《大数据分析导论》——函数与代码复用

Python

提纲

- 函数的基本使用
- 函数的参数传递
- 函数递归

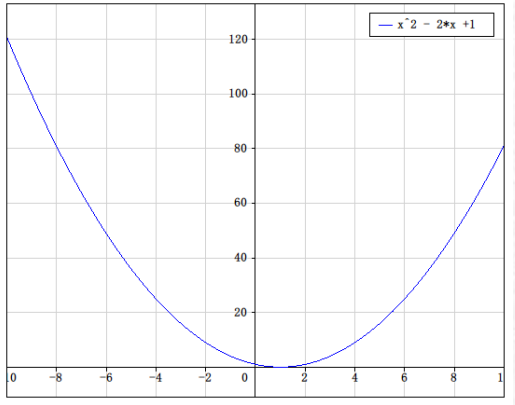
函数的定义

- 函数是一段具有特定功能的、可重用的语句组
  - 用函数名来表示并通过函数名进行完成功能调用
  - 输入参数传递
  - 返回输出
- 一种功能抽象：黑盒

函数名　参数

function(x) =

$$x^2 - 2x + 1$$



函数的定义

- 作用
  - 问题分解，降低编程难度
  - 代码复用
- Python自带函数
  - 内置函数
  - Python标准库中的函数

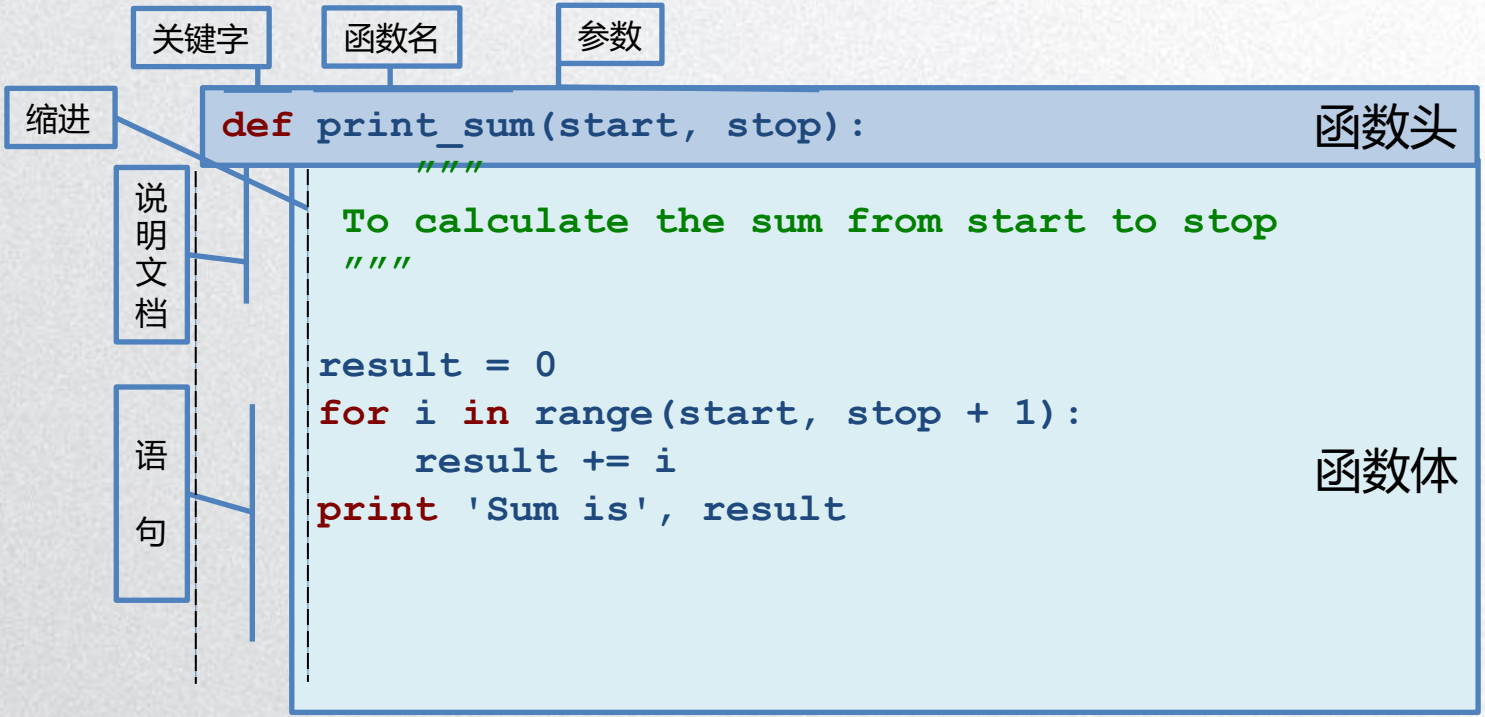
- 自定义函数：使用def保留字

```
def <函数名>(<参数列表>):
        <函数体>
        return <返回值列表>
```

```
import random
import math

DARTS=100000
hits=0.0

random.seed()
for i in range(1, DARTS+1):
    x,y = random.random(),random.random()
    dist = math.sqrt(x**2+y**2)
    if dist<=1.0:
        hits = hits + 1

pi= 4 * (hits / DARTS)
print("Pi值是{0}.".format(pi))
```

```
Pi值是3.13684.
```

定义函数


关键字
函数名
参数
缩进
说明文档
语句
def print_sum(start, stop):
"""
To calculate the sum from start to stop
"""
result = 0
for i in range(start, stop + 1):
result += i
print 'Sum is', result
函数头
函数体

定义和调用函数

- 定义函数

```
def print_sum(start, stop):
    result = 0
    for i in range(start, stop + 1):
        result += i
    print('Sum is', result)
```

形式参数（形参，parameter）

- 调用函数

```
print_sum(1, 10)
```

实际参数（实参，argument）

示例：生日歌

- 生日歌：歌词
  - Happy birthday to you!
  - Happy birthday to you!
  - Happy birthday, dear <名字>!
  - Happy birthday to you!
- 编写程序为Mike和Lily输出生日歌歌词

```
In [1]: print("Happy birthday to you!")
        print("Happy birthday to you!")
        print("Happy birthday, dear Mike!")
        print("Happy birthday to you!")

Happy birthday to you!
Happy birthday to you!
Happy birthday, dear Mike!
Happy birthday to you!
```

示例：生日歌

- 定义函数HappyB()
  - 括号中<名字>形参
  - 调用两次

In [2]:

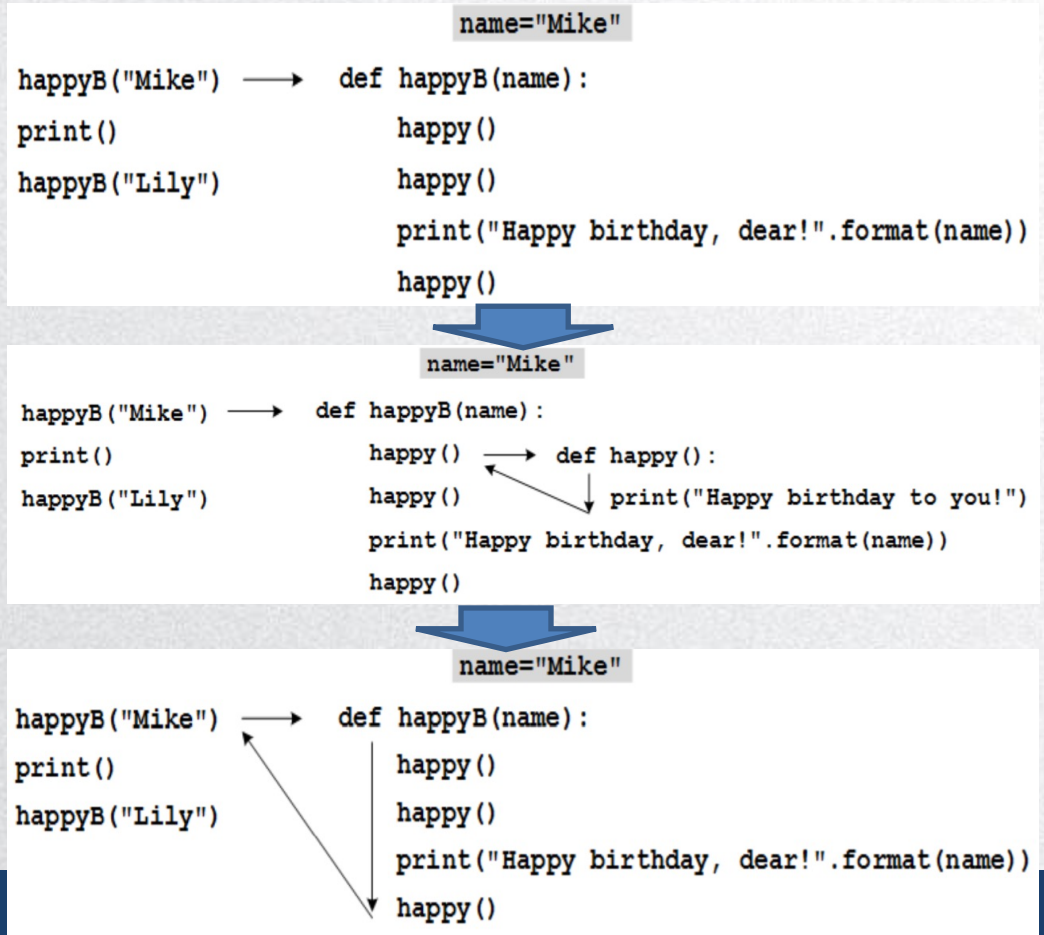
```
def happy():
    print("Happy birthday to you!")
def happyB(name):
    happy()
    happy()
    print("Happy birthday, dear {}!".format(name))
    happy()

happyB("Mike")
print()
happyB("Lily")
```

```
Happy birthday to you!
Happy birthday to you!
Happy birthday, dear Mike!
Happy birthday to you!

Happy birthday to you!
Happy birthday to you!
Happy birthday, dear Lily!
Happy birthday to you!
```

函数调用过程

- 四个步骤：
  - （1）调用程序在调用处暂停执行
  - （2）在调用时将实参复制给函数的形参
  - （3）执行函数体语句
  - （4）函数调用结束给出返回值
- 程序回到调用前的暂停处继续执行

函数调用过程


name="Mike"
happyB("Mike")
print()
happyB("Lily")
def happyB(name):
happy()
happy()
print("Happy birthday, dear!".format(name))
happy()
name="Mike"
happyB("Mike")
print()
happyB("Lily")
def happyB(name):
happy()
happy()
print("Happy birthday, dear!".format(name))
happy()
def happy():
print("Happy birthday to you!")
name="Mike"
happyB("Mike")
print()
happyB("Lily")
def happyB(name):
happy()
happy()
print("Happy birthday, dear!".format(name))
happy()

单选题 1分

设置

下列Python程序定义f1()时还没有定义f2()，这种函数调用是否合法？

```
In [8]: def f1():
            f2()
        def f2():
            print("This is function f2()")
        f1()
```

A 合法

B 不合法

提交

Python

提纲

- 函数的基本使用
- **函数的参数传递**
- 函数递归

可选参数

- 在定义函数时，有些参数可以存在默认值
- 函数被调用时，如果没有传入对应的参数值，则使用函数定义时的默认值代替
- 可选参数必须定义在非可选参数后面

```
In [6]: def dup(str, times=2):
            print(str*times)
        dup("ruc~")

ruc~ruc~

In [7]: dup("ruc!",4)

ruc!ruc!ruc!ruc!
```

可变数量参数

- 在函数定义时，可以设计可变数量参数，通过参数前增加星号（*）实现
- 可变参数只能出现在参数列表的最后
  - 可变参数被当作元组类型传入函数中

```
In [9]: def vfunc(a, *b):
            print(type(b))
            for n in b:
                a += n
            return a
        vfunc(1,2,3,4,5)

        <class 'tuple'>

Out[9]: 15
```

参数的位置和名称传递

- 实参默认采用按照位置顺序的方式传递给函数
  - *def func(x1,y1,z1,x2,y2,z2)*
  - *func(1,2,3,4,5,6)*

- Python提供了按照形参名称输入实参的方式，调用如下：
  - *result = func(x2=4, y2=5, z2=6, x1=1, y1=2, z1=3)*

- 由于调用函数时指定了参数名称，所以参数之间的顺序可以任意调整

函数的返回值

- return语句用来退出函数并将程序返回到函数被调用的位置继续执行
- return语句同时可以将0个、1个或多个函数运算完的结果返回给函数被调用处的变量
  - 用return返回多个值，多个值以元组类型保存
- 函数可以没有return，此时函数并不返回值

```
In [10]: def func(a,b):
             return a*b
         s = func("ruc!",2)
         print(s)

         ruc!ruc!
```

```
In [11]: def func(a,b):
             return b, a
         s = func("ruc!",2)
         print(s, type(s))

         (2, 'ruc!') <class 'tuple'>
```

函数对变量的作用

- 一个程序中的变量包括两类：全局变量和局部变量
  - 全局变量指在函数之外定义的变量，一般没有缩进，在程序执行全过程有效
  - 局部变量指在函数内部使用的变量，仅在函数内部有效当函数退出时变量将不存在

In [12]:
```
n = 1 #n是全局变量
def func(a, b):
    c = a * b #c是局部变量，a和b作为函数参数也是局部变量
    return c
s = func("ruc!", 2)
print(c)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-07b21183da77> in <module>
      4     return c
      5 s = func("ruc!", 2)
----> 6 print(c)

NameError: name 'c' is not defined
```

函数对变量的作用

- 函数内部使用全局变量

In [13]:
```
n = 1 #n是全局变量
def func(a, b):
    n = b #这个n是在函数内存中新生成的局部变量，不是全局变量
    return a*b
s = func("ruc!", 2)
print(s, n) #测试一下n值是否改变
```

```
ruc!ruc! 1
```

- 这里n被函数理解为一个局部变量，函数退出后释放

函数对变量的作用

- 如果全局变量的类型为组合数据类型，比如列表类型：

In [15]:
```
ls = [] #ls是全局列表变量
def func(a, b):
    ls.append(b) #将局部变量b增加到全局列表变量ls中
    return a*b
s = func("ruc!", 2)
print(s, ls) #测试一下ls值是否改变
```

```
ruc!ruc! [2]
```

- 全局列表变量在函数调用后发生变化

函数对变量的作用

- 如果func()函数内部存在一个真实创建过且名称为ls的列表，则func()将操作该列表而不会修改全局变量

In [16]:
```
ls = [] #ls是全局列表变量
def func(a, b):
    ls = [] #创建了名称为ls的局部列表变量列
    ls.append(b) #将局部变量b增加到全局列表变量ls中
    return a*b
s = func("ruc!", 3)
print(s, ls) #测试一下ls值是否改变
```

```
ruc!ruc!ruc! []
```

Python

提纲

- 函数的基本使用
- 函数的参数传递
- **函数递归**

递归的定义

- 函数作为一种代码封装，也可以被函数内部代码调用。
  - 这种函数定义中**调用函数自身**的方式称为递归。

- 数学上有个经典的递归例子叫阶乘，阶乘通常定义为：

$$n! = n(n-1)(n-2)...(1)$$

$$n! = \begin{cases} 1 & n = 0 \\ n(n-1)!otherwise \end{cases}$$

- 递归的两个关键特征：
  - 存在一个或多个基例，基例不需要再次递归（递归出口）
  - 所有递归链要以一个或多个基例结尾。

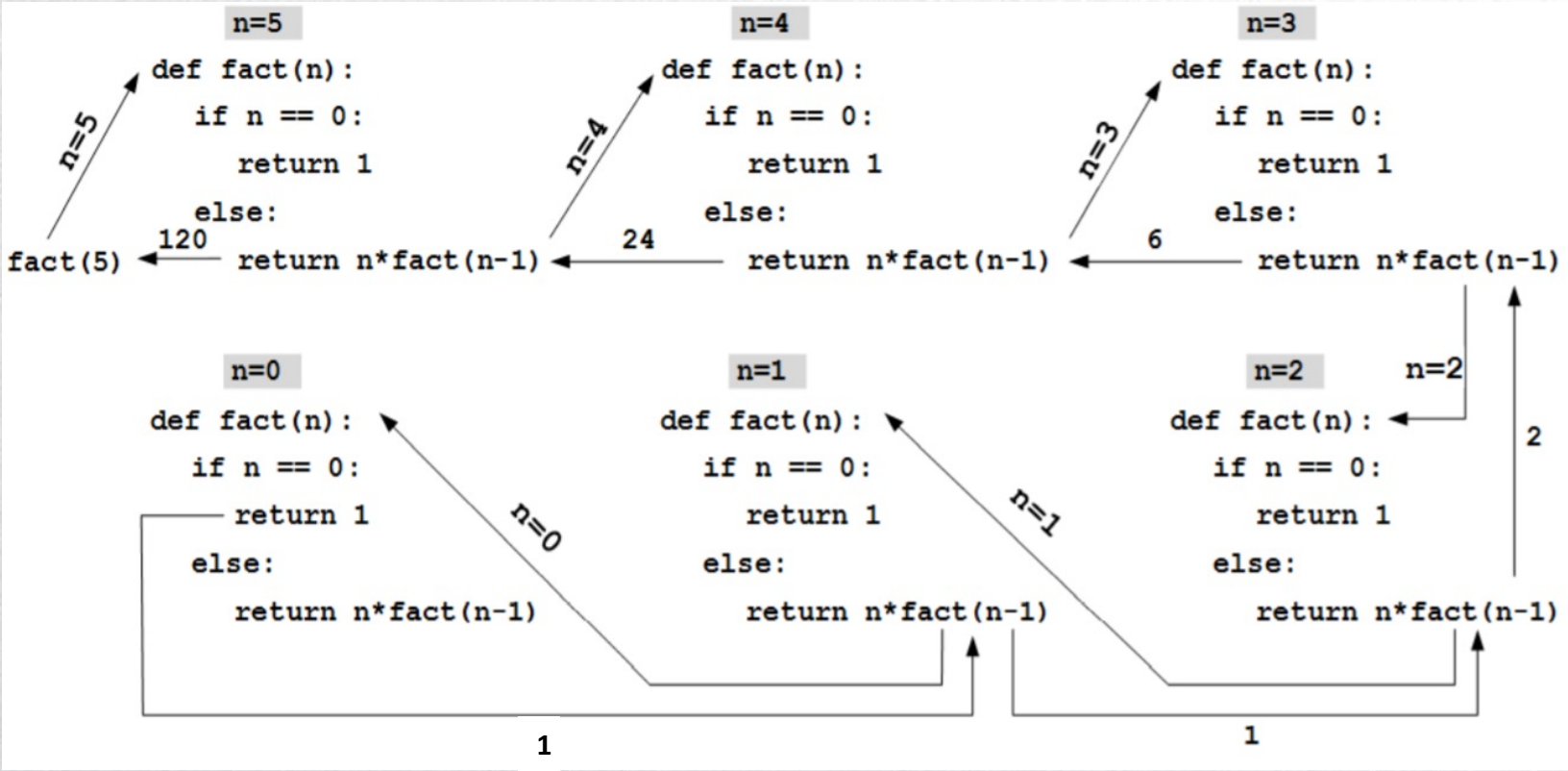
递归的使用方法

- 阶乘计算：用户输入整数n，计算并输出n的阶乘值

In [2]:
```
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n-1)
num = eval(input("请输入一个整数: "))
print(fact(abs(int(num))))
```

```
请输入一个整数: 10
3628800
```

递归的使用方法


n=5
def fact(n):
if n == 0:
return 1
else:
return n*fact(n-1)
n=5
fact(5)
120
n=4
def fact(n):
if n == 0:
return 1
else:
return n*fact(n-1)
n=4
24
n=3
def fact(n):
if n == 0:
return 1
else:
return n*fact(n-1)
n=3
6
n=2
def fact(n):
if n == 0:
return 1
else:
return n*fact(n-1)
n=2
2
n=1
def fact(n):
if n == 0:
return 1
else:
return n*fact(n-1)
n=1
1
n=0
def fact(n):
if n == 0:
return 1
else:
return n*fact(n-1)
n=0
1

递归的使用方法

- 字符串反转：对于用户输入的字符串s，输出反转后的字符串。
- 解决这个问题的基本思想是把字符串看作一个递归对象。

```
In [3]: def reverse(s):
            return reverse(s[1:]) + s[0]
        reverse("ABC")
```

```
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
<ipython-input-3-9f2c1408d503> in <module>
      1 def reverse(s):
      2     return reverse(s[1:]) + s[0]
----> 3 reverse("ABC")

<ipython-input-3-9f2c1408d503> in reverse(s)
      1 def reverse(s):
----> 2     return reverse(s[1:]) + s[0]
      3 reverse("ABC")

... last 1 frames repeated, from the frame below ...

<ipython-input-3-9f2c1408d503> in reverse(s)
      1 def reverse(s):
----> 2     return reverse(s[1:]) + s[0]
      3 reverse("ABC")

RecursionError: maximum recursion depth exceeded
```

- 原因：reverse函数没有基例
- 为防止无限递归，Python设置了默认的最大递归深度

递归的使用方法

- 完整代码

In [1]:
```
def reverse(s):
    if s=="":
        return s
    else:
        return reverse(s[1:]) + s[0]

str = input("请输入一个字符串：")
print(reverse(str))
```

```
请输入一个字符串：高瓠人工智能学院
院学能智工人瓠高
```

另一种实现思路

- 反转一个字符串，如：abcdef
  - 第一个字符和最后一个字符调换，f bcde a
  - 中间剩下的字符串再反转：f edcb a

```
def reverse(s):
    if s == "" or len(s) == 1:
        return s
    else:
        return s[-1] + reverse(s[1:-1]) + s[0]

str = "abcdefg"
print(reverse(str))
```

```
gfedcba
```

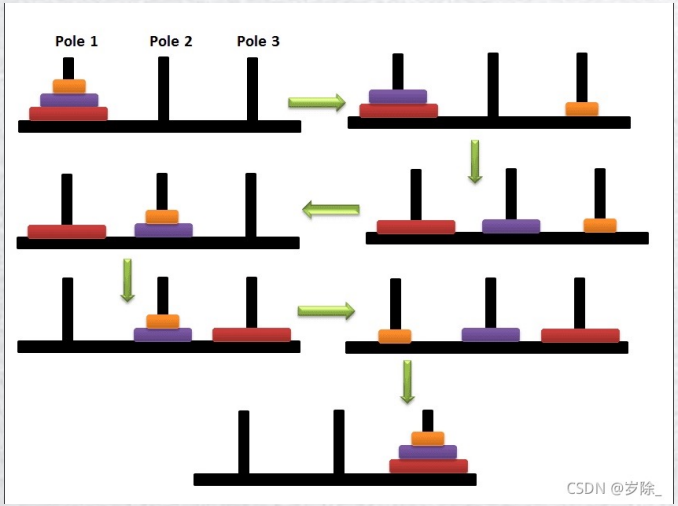
Python内置函数

- Python解释器提供了68个内置函数

abs()	id()	round()	compile()	locals()
all()	input()	set()	dir()	map()
any()	int()	sorted()	exec()	memoryview()
asci()	len()	str()	enumerate()	next()
bin()	list()	tuple()	filter()	object()
bool()	max()	type()	format()	property()
chr()	min()	zip()	frozenset()	repr()
complex()	oct()		getattr()	setattr()
dict()	open()		globals()	slice()
divmod()	ord()	bytes()	hasattr()	staticmethod()
eval()	pow()	delattr()	help()	sum()
float()	print()	bytearray()	isinstance()	super()
hash()	range()	callable()	issubclass()	vars()
hex()	reversed()	classmethod()	iter()	__import()__

作业1：汉诺塔

- 汉诺塔(又称河内塔)问题是源于印度一个古老传说的益智玩具。汉诺塔是在一块铜板装置上，有三根杆(编号A、B、C)，在A杆自下而上、由大到小按顺序放置一定数量的金盘，把A杆上的金盘全部移到C杆上，并仍保持原有顺序叠好。操作规则：每次只能移动一个盘子，并且在移动过程中三根杆上都始终保持大盘在下，小盘在上，操作过程中盘子可以置于A、B、C任一杆上。



作业2：倒转句子的单词顺序

- 倒转一句话中的单词顺序，但保持单词内部的字母顺序

- 举例
  - 输入： “this is a book”
  - 输出： “book a is this”

谢谢！