第八章

python计算生态

本章大纲

1.计算思维

2.程序设计方法论

3.计算生态

4.基本的python内置函数

1.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）

2.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。

3. python的内置函数

abs()	all()	any()	bin()	bool()	chr()
complex()	dict()	divmod()	eval()	exec()	float()
hex()	input()	int()	len()	list()	max()
min()	oct()	open()	ord()	pow()	print()
range()	reversed()	round()	set()	sorted()	str()
sum()	type()				

4.实例

“体育竞技分析”实例 n 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。

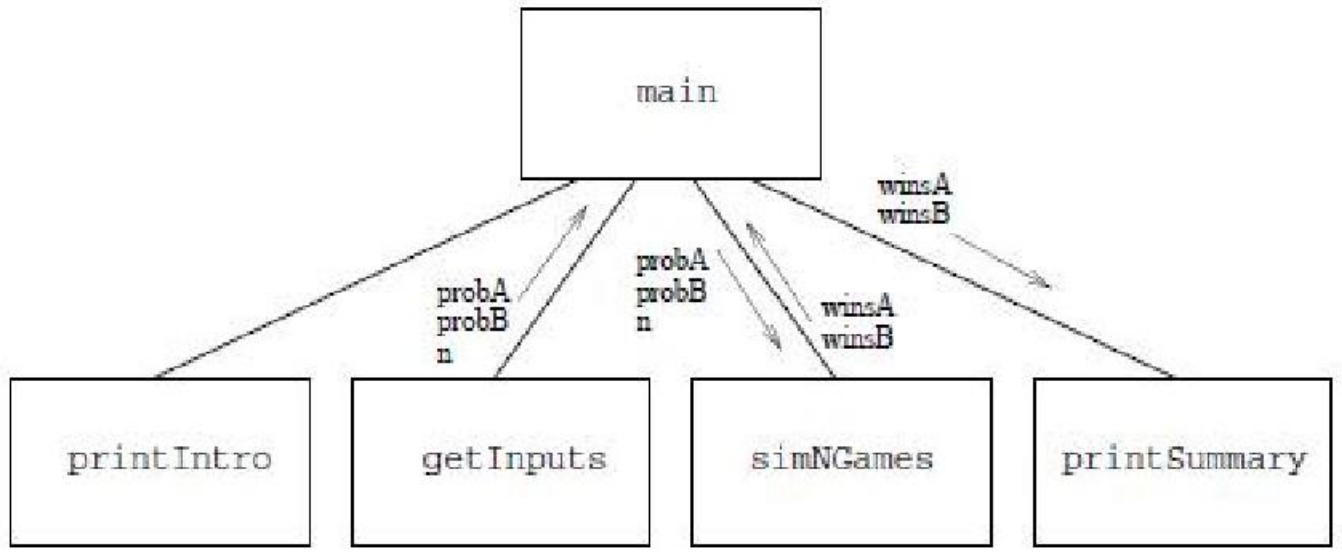
自顶向下设计中最重要的是顶层设计。体育竞技

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分析：

- 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。

通过以上分析：问题被划分为了4个独立的函数：printIntro()，getInputs()，simNGames()和printSummary()。



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。

1、print_Intro()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。

```
def print_intro():
    print("这是一个体育竞技分析项目")
    print("输出a和b的能力值(0-1)以及比赛次数")
```

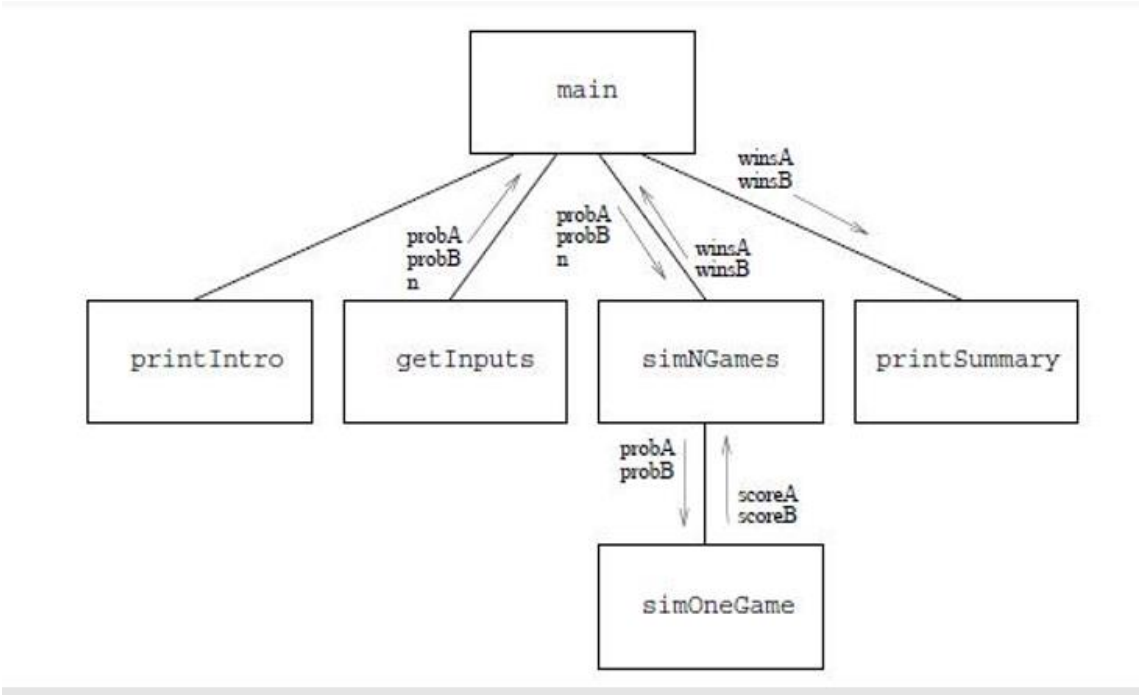
2、get_Inputs()函数根据提示得到三个需要返回主程序的值，代码如下。

```
def get_inputs():
    a = eval(input("请输入a的能力值(0-1): "))
    b = eval(input("请输入b的能力值(0-1): "))
    n = eval(input("请输入比赛次数: "))
    return a, b, n
```

3、sim_game()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 sim_game(prob_a,prob_b,n):
    #wins_a,wins_b代表两位选手赢得次数
    wins_a,wins_b = 0,0
    for i in range(n):
        score_a,score_b = get_one_score(prob_a,prob_b)
        if score_a>score_b:
            wins_a+=1
        else:
            wins_b+=1
    return wins_a,wins_b
```

4、代码中设计了get_one_score()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



5、接下来需要实现get_one_score()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

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。

```
def get_one_score(prob_a,prob_b):
    #两个选手的分数
    score_a,score_b = 0,0
    flag = 'A'
    while score_a<15 and score_b<15:
        if flag == 'A':
            if random.random()<prob_a:
                score_a+=1
            else:
                flag = 'B'
        else:
            if random.random()<prob_b:
                score_b+=1
            else:
                flag = 'A'
    return score_a,score_b
```

7、最后是print_res()函数，其Python代码如下。

```
def print_res(wins_a,wins_b):
    n = wins_a + wins_b
    print("一共进行了"+n+'次比赛')
    print("a赢了"+str(wins_a)+'次')
    print('b赢了'+str(wins_b)+ '次')
```

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；

n 步骤4：为每个小问题重复上述过程。

自底向上执行

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