**Daily Report**

12th February, 2018

1. **默认参数设置问题**

一是必选参数在前，默认参数在后，否则Python的解释器会报错

二是如何设置默认参数

定义默认参数要牢记一点：默认参数必须指向不变对象！

1. **可变参数定义**

定义可变参数和定义一个list或tuple参数相比，仅仅在参数前面加了一个\*号

定义可变参数后，如：

def calc(\*numbers):

sum = 0

for n in numbers:

sum = sum + n \* n

return sum

未定义可变参数时，如：

def calc(numbers):

sum = 0

for n in numbers:

sum = sum + n \* n

return sum

若对于已有list或tuple，可以在其前加\*号，使得其变为可变参数，如：

>>> nums = [1, 2, 3]

>>> calc(\*nums)

14

1. >>> extra = {'city': 'Beijing', 'job': 'Engineer'}

>>> person('Jack', 24, \*\*extra)

name: Jack age: 24 other: {'city': 'Beijing', 'job': 'Engineer'}

\*\*extra表示把extra这个dict的所有key-value用关键字参数传入到函数的\*\*kw参数，kw将获得一个dict，注意kw获得的dict是extra的一份拷贝，对kw的改动不会影响到函数外的extra

1. 如果要限制关键字参数的名字，就可以用命名关键字参数，例如，只接收city和job作为关键字参数。这种方式定义的函数如下：

def person(name, age, \*, city, job):

print(name, age, city, job)

1. 如果函数定义中已经有了一个可变参数，后面跟着的命名关键字参数就不再需要一个特殊分隔符\*：

def person(name, age, \*args, city, job):

print(name, age, args, city, job)

1. 参数定义的顺序必须是：必选参数、默认参数、可变参数、命名关键字参数和关键字参数；对于任意函数，都可以通过类似func(\*args, \*\*kw)的形式调用它，无论它的参数是如何定义的
2. 判断对象是否可迭代

通过collections模块的Iterable类型判断：

>>> from collections import Iterable

>>> isinstance('abc', Iterable) # str是否可迭代

True

>>> isinstance([1,2,3], Iterable) # list是否可迭代

True

>>> isinstance(123, Iterable) # 整数是否可迭代

False

1. Python内置的enumerate函数可以把一个list变成索引-元素对，这样就可以在for循环中同时迭代索引和元素本身：

>>> for i, value in enumerate(['A', 'B', 'C']):

print(i, value)

0 A

1 B

2 C

1. 生成器：generator

一边循环一边计算的机制

第一种方法很简单，只要把一个列表生成式的[]改成()，就创建了一个generator：

>>> L = [x \* x for x in range(10)]

>>> L

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

>>> g = (x \* x for x in range(10))

>>> g

<generator object <genexpr> at 0x1022ef630>

并且通过next（）获得generator内的元素；也可以使用循环，将g内的元素输出

第二种方法是将函数变为generator，只需将print变为yield

函数是顺序执行，遇到return语句或者最后一行函数语句就返回。而变成generator的函数，在每次调用next()的时候执行，遇到yield语句返回，再次执行时从上次返回的yield语句处继续执行