Lesson4

Monkey by 51reboot

大纲

- 回顾
- 集合set
- 内置函数
- 函数式编程
- 文件
- Q&A

回顾

51Reboot

- 为什么要序列化?
- 序列化和反序列化

```
- json、pickle(cPickle)
- json vs pickle
```

集合set

- 概念
- 定义
- 操作

概念

无序的，不能有重复的元素，也不能排序。

定义

```
s = set() // new empty set object
s = set(iterable) // new set object
s = {1,3,4,6}

注意:
- 空集， 不是s = {}，这是空字典!
```

操作

```
.add
.clear
.pop
.remove
.difference     //差集
.intersection   //交集
.union          //并集
```

内置函数

- sum
- enumerate | range
- random

sum

- 计算1到100的和，包括100

enumerate | range

```
nameList = ["xiaoming", "xiaohong", "xiaoli", "xiaowang"]

idx = 1
for name in nameList:
    print(idx, name)
    idx += 1

print("------------v1-------------")

for i in range(0, len(nameList)):
    print(i, nameList[i])

print("------------v2-------------")

for i, info in enumerate(nameList):
    print(i, info)

print("------------v3-------------")
```

random

- 概念

```
- 随机数相关的模块
```

- 操作

```
import random

r = random.random() // 生成0~1之间的浮点数
r = random.uniform(1, 100)  // 指定范围内的浮点数
r = random.randint(1, 100)  // 生成指定范围内的整数
```

函数

- 内置函数
- 自定义函数

内置函数

```
- print
- pprint
- input
- type
- len
- max
- min
- range
- time
- os    // 获取/user/local下每个文件的大小
- datetime
- logging
...
```

自定义函数

- 定义
- 语法
- 参数
- 示例
- 练习

定义


- 以 def 关键字开头，后接函数名称和圆括号 ()。
- 参数必须放在圆括号内，多个参数之间用逗号分隔。
- return [表达式] 结束函数，选择性地返回一个值给调用方。不带表达式的 return 相当于返回 None。
- return 可以返回多个值，此时返回的数据未元组类型。
- 定义参数时，带默认值的参数必须在无默认值参数的后面。

- 函数三要素：

1. 函数名
2. 函数参数
3. 返回值

语法


```
def 函数名(参数列表):
    '''函数体'''
```

示例1

```
def f1():
    print("hello world!")

def f2():
    s = "hello world"
    print(s)


f1()
f2()

msg = f2()
print(msg)
```

示例2

```
def f1():
    s = "hello world"
    return

def f2():
    s = "hello world"
    return s

def f3():
    num1 = 2
    num2 = 3
    return num1 + num2

def f4():
    num1 = 2
    num2 = 3
    return num1 + num2, num1 * num2

msg = f1()
msg = f2()
msg = f3()
msg = f4()
```

参数

51Reboot

- 位置参数
- 关键字参数
- 默认参数
- 可变参数

位置参数

51Reboot

```
def f(x, y, z):
    print(x, y, z)


f(1, 2, 3)
```

- 结论:

参数是通过其位置进行匹配的，从左往右，而且必须精确的传递和参数头部参数名一样多的参数;

关键字参数

```
def f(x, y, z):
    print(x, y, z)


f(x=1, y=2, z=3)
f(z=3, x=1, y=2)
```

- 结论:

在调用函数时， 能够更详尽的定义内容传递的位置，关键字参数允许通过变量名进行匹配，而不是通过位置

示例

51Reboot

- 位置参数 和 关键字参数 混合使用

```
def f(x, y, z):
    print(x, y, z)


f(1, y=2, z=3)
```

默认参数

```
def f(x, y=2, z=3):
    print(x, y, z)


f(1)
f(x=1)
f(10, 28)
f(1, 3, 5)
```

- 结论:

默认参数允许创建函数可选择的参数，如果没有传入值的话，在函数运行前，参数就被赋予了默认值;

示例

51Reboot

- 关键字参数 和 默认参数 混合使用

```
def f(x, y=2, z=3):
    print(x, y, z)



f(8, z=6)
```

可变参数

- 定义时，使用*和**进行任意数目参数收集；
- 调用时，使用*和**进行参数解包；

可变参数 ☞ 定义时

- 定义时，使用*和**进行任意数目参数收集；

```
def f(x, *args):
    print(x, args)

f(1, 2, 3, 4)
```

```
def f(**kwargs):
    print(kwargs)

f(x=1, y=2, z=3)
```

- 结论：

```
符号 *
用元组的形式收集不匹配的位置参数；当这个函数调用时，python将所有的位置相关的参数收集到一个新的元组中，
并将这个元素赋值给变量args

符号 **
它只对关键字参数有效，在这种情况下，**允许将关键字参数转化为字典，你能够在之后使用键调用来进行或字典迭代；
```

示例

51Reboot

- 混合方式

```
def f(x, *args, **kwargs):
    print(x, args, kwargs)

f(1, 2, 3, x=4, y=5)
```

- 分析

在这个列子中，1按位置传给x， 2和3收集到args位置的元组中，x和y放入到kwargs关键字字典中；

可变参数 ☞ 调用时

- 调用时，使用*和**进行参数解包；

```
def f(a,b,c,d):
    print(a,b,c,d)

args = (1,2,3,4)
f(*args)
```

```
def f(a,b,c,d):
    print(a,b,c,d)

kwargs = {'a': 1, 'b':2, 'c':3, 'd':4}
f(**kwargs)
```

示例

- 混合方式

```
def f(a, b, c, d, e, f):
    print(a, b, c, d, e, f)


args = (2, 3)
kwargs = {'d': 4, 'e': 5}

f(1, *args, f=6, **kwargs)
```

- 结论:

```
1. 函数调用的时候把形参关键字上面
2. 默认参数一定要放在位置参数后面
```

练习

- 登录验证

```
1. 函数名为authentication
2. 参数username和password2个参数;
3. 验证username=admin并且password=51reboot，如果验证成功返回True，否则返回False。
```

- 持久化

```
def readFile(filename):
    '''
    1. 处理逻辑
    2. 返回值是指文件的内容
    '''
    return


def writeFile(filename, data):
    '''
    1. 处理逻辑
    2. 返回值是指数据写入成功还是失败
    '''
    return
```

匿名函数

51Reboot

- 定义
- 语法
- 示例
- 练习

定义

- lambda 来创建匿名函数；
- lambda 只是一个表达式，函数体比 def 简单很多；
- lambda 的主体是一个表达式，而不是一个代码块。仅仅能在 lambda 表达式中封装有限的逻辑进去；

语法


```
lambda [arg1 [,arg2,.....argn]]:expression
```

示例

lambda (https://github.com/467754239/python/blob/master/basic/function_program.md)

51Reboot

- 排序

```
>>> dic = {'a' : 2, 'b' : 1, 'c' : 3}
>>> sorted(dic.items(), key=lambda x:x[1])

>>> help(sorted)
Signature: sorted(iterable, /, *, key=None, reverse=False)
Docstring:
Return a new list containing all items from the iterable in ascending order.

A custom key function can be supplied to customize the sort order, and the
reverse flag can be set to request the result in descending order.
```

文件

- 操作

```
.write
.read
.readline
.readlines
.fileno
.tell
.seek
```

- 学习tailf简易版(https://github.com/zhengyscn/python-doc/tree/master/tail)

作业

- 用户管理系统 V3

  - 函数
  - 导出csv
  - 日志审计

Q&A

51Reboot

Thank you

zhengyscn@gmail.com (mailto:zhengyscn@gmail.com)
http://github.com/zhengyscn (http://github.com/zhengyscn)