# 6.函数

## 6.1.函数的基本语法

### 6.1.1.为什么要用函数

1、提高代码复用性——抽象出来，封装为函数

2、将复杂的大问题分解成一系列小问题，分而治之——模块化设计的思想

3、利于代码的维护和管理

顺序式代码

*# 5的阶乘*

n = 5

res = 1

**for** i **in** range(1, n+1):

res \*= i

print(res)

*# 20的阶乘*

n = 20

res = 1

**for** i **in** range(1, n+1):

res \*= i

print(res)

120

2432902008176640000

抽象成函数

**def** factoria(n):

res = 1

**for** i **in** range(1,n+1):

res \*= i

**return** res

print(factoria(5))

print(factoria(20))

120

2432902008176640000

### 6.1.2.函数的定义及调用

白箱子：输入——处理——输出

三要素：参数、函数体、返回值

1、定义

def  函数名（参数）：

  函数体

  return 返回值

*# 求正方形的面积*

**def** area\_of\_square(length\_of\_side):

square\_area = pow(length\_of\_side, 2)

**return** square\_area

2、调用

函数名（参数）

area = area\_of\_square(5)

area

25

### 6.1.3.参数传递

**1、形参与实参**

形参（形式参数）：函数定义时的参数，实际上就是变量名

实参（实际参数）：函数调用时的参数，实际上就是变量的值

**2、位置参数**

严格按照位置顺序，用实参对形参进行赋值(关联）

一般用在参数比较少的时候

**def** function(x, y, z):

print(x, y, z)

function(1, 2, 3) *# x = 1; y = 2; z = 3*

1 2 3

实参与形参个数必须一一对应，一个不能多，一个不能少

function(1, 2)

**TypeError**: function() missing 1 required positional argument: 'z'

function(1, 2, 3, 4)

**TypeError**: function() takes 3 positional arguments but 4 were given

**3、关键字参数**

打破位置限制，直呼其名的进行值的传递（形参=实参）

必须遵守实参与形参数量上一一对应

多用在参数比较多的场合

**def** function(x, y, z):

print(x, y, z)

function(y=1, z=2, x=3) *# x = 1; y = 2; z = 3*

3 1 2

位置参数可以与关键字参数混合使用

但是，位置参数必须放在关键字参数前面

function(1, z=2, y=3)

1 3 2

function(1, 2, z=3)

1 2 3

不能为同一个形参重复传值

**def** function(x, y, z):

print(x, y, z)

function(1, z=2, x=3)

**TypeError**: function() got multiple values for argument 'x'

**4、默认参数**

在定义阶段就给形参赋值——该形参的常用值

默认参数必须放在非默认参数后面

调用函数时，可以不对该形参传值

机器学习库中类的方法里非常常见

**def** register(name, age, sex="male"):

print(name, age, sex)

register("大杰仔", 18)

大杰仔 18 male

也可以按正常的形参进行传值

register("林志玲", 38, "female")

林志玲 38 female

默认参数应该设置为不可变类型（数字、字符串、元组）

**def** function(ls=[]):

print(id(ls))

ls.append(1)

print(id(ls))

print(ls)

function()

1759752744328

1759752744328

[1]

function()

1759752744328

1759752744328

[1, 1]

function()

1759752744328

1759752744328

[1, 1, 1]

**def** function(ls="Python"):

print(id(ls))

ls += "3.7"

print(id(ls))

print(ls)

function()

1759701700656

1759754352240

Python3.7

function()

1759701700656

1759754353328

Python3.7

function()

1759701700656

1759754354352

Python3.7

让参数变成可选的

**def** name(first\_name, last\_name, middle\_name=**None**):

**if** middle\_name:

**return** first\_name+middle\_name+last\_name

**else**:

**return** first\_name+last\_name

print(name("大","仔"))

print(name("大", "仔", "杰"))

大仔

大杰仔

**5、可变长参数 \*args**

不知道会传过来多少参数 \*args

该形参必须放在参数列表的最后

**def** foo(x, y, z, \*args):

print(x, y ,z)

print(args)

foo(1, 2, 3, 4, 5, 6) *# 多余的参数，打包传递给args*

1 2 3

(4, 5, 6)

实参打散

**def** foo(x, y, z, \*args):

print(x, y ,z)

print(args)

foo(1, 2, 3, [4, 5, 6])

1 2 3

([4, 5, 6],)

foo(1, 2, 3, \*[4, 5, 6]) *# 打散的是列表、字符串、元组或集合*

1 2 3

(4, 5, 6)

**6、可变长参数 \*\*kwargs**

**def** foo(x, y, z, \*\*kwargs):

print(x, y ,z)

print(kwargs)

foo(1, 2, 3, a=4, b=5, c=6) *# 多余的参数，以字典的形式打包传递给kwargs*

1 2 3

{'a': 4, 'b': 5, 'c': 6}

字典实参打散

**def** foo(x, y, z, \*\*kwargs):

print(x, y ,z)

print(kwargs)

foo(1, 2, 3, \*\*{"a": 4, "b": 5, "c":6})

1 2 3

{'a': 4, 'b': 5, 'c': 6}

可变长参数的组合使用

**def** foo(\*args, \*\*kwargs):

print(args)

print(kwargs)

foo(1, 2, 3, a=4, b=5, c=6)

(1, 2, 3)

{'a': 4, 'b': 5, 'c': 6}

### 6.1.4.函数体与变量作用域

函数体就是一段只在函数被调用时，才会执行的代码，代码构成与其他代码并无不同

**局部变量**

仅在函数体内定义和发挥作用

**def** multipy(x, y):

z = x\*y

**return** z

multipy(2, 9)

print(z) *# 函数执行完毕，局部变量z已经被释放掉了*

**NameError**: name 'z' is not defined

**全局变量**

外部定义的都是全局变量

全局变量可以在函数体内直接被使用

n = 3

ls = [0]

**def** multipy(x, y):

z = n\*x\*y

ls.append(z)

**return** z

print(multipy(2, 9))

ls

54

[0, 54]

通过global 在函数体内定义全局变量

**def** multipy(x, y):

**global** z

z = x\*y

**return** z

print(multipy(2, 9))

print(z)

18

18

### 6.1.5.返回值

**1、单个返回值**

**def** foo(x):

**return** x\*\*2

res = foo(10)

res

100

**2、多个返回值**——以元组的形式

**def** foo(x):

**return** 1, x, x\*\*2, x\*\*3 *# 逗号分开，打包返回*

print(foo(3))

(1, 3, 9, 27)

a, b , c, d = foo(3) *# 解包赋值*

print(a)

print(b)

print(c)

print(d)

1

3

9

27

**3、可以有多个return 语句，一旦其中一个执行，代表了函数运行的结束**

**def** is\_holiday(day):

**if** day **in** ["Sunday", "Saturday"]:

**return** "Is holiday"

**else**:

**return** "Not holiday"

print("啦啦啦德玛西亚，啦啦啦啦") *# 你丫根本没机会运行。。。*

print(is\_holiday("Sunday"))

print(is\_holiday("Monday"))

Is holiday

Not holiday

**4、没有return语句，返回值为None**

**def** foo():

print("我是孙悟空")

res = foo()

print(res)

我是孙悟空

None

### 6.1.6.几点建议

1、函数及其参数的命名参照变量的命名

字母小写及下划线组合

有实际意义

2、应包含简要阐述函数功能的注释，注释紧跟函数定义后面

**def** foo():

*# 这个函数的作用是为了给大家瞅一瞅，你瞅啥，瞅你咋地。。。。*

**pass**

3、函数定义前后各空两行

**def** f1():

**pass**

*# 空出两行，以示清白*

**def** f2():

**pass**

**def** f3(x=3): *# 默认参数赋值等号两侧不需加空格*

**pass**

*# ...*

4、默认参数赋值等号两侧不需加空格

## 6.2.函数式编程实例

**模块化编程思想**

自顶向下，分而治之

【问题描述】

小丹和小伟羽毛球打的都不错，水平也在伯仲之间，小丹略胜一筹，基本上，打100个球，小丹能赢55次，小伟能赢45次。

但是每次大型比赛（1局定胜负，谁先赢到21分，谁就获胜），小丹赢的概率远远大于小伟，小伟很是不服气。

亲爱的小伙伴，你能通过模拟实验，来揭示其中的奥妙吗？

【问题抽象】

1、在小丹Vs小伟的二元比赛系统中，小丹每球获胜概率55%，小伟每球获胜概率45%；

2、每局比赛，先赢21球（21分）者获胜；

3、假设进行n = 10000局独立的比赛，小丹会获胜多少局？（n 较大的时候，实验结果≈真实期望）

【问题分解】

**def** main():

*# 主要逻辑*

prob\_A, prob\_B, number\_of\_games = get\_inputs() *# 获取原始数据*

win\_A, win\_B = sim\_n\_games(prob\_A, prob\_B, number\_of\_games) *# 获取模拟结果*

print\_summary(win\_A, win\_B, number\_of\_games) *# 结果汇总输出*

1、输入原始数据

**def** get\_inputs():

*# 输入原始数据*

prob\_A = eval(input("请输入运动员A的每球获胜概率(0~1)："))

prob\_B = round(1-prob\_A, 2)

number\_of\_games = eval(input("请输入模拟的场次（正整数）："))

print("模拟比赛总次数：", number\_of\_games)

print("A 选手每球获胜概率：", prob\_A)

print("B 选手每球获胜概率：", prob\_B)

**return** prob\_A, prob\_B, number\_of\_games

单元测试

prob\_A, prob\_B, number\_of\_games = get\_inputs()

print(prob\_A, prob\_B, number\_of\_games)

请输入运动员A的每球获胜概率(0~1)：0.55

请输入模拟的场次（正整数）：10000

模拟比赛总次数： 10000

A 选手每球获胜概率： 0.55

B 选手每球获胜概率： 0.45

0.55 0.45 10000

2、多场比赛模拟

**def** sim\_n\_games(prob\_A, prob\_B, number\_of\_games):

*# 模拟多场比赛的结果*

win\_A, win\_B = 0, 0 *# 初始化A、B获胜的场次*

**for** i **in** range(number\_of\_games): *# 迭代number\_of\_games次*

score\_A, score\_B = sim\_one\_game(prob\_A, prob\_B) *# 获得模拟依次比赛的比分*

**if** score\_A > score\_B:

win\_A += 1

**else**:

win\_B += 1

**return** win\_A, win\_B

**import** **random**

**def** sim\_one\_game(prob\_A, prob\_B):

*# 模拟一场比赛的结果*

score\_A, score\_B = 0, 0

**while** **not** game\_over(score\_A, score\_B):

**if** random.random() < prob\_A: *# random.random() 生产[0,1)之间的随机小数,均匀分布*

score\_A += 1

**else**:

score\_B += 1

**return** score\_A, score\_B

**def** game\_over(score\_A, score\_B):

*# 单场模拟结束条件，一方先达到21分，比赛结束*

**return** score\_A == 21 **or** score\_B == 21

  单元测试 用assert——断言

assert expression

表达式结果为 false 的时候触发异常

**assert** game\_over(21, 8) == **True**

**assert** game\_over(9, 21) == **True**

**assert** game\_over(11, 8) == **False**

**assert** game\_over(21, 8) == **False**

**AssertionError**:

print(sim\_one\_game(0.55, 0.45))

print(sim\_one\_game(0.7, 0.3))

print(sim\_one\_game(0.2, 0.8))

(21, 7)

(21, 14)

(10, 21)

print(sim\_n\_games(0.55, 0.45, 1000))

(731, 269)

3、结果汇总输出

**def** print\_summary(win\_A, win\_B, number\_of\_games):

*# 结果汇总输出*

print("共模拟了**{}**场比赛".format(number\_of\_games))

print("选手A获胜**{0}**场，占比**{1:.1%}**".format(win\_A, win\_A/number\_of\_games))

print("选手B获胜**{0}**场，占比**{1:.1%}**".format(win\_B, win\_B/number\_of\_games))

print\_summary(729, 271, 1000)

共模拟了1000场比赛

选手A获胜729场，占比72.9%

选手B获胜271场，占比27.1%

综合所有代码：

**import** **random**

**def** get\_inputs():

*# 输入原始数据*

prob\_A = eval(input("请输入运动员A的每球获胜概率(0~1)："))

prob\_B = round(1-prob\_A, 2)

number\_of\_games = eval(input("请输入模拟的场次（正整数）："))

print("模拟比赛总次数：", number\_of\_games)

print("A 选手每球获胜概率：", prob\_A)

print("B 选手每球获胜概率：", prob\_B)

**return** prob\_A, prob\_B, number\_of\_games

**def** game\_over(score\_A, score\_B):

*# 单场模拟结束条件，一方先达到21分，比赛结束*

**return** score\_A == 21 **or** score\_B == 21

**def** sim\_one\_game(prob\_A, prob\_B):

*# 模拟一场比赛的结果*

score\_A, score\_B = 0, 0

**while** **not** game\_over(score\_A, score\_B):

**if** random.random() < prob\_A: *# random.random() 生产[0,1)之间的随机小数,均匀分布*

score\_A += 1

**else**:

score\_B += 1

**return** score\_A, score\_B

**def** sim\_n\_games(prob\_A, prob\_B, number\_of\_games):

*# 模拟多场比赛的结果*

win\_A, win\_B = 0, 0 *# 初始化A、B获胜的场次*

**for** i **in** range(number\_of\_games): *# 迭代number\_of\_games次*

score\_A, score\_B = sim\_one\_game(prob\_A, prob\_B) *# 获得模拟依次比赛的比分*

**if** score\_A > score\_B:

win\_A += 1

**else**:

win\_B += 1

**return** win\_A, win\_B

**def** print\_summary(win\_A, win\_B, number\_of\_games):

*# 结果汇总输出*

print("共模拟了**{}**场比赛".format(number\_of\_games))

print("**\033**[31m选手A获胜**{0}**场，占比**{1:.1%}**".format(win\_A, win\_A/number\_of\_games))

print("选手B获胜**{0}**场，占比**{1:.1%}**".format(win\_B, win\_B/number\_of\_games))

**def** main():

*# 主要逻辑*

prob\_A, prob\_B, number\_of\_games = get\_inputs() *# 获取原始数据*

win\_A, win\_B = sim\_n\_games(prob\_A, prob\_B, number\_of\_games) *# 获取模拟结果*

print\_summary(win\_A, win\_B, number\_of\_games) *# 结果汇总输出*

**if** \_\_name\_\_ == "\_\_main\_\_":

main()

请输入运动员A的每球获胜概率(0~1)：0.52

请输入模拟的场次（正整数）：10000

模拟比赛总次数： 10000

A 选手每球获胜概率： 0.52

B 选手每球获胜概率： 0.48

共模拟了10000场比赛选手A获胜6033场，占比60.3%

选手B获胜3967场，占比39.7%

真实数据：

经统计，小丹跟小伟14年职业生涯，共交手40次，小丹以28:12遥遥领先。

其中，两人共交战整整100局：

小丹获胜61局，占比61%；

小伟获胜39局，占比39%。

**你以为你跟别人的差距只是一点点，实际上，差距老大了**

## 6.3.匿名函数

**1、基本形式**

lambda 变量: 函数体

**2、常用用法**

在参数列表中最适合使用匿名函数，尤其是与key = 搭配

排序sort() sorted()

ls = [(93, 88), (79, 100), (86, 71), (85, 85), (76, 94)]

ls.sort()

ls

[(76, 94), (79, 100), (85, 85), (86, 71), (93, 88)]

ls.sort(key = **lambda** x: x[1])

ls

[(86, 71), (85, 85), (93, 88), (76, 94), (79, 100)]

ls = [(93, 88), (79, 100), (86, 71), (85, 85), (76, 94)]

temp = sorted(ls, key = **lambda** x: x[0]+x[1], reverse=**True**)

temp

[(93, 88), (79, 100), (85, 85), (76, 94), (86, 71)]

max() min()

ls = [(93, 88), (79, 100), (86, 71), (85, 85), (76, 94)]

n = max(ls, key = **lambda** x: x[1])

n

(79, 100)

n = min(ls, key = **lambda** x: x[1])

n

(86, 71)

## 6.4.面向过程和面向对象

面向过程：

以过程为中心的编程思想，以“什么正在发生”为主要目标进行编程。 冰冷的，程序化的

面向对象：

将现实世界的事物抽象成对象，更关注“谁在受影响”，更加贴近现实。  有血有肉，拟人（物）化的

以公共汽车为例

“面向过程”：

汽车启动是一个事件，汽车到站是另一个事件。。。。

在编程序的时候我们关心的是某一个事件，而不是汽车本身。

我们分别对启动和到站编写程序。

"面向对象"：

构造“汽车”这个对象。

对象包含动力、服役时间、生产厂家等等一系列的“属性”；

也包含加油、启动、加速、刹车、拐弯、鸣喇叭、到站、维修等一系列的“方法”。

通过对象的行为表达相应的事件

## 6.5.作业练习

函数的定义及调用：

1、判断题：

\*函数定义时，默认参数一定要放在非默认参数的后面。

\*函数调用时，关键字参数一定要放在位置参数的后面。

\*下列代码执行是否会报错，如不报错，输出何值？

def func(x, \*y, \*\*z):

print(x)

print(y)

print(z)

ls = ["a", "b", "c"]

d = {"name": "Sarah", "age":18}

func(\*ls, "d", \*\*d)

函数式编程：

2、试对本章函数式编程的例子进行如下修改：

\*对getinputs函数增加容错，当输入的单球获胜概率不在（0，1）区间内时，提示输入错误，重新输入。

\*将每局比赛的获胜规则更改为：一方得分大于等于21分，且与对手分差至少为2分，该种情形第一次出现时，得分高者获得该局比赛的胜率。

\*将每场比赛获胜规则更改为：三局两胜，率先获得两局胜利者，获得该场比赛胜利。

\*其他条件于本章例题相同，试模拟1万场比赛的结果。

匿名函数：

3、统计下列绕口令中字符出现的频次，并按照频次的降序对字符进行排列：

八百标兵奔北坡，

北坡八百炮兵炮。

标兵怕碰炮兵炮，

炮兵怕把标兵碰。

答案：

1. \* True

\* True

\* a

('b', 'c', 'd')

{'name': 'Sarah', 'age': 18}

2.

**import** **random**

**def** get\_inputs():

*# 输入原始数据*

**while** **True**:

prob\_A = eval(input("请输入运动员A的每球获胜概率(0~1)："))

**if** prob\_A > 0 **and** prob\_A < 1:

**break**

prob\_B = round(1-prob\_A, 2)

number\_of\_games = eval(input("请输入模拟的场次（正整数）："))

print("模拟比赛总次数：", number\_of\_games)

print("A 选手每球获胜概率：", prob\_A)

print("B 选手每球获胜概率：", prob\_B)

**return** prob\_A, prob\_B, number\_of\_games

**def** game\_over(score\_A, score\_B):

*# 单场模拟结束条件，一方得分大于等于21分，且与对手分差至少为2分*

**return** (score\_A == 21 **or** score\_B == 21) **and** (abs(score\_A - score\_B) >= 2)

**def** sim\_one\_game(prob\_A, prob\_B):

*# 模拟一场比赛的结果*

cnt\_A, cnt\_B = 0, 0

score\_A, score\_B = 0, 0

flag = **True**

**for** i **in** range(3):

**if** flag == **False**:

**break**

**while** **not** game\_over(score\_A, score\_B):

**if** random.random() < prob\_A: *# random.random() 生产[0,1)之间的随机小数,均匀分布*

score\_A += 1

cnt\_A += 1

**else**:

score\_B += 1

cnt\_B += 1

**if** cnt\_A == 2 **or** cnt\_B == 2:

flag = **False**

**break**

**return** score\_A, score\_B

**def** sim\_n\_games(prob\_A, prob\_B, number\_of\_games):

*# 模拟多场比赛的结果*

win\_A, win\_B = 0, 0 *# 初始化A、B获胜的场次*

**for** i **in** range(number\_of\_games): *# 迭代number\_of\_games次*

score\_A, score\_B = sim\_one\_game(prob\_A, prob\_B) *# 获得模拟依次比赛的比分*

**if** score\_A > score\_B:

win\_A += 1

**else**:

win\_B += 1

**return** win\_A, win\_B

**def** print\_summary(win\_A, win\_B, number\_of\_games):

*# 结果汇总输出*

print("共模拟了**{}**场比赛".format(number\_of\_games))

print("**\033**[31m选手A获胜**{0}**场，占比**{1:.1%}**".format(win\_A, win\_A/number\_of\_games))

print("选手B获胜**{0}**场，占比**{1:.1%}**".format(win\_B, win\_B/number\_of\_games))

**def** main():

*# 主要逻辑*

prob\_A, prob\_B, number\_of\_games = get\_inputs() *# 获取原始数据*

win\_A, win\_B = sim\_n\_games(prob\_A, prob\_B, number\_of\_games) *# 获取模拟结果*

print\_summary(win\_A, win\_B, number\_of\_games) *# 结果汇总输出*

**if** \_\_name\_\_ == "\_\_main\_\_":

main()

请输入运动员A的每球获胜概率(0~1)：0.55

请输入模拟的场次（正整数）：10000

模拟比赛总次数： 10000

A 选手每球获胜概率： 0.55

B 选手每球获胜概率： 0.45

共模拟了10000场比赛

选手A获胜5745场，占比57.5%

选手B获胜4255场，占比42.5%

3.

s = "八百标兵奔北坡，北坡八百炮兵炮。标兵怕碰炮兵炮，炮兵怕把标兵碰。"

d = {}

**for** i **in** s:

d[i] = d.get(i, 0) + 1

print(d)

{'八': 2, '百': 2, '标': 3, '兵': 6, '奔': 1, '北': 2, '坡': 2, '，': 2, '炮': 5, '。': 2, '怕': 2, '碰': 2, '把': 1}

print(sorted(d.items(), key = **lambda** item : item[1], reverse=**True**))

[('兵', 6), ('炮', 5), ('标', 3), ('八', 2), ('百', 2), ('北', 2), ('坡', 2), ('，', 2), ('。', 2), ('怕', 2), ('碰', 2), ('奔', 1), ('把', 1)]