Python

1 什么是函数

```
# 函数调用s=area_of_circle(x)
```

2 调用函数

要调用一个函数，需要知道函数的名称和参数

调用abs函数

```
abs(-20)
```

如果函数传入参数数量不是一个或格式不正确都会报错。

而比较函数cmp(x,y)就需要两个参数，如果x¡y,返回-1，如果x==y,返回0,如果x¿y,返回1

Python的内置常用函数还包括数据类型转换函数，比如int()函数可以把其他数据类型转换为整数，str()可以把其他类型转换成str

3 编写函数

定义一个函数要用def语句，依次写出函数名，括号，括号中的参数和冒号：，然后在缩进块中编写函数体，函数的返回值用return语句返回。

#我们以自定义一个求绝对值的my_abs函数为例：

```
def my_abs(x):
    if x>=0:
        return x
    else:
        return -x
```

请注意，函数体内部的语句在执行时，一旦执行到return时，函数就执行完毕，并将结果返回。因此，函数内部通过条件判断和循环可以实现非常复杂的逻辑。

4 返回多值

```
# #math包提供了sin()和cos()函数，我们先用import引用它
import math
def move(x,y,step,angle):
    nx=x+step*math.cos(angle)
    ny=y-step*math.sin(angle)
    return nx,ny
```

Python函数返回的仍然是单一值，是一个tuple，在语法上，返回一个tuple可以省略括号，而多个变量可以同时接受一个tuple，按位置赋给对应的值，所以，Pathon的函数返回多值其实就是返回一个tuple，但写起来更方便。

5 递归函数

在函数内部，可以调用其它函数。如果一个函数在内部调用自身本身，这个函数就是递归函数。

```
# fact(n)用递归的方式写出来就是:
def fact(n):
    if n==1:
        return 1
    return n*fact(n-1)
```

6 定义默认参数

定义函数的时候，还可以有默认参数。

例如python自带的int()函数，其实就有两个参数，我们既可以传一个参数，又可以传两个参数，int的第二个参数是转换进制，如果不传，默认是十进制。

我们来定义一个计算x的N次方的函数:

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

由于函数的参数按从左到右的顺序匹配，所以默认参数只能定义在必须参数的后面。

7 可变参数

如果想让一个函数能接受任意个参数，我们就可以定义一个可变参数:

```
def fn(*args):
    print args
```

可变参数的名字前面有个*号，我们可以传入0个，1个或多个参数给可变参数:

```
fn()
()
fn('a')
('a',)
fn('a','b')
('a','b')
fn('a','b','c')
('a','b','c')
```

Python解释器会把传入的一组参数组装成一个tuple传递给可变参数，因此，在函数内部，直接把变量args看成一个tuple就好了。

8 对list进行切片

取前三个元素，用一行代码就可以完成切片:

```
L[0:3]
#表示从索引0开始取，直到索引3为止。但不包括索引3.即索引0,1,2，正好3个元素。
#如果第一个索引是0，还可以省略
L[:3]
#也可以从索引1开始，取出2个元素出来
L[1:3]
#只用一个:，表示从头到尾
L[:]
#因此，L[:]实际上复制出了一个新list
#切片操作还可以指定第三个参数
L[::2]
L = ['Adam', 'Lisa', 'Bart', 'Paul']
>>> L[::2]
['Adam', 'Bart']
#第三个参数表示每N个取一个，上面的L[::2]会每两个元素取出一个来，也就是隔一个取一个。
#把list换成tuple，切片操作完全相同，只是切片的结果也变成了tuple。
```

```
#range()函数可以创建一个数列
range(1,101)
[1,2,3,...,100]
```

9 倒序切片

```
L = ['Adam', 'Lisa', 'Bart', 'Paul']

>>> L[-2:]
['Bart', 'Paul']

>>> L[:-2]
['Adam', 'Lisa']

>>> L[-3:-1]
['Lisa', 'Bart']

>>> L[-4:-1:2]
['Adam', 'Bart']
#记住倒数第一个元素的索引是-1.倒叙切片包含起始索引，不包含结束索引。
```

10 对字符串切片

```
>>> 'ABCDEFG'[:3]
'ABC'
>>> 'ABCDEFG'[-3:]
'EFG'
>>> 'ABCDEFG'[::2]
'ACEG'
```

字符串有个方法upper()可以把字符变成大写字母

```
>>> 'abc'.upper()
'ABC'
```

11 什么是迭代

迭代就是对于一个集合，无论该集合是有序还是无序，我们用for循环总是可以依次取出集合的每一个元素。

集合是指包含一组元素的数据结构，我们已经介绍的包括：
1.有序集合：list,tuple,str和unicode;
2.无序集合：set
3.无序集合并且具有key-value对：dict

迭代是一个动词，它指的是一种操作，在python中，就是for循环。

12 索引迭代

Python中，迭代永远是取出元素本身，而非元素的索引。我们想要在for循环中拿出索引，要使用enumerate()函数：

```
>>> L = ['Adam', 'Lisa', 'Bart', 'Paul']
>>> for index, name in enumerate(L):
...     print index, '-', name
...
0 - Adam
1 - Lisa
2 - Bart
3 - Paul
```

使用enumerate()函数，我们可以在for循环中同时绑定索引index和元素name。但是，他实际相当于将迭代的每一个元素变成了一个tuple:

```
for t in enumerate(L):
    index = t[0]
    name = t[1]
    print index, '-', name
```

如果我们知道每个tuple元素都包含两个元素，for循环又可以进一步简写为：

```
for index, name in enumerate(L):
    print index, '-', name
```

可见，索引迭代也不是真的按索引访问，而是由 enumerate() 函数自动把每个元素变成 (index, element) 这样的tuple，再迭代，就同时获得了索引和元素本身。

zip()函数可以把两个 list 变成一个 list：

```
>>> zip([10, 20, 30], ['A', 'B', 'C'])
[(10, 'A'), (20, 'B'), (30, 'C')]
```

13 迭代dict的value

dict 对象有一个 values() 方法，这个方法把dict转换成一个包含所有value的list，这样，我们迭代的就是 dict的每一个 value:

```
d = { 'Adam': 95, 'Lisa': 85, 'Bart': 59 }
print d.values()
# [85, 95, 59]
for v in d.values():
    print v
# 85
# 95
# 59
```

dict除了values()方法外，还有一个 itervalues() 方法，用 itervalues() 方法替代 values() 方法，迭代效果完全一样:

```
d = { 'Adam': 95, 'Lisa': 85, 'Bart': 59 }
print d.itervalues()
# <dictionary-valueiterator object at 0x106adbb50>
for v in d.itervalues():
    print v
# 85
# 95
# 59
```

14 迭代dict中的key和value

dict 对象的 items() 方法返回的值:

```
>>> d = { 'Adam': 95, 'Lisa': 85, 'Bart': 59 }
>>> print d.items()
[('Lisa', 85), ('Adam', 95), ('Bart', 59)]
```

可以看到，items() 方法把dict对象转换成了包含tuple的list，我们对这个list进行迭代，可以同时获得key和value:

```
>>> for key, value in d.items():
...     print key, ':', value
...
Lisa : 85
Adam : 95
Bart : 59
```