变量x转换成一个列表类型 list({10,9,8})的结果是[8,9,10]
max(a1,a2,…)	返回参数的最大值 max(1,2,3,4,5)的结果是5
min(a1,a2,…)	返回参数的最小值 min(1,2,3,4,5)的结果是1
oct(x)	将整数x转换成等值的八进制字符串形式 oct(1010)的结果是'0o1762'
open(fname, m)	打开文件，包括文本方式和二进制方式等 其中，m部分可以省略，默认是以文本可读形式打开

基本的Python内置函数

函数名称	函数说明
ord(c)	返回一个字符的Unicode编码值 ord('字')的结果是23383
pow(x,y)	返回x的y次幂 pow(2,pow(2,2))的结果是16
print(x)	打印变量或字符串x print()的end参数用来表示输出的结尾字符
range(a,b,s)	从a到b(不含)以s为步长产生一个序列 list(range(1,10,3))的结果是[1, 4, 7]
reversed(r)	返回组合类型r的逆序迭代形式 for i in reversed([1,2,3])将逆序遍历列表
round(n)	四舍五入方式计算n round(10.6)的结果是11
set(x)	将组合数据类型x转换成集合类型 set([1,1,1,1])的结果是{1}
sorted(x)	对组合数据类型x进行排序，默认从小到大 sorted([1,3,5,2,4])的结果是[1,2,3,4,5]
str(x)	将x转换为等值的字符串类型 str(0x1010)的结果是'4112'
sum(x)	对组合数据类型x计算求和结果 sum([1,3,5,2,4])的结果是15
type(x)	返回变量x的数据类型 type({1:2})的结果是<class 'dict'>

实例解析：Web页面元素提取

Web页面元素提取

- Web页面，一般是HTML页面，是Internet组织信息的基础元素。Web页面元素提取是一类常见问题，在网络爬虫、浏览器等程序中有着不可或缺的重要作用。
- HTML指超文本标记语言，严格来说，HTML不是一种编程语言，而是一种对信息的标记语言，对Web的内容、格式进行描述。

Web页面元素提取

- 自动地从一个链接获取HTML页面是网络爬虫的功能，本实例功能可以整体分成如下4个步骤：
  - 步骤1：读取保存在本地的html文件；
  - 步骤2：解析并提取其中的图片链接；
  - 步骤3：输出提取结果到屏幕；
  - 步骤4：保存提取结果为文件。

Web页面元素提取

- 根据上述步骤，可以写出主程序如下。其中设置了4个函数getHTMLlines()、extractImageUrls()、showResults()和saveResults()分别对应上述4个步骤。

```
def main():
    inputfile  = 'nationalgeographic.html'
    outputfile = 'nationalgeographic-urls.txt'
    htmlLines = getHTMLlines(inputfile)
    imageUrls = extractImageUrls(htmlLines)
    showResults(imageUrls)
    saveResults(outputfile, imageUrls)
```

Web页面元素提取

- 定义main()函数的目的是为了让代码更加清晰，作为主程序，也可以不采用函数形式而直接编写。main()前两行分别制定了拟获取HTML文件的路径和结果输出路径。
- 主函数设计完成后，逐一编写各函数功能。

Web页面元素提取

- getHTMLlines()函数读取HTML文件并内容，并将结果转换为一个分行列表，为了兼容不同编码，建议在open()函数中增加encoding字段，设置采用UTF-8编码打开文件。代码如下。

```
1  def getHTMLlines(htmlpath):
2      f = open(htmlpath, "r", encoding='utf-8')
3      ls = f.readlines()
4      f.close()
5      return ls
```

Web页面元素提取

- extractImageUrls()是程序的核心，用于解析文件并提取图像的URL。观察HTML可以发现，图像采用img标签表示，例如：

```
<img title="photo story"
src="http://image.nationalgeographic.com.cn/2018/0122/20180122042251164.jpg" width="968px" />
```

Web页面元素提取

- 其中，<img开头是图像标签的特点，其中由src=所引导的URL是这个图像的真实位置。每个URL都以http开头。因此，可以通过字符串操作提取其中的图像链接。

```
def extractImageUrls(htmllist):
    urls = []
    for line in htmllist:
        if 'img' in line:
            url = line.split('src=')[-1].split('"')[1]
            if 'http' in url:
                urls.append(url)
    return urls
```

Web页面元素提取

- showResults()函数将获取的链接输出到屏幕上，增加一个计数变量提供更好用户体验，代码如下。

```
def showResults(urls):
    count = 0
    for url in urls:
        print('第{:2}个URL:{}'.format(count, url))
        count += 1
```

- saveResults()保存结果到文件，代码如下。

```
def saveResults(filepath, urls):
    f = open(filepath, "w")
    for url in urls:
        f.write(url+"\n")
    f.close()
```

Web页面元素提取

- 各部分函数代码编写后，全部代码功能具备，需要额外调用main()函数

```
>>>
第 0个URL:http://image.nationalgeographic.com.cn/2018/0125/20180125103409949.jpg
第 1个URL:http://image.nationalgeographic.com.cn/2018/0124/20180124105929614.jpg
第 2个URL:http://image.nationalgeographic.com.cn/2018/0122/20180122042251164.jpg
第 3个URL:http://image.nationalgeographic.com.cn/2018/0122/20180122120753804.jpg
第 4个URL:http://image.nationalgeographic.com.cn/2018/0122/20180122102058707.jpg
第 5个URL:http://image.nationalgeographic.com.cn/2018/0125/20180125050715797.jpg
第 6个URL:http://image.nationalgeographic.com.cn/2018/0124/20180124052148836.jpg
第 7个URL:http://image.nationalgeographic.com.cn/2018/0123/20180123023114680.jpg
第 8个URL:http://image.nationalgeographic.com.cn/2018/0122/20180122035438691.jpg
第 9个URL:http://image.nationalgeographic.com.cn/2018/0118/20180118040311659.jpg
第10个URL:http://image.nationalgeographic.com.cn/2018/0125/20180125115939982.jpg
（此后略去40个输出）
```

本章小结

本章主要讲解程序设计方法学，包括计算思维、自顶向下设计和自底向上执行等，进一步本章介绍了计算生态的概念及Python标准库和第三方库的划分。通过Web页面元素提取的实例帮助读者理解自顶向下设计的基本方法。

从最基本的IPO到自顶向下设计，是否感受到了函数式编程的优势？