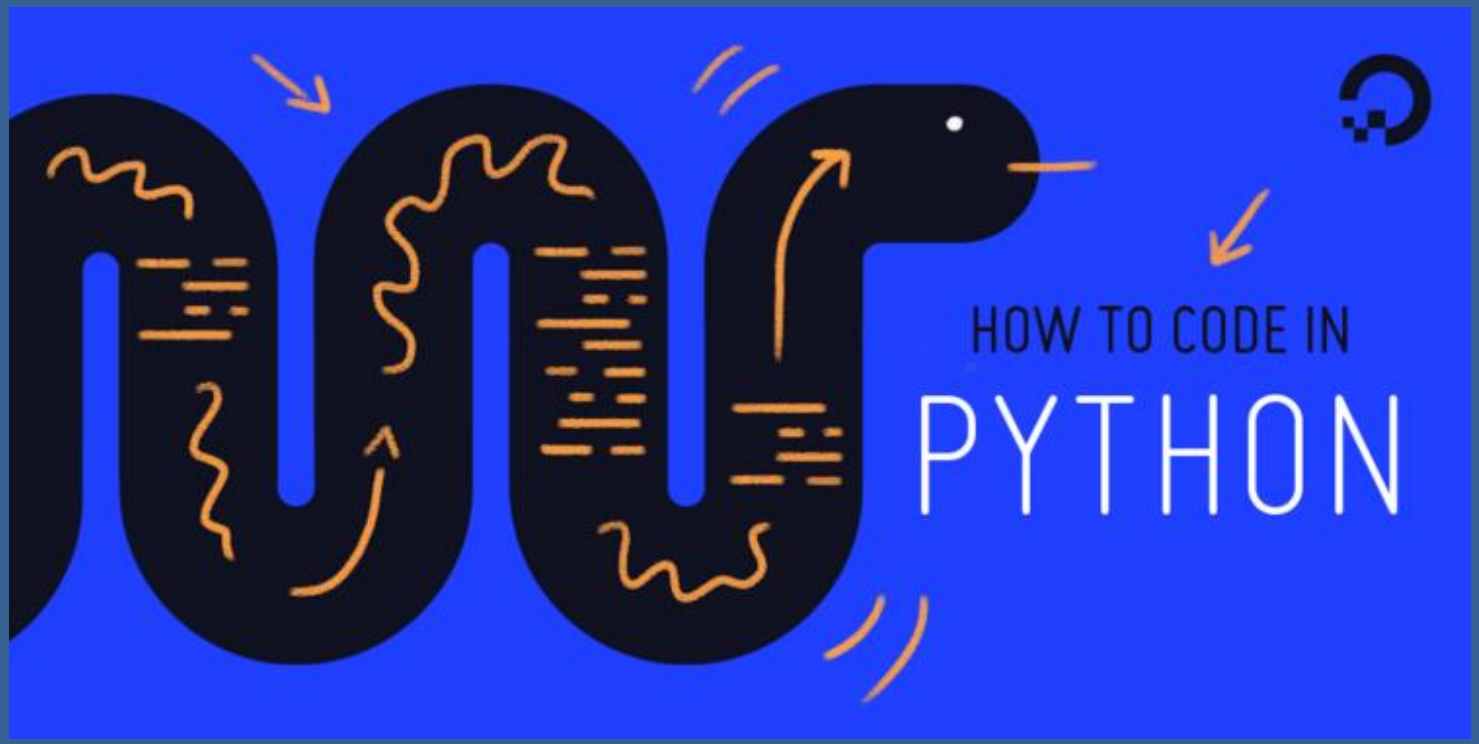


Python第三课
控制结构与函数

张祎璘 经济学院2015级统计系本科

Tips：

- 课前请提前装好Python（Anaconda），安装不成功的可以在课前课后随时提问。

- 课程会分块先讲解语法和例子，然后留给大家充足的时间进行练习。

- 可以复习一下上一节课的Python的数据结构。

控制结构

- 什么是控制结构？
- if else的用法及例子
- for的用法及例子
- while的用法及例子
- Python的语法规则

控制结构

- 什么是控制结构？

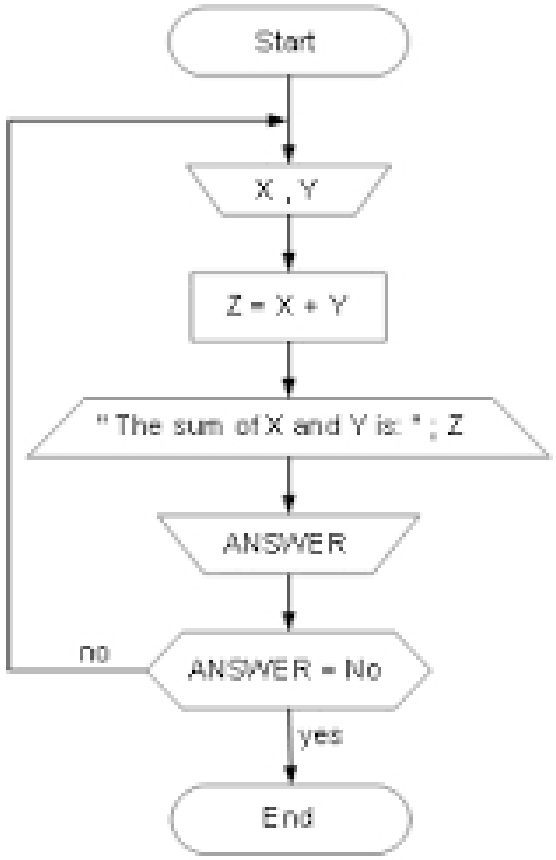
什么是控制结构？

编程语言一般包括三种结构：

- **顺序结构**
- **判断结构（while，if）**
- **循环结构（for）**

Python一般按照次序从头到尾执行文件中嵌套块的语句，但是像if、for等这种语句使得**解释器在程序内跳跃。**

因为Python经过一个程序的路径叫做**控制流程**，像if这类会**对其产生影响的语句就叫做控制结构。**



控制结构

- if else的用法及例子

if else的用法及例子

if else属于判断结构，具体的语法如下：

- 注意**缩进**，statements必须比if/elif /else多缩进一层，以表明他们属于if语句的一部分。
- 不要忘了**冒号**。
- 如果statement只有一行语句可以将其写在冒号后面，但最好不要这样。
- 缩进时可以用**tab**，也可以用**空格（4个），**但在同一段代码里**不要混用这两种方法。**

```
if <test1> :
    <statement1>
elif <test2> :
    <statement2>
elif <test3> :
    <statement3>
  ……
else :
    <statementN>
```

if else的用法及例子

使用的时候可以只使用一个 **if** 语句：

```
a = 1
if a==1:
    print (" You are right")
```

```
In [1]: a = 1
        if a==1:
            print("you are right")
```

```
you are right
```

如果执行语句只有一句，可以和 **if** 写到同一行：

```
if a==1: print (" You are right")
```

经常用到的为if else：

```
if 0:
    print("False")
else:
    print("True")
```

```
In [4]: if 0:
            print("False")
        else:
            print("True")
```

```
True
```

* **if** 后面的0表示**bool**类型，等价于**False**.

if else的用法及例子

if 后面可以加 **not**：

```
if not 0:
    print("False")
else:
    print("True")
```

```
In [5]: if not 0:
            print("False")
        else:
            print("True")
```

```
False
```

有不同条件的情况下，用**elif**：

```
if "a" in "ok":
    print("a")
elif "k" in "ok":
    print("k")
else:
    print("nothing")
```

```
In [6]: if "a" in "ok":
            print("a")
        elif "b" in "ok":
            print("b")
        elif "k" in "ok":
            print("k")
        else:
            print("nothing")
```

```
k
```

if else的用法及例子

if 有多个条件，这些条件为**与**关系时：

```
if 1>-1 and 1<3:
    print("right")
```

```
In [11]: if 1>-1 and 1<3:
             print("right")
```

```
right
```

为**或**关系时：

```
if 's' in 'alpha' or 'e' in 'alpha' or 'h'in 'alpha':
    print("right")
```

```
In [15]: if 's' in 'alpha' or 'e' in 'alpha' or 'h'in 'alpha':
             print("right")
```

```
right
```

```
if (True or
    False):
    print("right")
```

```
In [16]: if (True or
             False):
             print("right")
```

```
right
```

if else的用法及例子

练习一：

小明身高1.75，体重80.5kg。请根据BMI公式（体重除以身高的平方）帮小明计算他的BMI指数，并根据BMI指数：

低于18.5：过轻
18.5-25：正常
25-28：过重
28-32：肥胖
高于32：严重肥胖

用if判断并打印结果

控制结构

- for的用法及例子

for的用法及例子

for属于循环结构，具体的语法如下：

```
for <target> in <object>:
    <statements>
```

- target是变量，object是列表，元组，range或其他的序列类型。若target在object中，则反复执行。Statements，不要忘了**冒号**。
- **break**： 结束整个循环语句，执行循环语句之后的第一条语句。
- **continue**：结束当前一轮的循环，回到循环的首行。
- **pass**：相当于空行，什么也不做。

for的用法及例子

对**字符串**使用for循环：

```
for i in 'cat':
    print(i, end=' ')
```

```
In [22]: for i in 'cat':
             print(i, end=' ')

c a t
```

range函数经常在for循环中使用，生成a到b的整数：

```
for i in range(10):
    print(i,end=' ')
```

```
In [6]: for i in range(10):
            print(i, end=' ')

0 1 2 3 4 5 6 7 8 9
```

```
for i in range(3,6):
    print(i,end=' ')
```

```
In [20]: for i in range(3, 6):
             print(i, end=' ')

3 4 5
```

对**列表**使用for循环：

```
for i in ['s','o','e']:
    print(i,end=',')
```

```
In [25]: for i in ['s','o','e']:
             print(i, end=',')

s, o, e,
```

for的用法及例子

对**字典key**使用for循环：

```
eat = {'cat':'fish', 'dog':'bone', 'rabbit':'carrot'}
for i in eat:
    print(i, end=' ')
```

```
In [27]: eat = {'cat':'fish', 'dog':'bone', 'rabbit':'carrot'}
         for i in eat:
             print(i, end=' ')

cat dog rabbit
```

```
for i in eat.keys():
    print(i, end=' ')
```

```
In [33]: for i in eat.keys():
             print(i, end=' ')

cat dog rabbit
```

对**字典value**使用for循环：

```
for i in eat.values():
    print(i, end=' ')
```

```
In [32]: for i in eat.values():
             print(i, end=' ')

fish bone carrot
```

for的用法及例子

对**字典value**使用for循环：

```
for i in eat:
    print(eat[i], end=' ')
```

```
In [34]: for i in eat:
             print(eat[i], end=' ')

fish bone carrot
```

对**整个字典**使用for循环：

```
for i in eat.items():
    print(i, end=' ')
```

```
In [35]: for i in eat.items():
             print(i, end=' ')

('cat', 'fish') ('dog', 'bone') ('rabbit', 'carrot')
```

```
for key,value in eat.items():
    print(key, 'eat',value, end='    ')
```

```
In [37]: for key,value in eat.items():
             print(key, 'eat',value, end='    ')

cat eat fish    dog eat bone    rabbit eat carrot
```

for的用法及例子

break可以终止整个的for循环：

```
for i in range(10):
    print(i,end=' ')
    if i==5:
        break
```

```
In [54]: for i in range(10):
             print(i, end=' ')
             if i==5:
                 break
```

```
0 1 2 3 4 5
```

continue可以跳过本轮循环：

```
for i in range(10):
    if i==5:
        continue
    print(i,end=' ')
```

```
In [38]: for i in range(10):
             if i==5:
                 continue
             print(i, end=' ')
```

```
0 1 2 3 4 6 7 8 9
```

for的用法及例子

练习二：

请利用**for**依次对list中的每个名字打印出Hello, xxx!：

```
L = ['Duck', 'Cat']
```

控制结构

- while的用法及例子

while的用法及例子

while属于循环结构，具体的语法如下：

```
while <test>:
    <statement1>
```

- 若<test>不等于0，True，非空字符串或数据结构，反复执行statement1。
- **break**： 结束整个循环语句，执行循环语句之后的第一条语句。
- **continue**：结束当前一轮的循环，回到循环的首行。
- **pass**：相当于空行，什么也不做。

while的用法及例子

计算100以内的奇数和：

```
sum = 0
n = 99
while n > 0:
    sum = sum + n
    n = n - 2
print(sum)
```

```
In [1]: sum = 0
        n = 99
        while n > 0:
            sum = sum + n
            n = n - 2
        print(sum)
```

```
2500
```

我们利用continue完成相同的任务：

```
sum = 0; n = 100
while n > 0:
    n = n - 1
    if n%2 ==0:
        continue
    sum = sum + n
```

```
In [4]: sum = 0; n = 100
        while n > 0:
            n = n - 1
            if n%2 ==0:
                continue
            sum = sum + n
        print(sum)
```

```
2500
```

while的用法及例子

当没有条件的时候，经常用break来跳出循环：

```
k = 2
while 1:
    k = k * 2
    if k>1000:
        break
print(k)
```

```
In  [5]: k = 2
         while 1:
             k = k * 2
             if k>1000:
                 break
         print(k)

         1024
```

注意一定不要写死循环！！！

```
s = 1
while s==1:
    print(1,end=)
```

```
In  [*]: s = 1
         while s==1:
             print(1,end=' ')

         1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
         1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
         1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

while的用法及例子

while和for都为循环结构，那它们有什么区别吗？或者在什么情况下适合使用？

while的用法及例子

while是有条件但不知道循环次数的条件下使用。
for则是知道循环次数的条件下使用。

但实际上，对于大多数问题两个基本都可以使用。

```
 L = ['egg', 'and', 'rice']

for i in L:
    print(i)

i = 0
while(i<len(L)):
    print(L[i])
    i = i+1
```

In [19]:
```
L = ['egg', 'and', 'rice']
for i in L:
    print(i)

i = 0
while(i<len(L)):
    print(L[i])
    i = i+1
```

```
egg
and
rice
egg
and
rice
```

while的用法及例子

练习三：

请利用**while**依次对list中的每个名字打印出Hello, xxx!：

```
L = ['Duck', 'Cat']
```

控制结构

- Python的语法规则

Python的语法规则

块和语句的边界会自动检测：python没有大括号或者“begin/end”等字符，一般也不以分号终止。Python使用首行下的语句缩进把嵌套块内的语句组合起来。

文件的空白行、空格、注释都会忽略：注释以#开头，或为"""　"""代码块。

整齐的代码风格：Python因为通过缩进组织代码，所以对缩进有严格要求。整个代码通常看起来非常整齐。

不要混合使用tabs和空格：尽管两者都可以使用，但是一旦混用，会导致在不同的编辑器上呈现不同的缩进效果，并且难以修改。

Python函数

- 为什么使用函数？
- 调用函数
- 定义函数
- 函数的参数
- 高阶函数

Python函数

- 为什么使用函数？

为什么使用函数？

之前我们学习了Python中一些简单的流程语句。为了更好的组织这些语句，我们学习如何**调用以及创建函数**。

简而言之一个**函数就是将一些语句集合在一起的部件**，它们能不止一次在程序中运行。并且输入不同的参数，计算出不同的返回值。以函数的形式去编写一个操作可以使它成为一个能够**广泛运行的工具**。

更具体的说，**函数是在编程中剪剪贴贴的替代**。我们可以大大减少代码中的冗余的部分，修改其中的一小部分，

总而言之，函数是**Python为了代码最大程度的重用和最小化代码冗余而提供的最基本的程序结构。**

函数是一种设计工具。

Python函数

- 调用函数

调用函数

Python本身有大量的**内置函数**，便于最基本的操作。这也是应当熟悉并且掌握的。下面给一些基本的例子。

```
abs(-1)       # 求绝对值
max(2,3)      # 求最大值
pow(3,2)      # 乘方
len([1,2,3])  # 求列表，字典，元组等长度
list({1,2})   # 转化为列表
float(1)      # 转化为浮点数
help(min)     # 函数的帮助文档
```

```
In [2]: print(abs(-1))          # 求绝对值
        print(max(2,3))         # 求最大值
        print(pow(3,2))         # 乘方
        print(len([1,2,3]))     # 求列表，字典，元组等长度
        print(list({1,2}))      # 转化为列表
        print(float(1))         # 转化为浮点数
```

```
1
3
9
3
[1, 2]
1.0
```

调用函数

所有的内置函数见：
https://docs.python.org/3/library/functions.html

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

调用函数

除了内置函数，我们也调用库中的函数。在调用库中的函数前需要**首先导入库**。以Python常用的时间相关的库time为例：

```
import time
# 调用为 time.函数名( )
print(time.time())
#自从1970年1月1日午夜经过了多少秒
print(time.localtime(time.time()))
# 打印当地时间
```

```
import time

print(time.time())
print(time.localtime(time.time()))
```

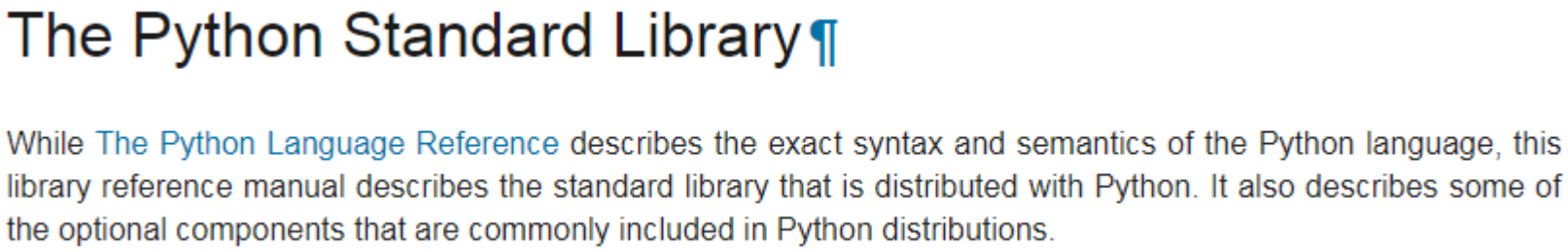
```
1537535064.0985684
time.struct_time(tm_year=2018, tm_mon=9, tm_mday=21, tm_hour=21, tm_min=4, tm_sec=24
```

调用函数

Python包含了大量的标准库，涵盖了包括：**数学(math)，系统操作(os)，日期和时间(datetime)，字符串正则匹配(re)**等方方面面的库。

这些库**无需安装，直接调用**即可。对于想熟练运用Python的同学，可以多了解这些库以及其中常用的函数。

官方文档见：https://docs.python.org/3/library/


The Python Standard Library¶

While The Python Language Reference describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python函数

- 定义函数

定义函数

更多的情况下，我们需要**自己定义函数**。Python定义函数的语法如下：

```
def <函数名>(<形参列表>):
    <函数体>
    return <返回值>
```

- 形参列表由变量组成，之间用逗号分隔。
- 函数体为普通的python语句，但比def增加一层缩进。
- return语句终止当前函数的执行，return后的语句将被忽略 。

我们先来定义一个**没有参数和返回值**的简单函数：

```
def my_function():
    print("Hello, World!")
my_function()
```

```
In [2]: def my_function():
            print("Hello, World!")
        my_function()

Hello, World!
```

定义函数

下来我们尝试**有参数和返回值**的函数：

```
def times(x,y):
    return x*y
times(7,6)
```

```
In  [3]: def times(x,y):
             return x*y
         times(7,6)

Out[3]: 42
```

- 这是我们自己定义的乘法函数。
- 注意到这里return**直接返回的是表达式，没有中间变量。**

这样可以**减少分配存储空间**的时间，加快运行速度。

我们可以在函数里面**使用for循环，要注意缩进**：

```
def sum_num(a, b):
    s=0
    for i in range(a,b+1):
        s+=i
    return s
print(sum_num(1,100))
```

```
In  [4]: def sum_num(a, b):
             s=0
             for i in range(a,b+1):
                 s+=i
             return s
         print(sum_num(1,100))

5050
```

定义函数

有时候我们要**返回多个值**。针对这种情况，我们可以在return时，直接用**逗号**把返回值隔开。这样返回的是**tuple**。例如：从一个点移动到另一个点，给出坐标、位移和角度，就可以计算出新的新的坐标：

```
import math
def move(x, y, s, angle=0):
    nx = x + s*math.cos(angle)
    ny = y - s*math.sin(angle)
    return nx, ny
x, y = move(10, 10, 60, math.pi / 6)
print(x, y)
```

```
61.96152422706632 -19.999999999999996
```

```
move(10, 10, 60, m.pi / 6)
```

```
Out[6]: (61.96152422706632, -19.999999999999996)
```

定义函数

当然更多的情况下，我们会把**所有的返回值放在list**里面，一次返回整个**list**。（或者是**dict**）

```
food =['bread','milk','coffee','cake','chocolate','grape']
bag = ['hat','milk','cake','phone','chocolate']
def find_food(food,bag):
    my_food = []
    for i in bag:
        if i in food:
            my_food.append(i)
    return my_food
find_food(food,bag)
```

```
def find_food(food,bag):
    my_food = []
    for i in bag:
        if i in food:
            my_food.append(i)
    return my_food
find_food(food,bag)
```

```
['milk', 'cake', 'chocolate']
```

定义函数

练习四：

请自己写一个绝对值函数。等效于abs的功能。

Python函数

- 函数的参数

函数的参数

我们可以通过添加函数的参数来给函数添加功能。例如原函数是x的平方：

```
def power(x):
    return x * x
power(5)
```

```
In  [9]: def power(x):
             return x * x
         power(5)

Out[9]: 25
```

现在我们想改为求x的n次方：

```
def power(x, n):
    s = 1
    while n > 0:
        n = n - 1
        s = s * x
    return s
power(5,5)
```

```
In  [10]: def power(x, n):
              s = 1
              while n > 0:
                  n = n - 1
                  s = s * x
              return s
          power(5,5)

Out[10]: 3125
```

函数的参数

对于函数，我们可以添加**默认参数**。意思是在没有这个参数输入时，**函数选择默认的参数值进行计算**。例如，刚才的例子当中，我们默认求x的平方。

```
def power(x, n=2):
    s = 1
    while n > 0:
        n = n - 1
        s = s * x
    return s
power(5)
```

```
In [11]: def power(x, n=2):
             s = 1
             while n > 0:
                 n = n - 1
                 s = s * x
             return s
         power(5)

Out[11]: 25
```

为了确保函数的参数接收到对应的值，可以加上参数名称。

```
power(x=5, n=7)
```

* 默认参数必须在在普通参数之后定义。

函数的参数

对于**参数的数目不确定**的情况，我们可以使用**可变参数**进行处理。
在参数前面加*表示把所有的参数看成**tuple**。

```
def change (*s):
    print(s)
change(1,2,3)
```

```
In  [12]: def change (*s):
              print(s)
          change(1,2,3)

          (1, 2, 3)
```

任意个数字求和：

```
def sum_num(*s):
    total=0
    for i in s:
        total+=i
    return total
sum_num(1,2,3)
```

```
In  [13]: def sum_num(*s):
              total=0
              for i in s:
                  total+=i
              return total
          sum_num(1,2,3)

Out[13]: 6
```

函数的参数

除了接收**tuple**，也可以接收**dict**，我们需要在参数前面加**。

```
def person(name, age, **kw):
    print('name:', name, 'age:', age, 'other:', kw)
person('Michael', 30)
```

```
def person(name, age, **kw):
    print('name:', name, 'age:', age, 'other:', kw)
person('Michael', 30)
```

```
name: Michael age: 30 other: {}
```

```
person('Bob', 35, city='Beijing', job='docotor')
```

```
person('Bob', 35, city='Beijing',job='docotor')
```

```
name: Bob age: 35 other: {'city': 'Beijing', 'job': 'docotor'}
```

函数的参数

练习五：

请写一个接受任意个参数的字符串，并且将其连接起来打印的函数。

```
w1 = 'My '
w2 = 'cats '
w3 = 'are '
w4 = 'fighting.'
def my_function(*arg):
    pass
```

Python函数

- 高阶函数

高阶函数

我们在这里浅要介绍一些**高阶函数**的内容。这个概念在**函数式编程**中进一步拓展。
我们通过一些例子来理解高阶函数。

- **变量可以指向函数**

```
abs(-10)
abs
```

```
abs(-10)
```

```
10
```

```
abs
```

```
<function abs>
```

我们可以看到abs是函数本身。
更确切的说，**它是一段代码的名字。**
因此，我们可以将将**这串代码赋予其他的名字。**

```
f = abs
f
f(-10)
```

```
f = abs
f
```

```
<function abs>
```

```
f(-10)
```

```
10
```

这里变量**f指向了abs的函数。**

高阶函数

- **函数名也是变量**

```
def add(x, y, f):
    return f(x) + f(y)
add(-5, 6, abs)
```

```
def add(x, y, f):
    return f(x) + f(y)
```

```
print(add(-5, 6, abs))
```

这里abs作为了参数传入函数中进行使用。

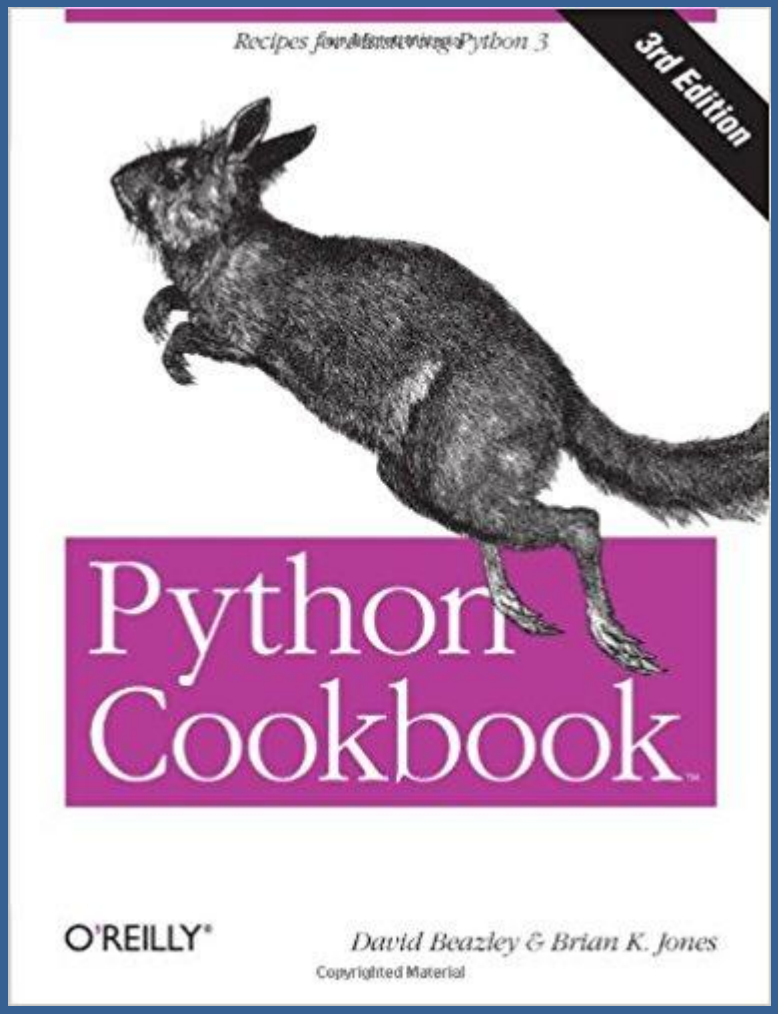
通过上面两个例子，我们发现函数名本身也是一个变量。
它可以将整个函数传给其他的变量。

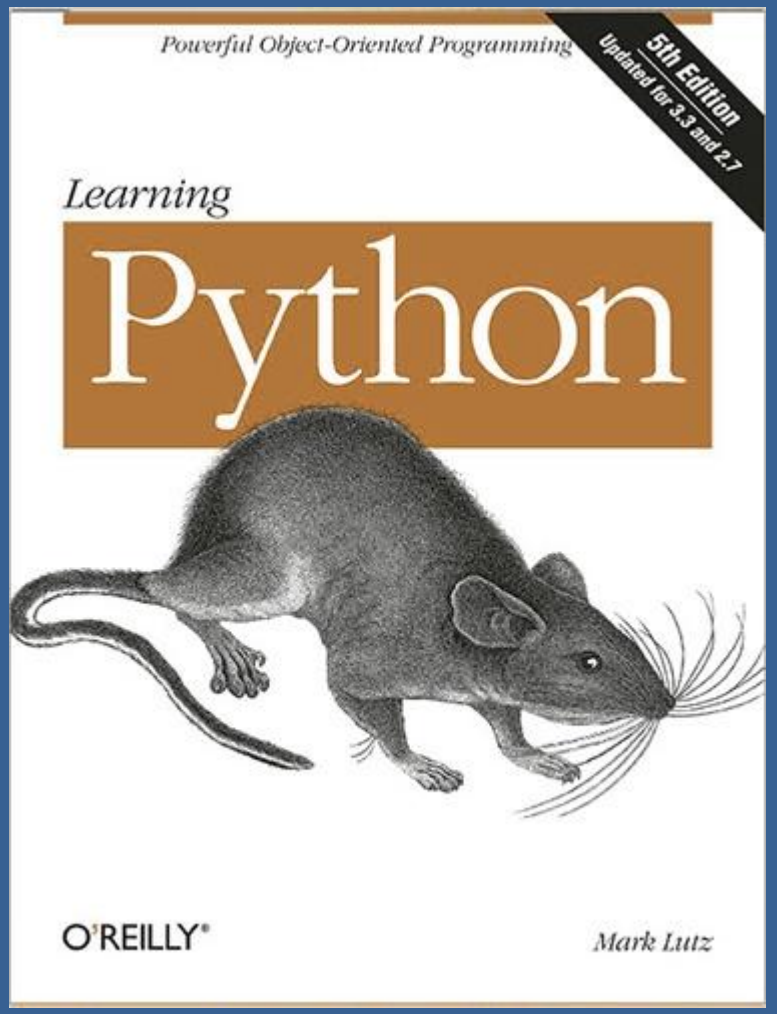
而高阶函数，就是让函数的参数能够接收别的函数。

Py简要回顾：

控制结构
- 什么是控制结构？
- if else的用法及例子
- for的用法及例子
- while的用法及例子
- Python的语法规则

Python函数
- 为什么使用函数？
- 调用函数
- 定义函数
- 函数的参数
- 高阶函数


Recipes for Mastering Python 3
3rd Edition
Python Cookbook
O'REILLY
David Beazley & Brian K. Jones
Copyrighted Material


Powerful Object-Oriented Programming
5th Edition
Updated for 3.3 and 2.7
Learning
Python
O'REILLY
Mark Lutz

introduction to Python

谢谢大家！