**ex1**

如果代码中有中文，需要在头顶#coding:utf-8

**ex2**

注释用#，输出用print

**ex3，ex4**

基本的加减乘除整除求余等跟c语言是一样的

**ex5**

格式化输出：

%r无论是什么都直接打印出来

%%输出一个百分号

%c字符及其ASCII码

%s字符串

%d有符号整数（十进制）

%u无符号整数（十进制）

%o无符号整数（八进制）

%x无符号整数（十六进制）

%X无符号整数（十六进制大写字母）

%e浮点数字（科学计数法）

%E浮点数字（科学计数法，用E代替e）

%f浮点数字（用小数点符号）

%g浮点数字（根据值的大小采用%e或者%f）

%G浮点数字（类似%g）

%p指针（用于十六进制打印值的内存地址）

%n存储输出字符的数量放进参数列表的下一个变量中

**ex6**

%除了作为求余的运算符，还可用于格式化输出

1. 百分号后面可以直接跟数值或者变量名，多个变量的话用括号括起来（变量1，变量2）
2. print 变量 ---- 这样输出是不带引号的

print ‘内容’或者 print “内容”---- 这样输出都是带单引号的

1. 用%s是只输出字符串的内容，不带引号；而%r是照原样输出，带上原来的单引号/双引号一起输出
2. 可以用+来连接2个字符串，也可以用%

用法：字符串1写完以后加上%r，字符串2正常，输出的时候用 字符串1%字符串2

**ex7**

通常用单引号来创建比较简短的字符，用双引号来创建比较长的一句话

print “.”\* 10 表示输出10个.

print语句的后面默认会换行，加上逗号，表示不换行

**ex8**

想要输出中文的话用%s，如果用%r会输出为ASCII码，\xe4\xb8\x80这种

python会用最有效的方式进行输出，而不严格按照我们的单引号还是双引号

**ex9**

%r输出所有内容，换行符\n不生效

想输出大段文字，包括换行符可以用”””

与注释的区别在于：第一次”””之前有没有换行

”””

这是注释

”””

print ”””

这是大段文字

可以换行

”””

**ex10**

使用”””和’’’进行注释，或者输出大段文字的效果都是一样的

转义字符：

\\输出反斜杠

\’输出单引号

\”输出双引号

\a响铃符（用print '\a'确实可以听到pc嘟一声）

\b退格符

\f进纸符

\n换行符

\N{name} Unicode数据库中的字符名

\r ASCII回车符

\t ASCII制表符

\uxxxx 值为16位十六进制xxxx的字符

\Uxxxxxxxx 值为32位十六进制xxxxxxxx的字符

\v垂直制表符

\ooo值为八进制ooo的字符

\xhh值为十六进制hh的字符

**ex11**

print “How old are you？”

age = raw\_input()

或者直接用

age = raw\_input(“How old are you?\n”)不用\n的话就不换行

python 2.x 用的都是raw\_input，随便输入都是字符串

python 3用的是input，需要带引号才是字符串

**ex12**

用pydoc raw\_input进入帮助文档页面以后，用q来退出

raw\_input后面不带逗号的话，输出的时候将用户输入的值用引号括起来，如‘12’

age = raw\_input ("How old are you?")

height = raw\_input("How tall are you?")

weight = raw\_input("How much do you weigh?")

longjiazhendeMacBook-Air:py longjiazhen$ python ex12.py

How old are you?12

How tall are you?23

How much do you weigh?34

So, you are '12' old, '23' tall and '34' heavy.

raw\_input后面带逗号的话，输出的时候将用户输入的值用括号引号逗号括起来，如(‘12’,)

age = raw\_input ("How old are you?"),

height = raw\_input("How tall are you?"),

weight = raw\_input("How much do you weigh?"),

longjiazhendeMacBook-Air:py longjiazhen$ python ex12.py

How old are you?12

How tall are you?23

How much do you weigh?34

So, you are ('12',) old, ('23',) tall and ('34',) heavy.

**ex13**

代码：

#运行代码的时候引入argv参数，即用python ex13.py name age height来执行

from sys import argv

#将argv中的东西解包，依次赋值给左边的变量

script,first,second,third = argv

first = raw\_input('第一个参数：')

second = raw\_input('第二个参数：')

third = raw\_input('第三个参数：')

#这和raw\_input的区别在于，raw\_input需要在运行过程中等待用户输入，而agrv是在运行整个程序之初就要输入的

#程序执行到这里的时候，script，first，second，third都是已知的，其实跟用%输出变量差不多，不过这种情况用的是，逗号

print "The script is called:", script

print "You first variable is:", first

print "You second variable is:", second

print "You third variable is:", third

运行结果：

longjiazhendeMacBook-Air:py longjiazhen$ python ex13.py name age height

第一个参数：ljz

第二个参数：21

第三个参数：155

The script is called: ex13.py

You first variable is: ljz

You second variable is: 21

You third variable is: 155

**ex14**

将argv和raw\_input结合起来

可以将等待用户输入时候的提示符指定为>

prompt = ‘>’

a = raw\_input(prompt)

**ex15**

from sys import argv 这里sys是一个代码库，从sys中取出argv功能

还有另一种更常见的写法：

import sys

a = sys.argv[0]

print (a)

这样输出的a就是脚本的名称

换句话讲，sys.argv[]列表存储的是用户执行脚本时候输入的各项参数

python读写文件的操作也与c类似，用open()打开文件，用read()读取其中的内容，读完以后用close()关闭

可以写一个这样的脚本a.py，另有一份文档名为b.txt

运行脚本的时候用python a.py b.txt

from sys import argv

script,filename = argv

text = open(filename)

print text.read()

text.close()

还有另一种方法，是直接在python命令行里面打开文件

longjiazhendeMacBook-Air:py longjiazhen$ python

>>> print open("b.txt").read()

**ex16**

close关闭文件

read读取文件，可以把读取的结果赋值给一个变量

readline读取文本文件中的一行

truncate清空文件

write(stuff)将stuff写入文件中

open(filename)默认以只读权限打开

open(filename,’w’)以写权限打开

打开文件的方式：

r读 w写 a追加

w+ r+ a+ 同时以读写方式打开

**ex17**

from sys import argv

from os.path import exists

检查文件是否存在，如果存在返回true，如果不存在返回false

indata = open(from\_file).read

print “The input file is %d bytes long.”% len(indata) 计算文本长度

print “Does the output file exist?”% exists(to\_file)

out\_file = open(to\_file,’w’).write(indata)

**ex18**

def 函数名(参数):

函数体

函数体

注意上面的冒号，还有函数体是要对齐的（4个空格的缩进），换行了也要对齐

不带参数 def none():

带一个参数 def one(argv):

带两个参数 def two(argv1,argv2):

带两个参数 def two(\*args):

agrv1,argv2 = args

**ex19**

函数所需要的参数的值可以由用户自己输入，例如定义这样一个函数

def cheese\_and\_crackers(cheese\_count, box\_of\_crackers):

print "You have %d cheese." % cheese\_count

print "You have %d crackers." % box\_of\_crackers

print "That's enough.\n"

调用函数的时候可以用

a = int(raw\_input("cheese numbers:"))

b = int(raw\_input("crackers numbers:"))

cheese\_and\_crackers(a,b)

使用int()把raw\_input()的值转换为整型，或者直接使用input()

a = input("cheese numbers:")

**ex20**

file.seek()的目的是将游标移到任意位置，然后从该位置开始操作，0表示移到开头

file.readline()会读到\n为止，然后停止读取文件，返回之前读到的内容，file会记录每次调用readline()读到的位置，然后下次被调用时读取下一行

**ex21**

无论输入什么，都照本输出raw\_input()

将自定义的输入值转换成int类型，int(raw\_input())

将自定义的输入值转换成float类型，float(raw\_input())

**ex24**

主要复习了之前的转义字符、输出、函数、加减乘除

**ex25**

回顾一下ex15：

可以写一个这样的脚本a.py，另有一份文档名为b.txt

运行脚本的时候用python a.py b.txt

from sys import argv

script,filename = argv

text = open(filename)

print text.read()

text.close()

还有另一种方法，是直接在python命令行里面打开文件

longjiazhendeMacBook-Air:py longjiazhen$ python

>>> print open("b.txt").read()

学习ex25：

1. 可以在ex25.py中定义函数，在ex25.py直接调用，用python ex25.py来运行

s.splite (' ') 表示按空格来分割，s.pop(0)表示弹出第一个，s.pop(-1)表示弹出最后一个

1. 在ex25.py中定义函数，在ex25\_3.py中用from ex25 import \* 来导入，接着就可以正常使用那些函数了（注意1：这里是from ex25不是from ex25.py 注意2：书里说直接import ex25也可，但是我试了不行，得用from ex25 import \*）
2. 在ex25.py中定义函数，然后在python中运行代码，即手动在python命令行中输入需要的参数，调用函数等等

longjiazhendeMacBook-Air:py longjiazhen$ python

>>> from ex25 import \*

>>> sentence = "chance always for those who happy"

>>> print\_first\_and\_last(sentence)

chance

happy

>>> print\_first\_and\_last\_sorted(sentence)

always

who

>>>

进入python以后，用help(ex25)可以查看帮助文档（定义函数时候的"""和"""之间的并不是普通的注释，而是帮助文档啊，以后要养成这个习惯）

Help on module ex25:

NAME

ex25 - #coding:utf-8

FILE

/Volumes/Transcend/py/ex25.py

FUNCTIONS

break\_words(stuff)

This function will break words for us.

print\_first\_and\_last(sentence)

Prints the first and last word of a sentence.

print\_first\_and\_last\_sorted(sentence)

Sorts the words then prints the first and last one.

print\_first\_word(words)

Prints the first word after poping it off.

print\_last\_word(words)

Prints the last word after poping it off.

sort\_sentence(sentence)

Takes in a full sentence and returns the sorted words.

sort\_words(words)

Sorts the words.

(END)

用help(ex25. break\_words)可以查看ex25.py里面break\_words()这个函数所需要的参数和说明

Help on function break\_words in module ex25:

break\_words(stuff)

This function will break words for us.

(END)

进入帮助文档以后用q退出，进入python以后用ctrl+D退出

总结：有点像C语言里面，在.h头文件里面定义函数，然后在.c文件里面#include头文件，然后调用函数。前面几节学到的from sys import argv大概也是这样吧

**ex28 布尔表达式**

转移字符\’输出’,\\输出\，\n换行符，\t制表符

"test" and "test" 返回'test'（python不会严格按照我们输入的双引号/单引号来输出）；1 and 1 返回1，而不是返回True

False and 1和1 and False都是返回False，原因是：False and 1的时候，判断了and前面是False，后面就不用继续判断了，所以直接返回False；1 and False的时候，判断了前面是1还不够，还要继续往后面走，读到False的时候才判断为False，所以返回False

or也是一样，如果or前面能判断出正确还是错误，那么后面就不用判断了，直接返回。

**ex29 if**

在python里面，如果一行以冒号结束，那么它接下来的内容应该缩进（前面学的函数要缩进，这节学的if语句也要缩进）

**ex30 elif**

if

elif （C语言里的else if被缩写成elif）

else

如果多个elif的区块都是true的话，python只会处理它遇到的第一个true的区块，所以只有第一个true的会被运行

**ex31 if-elif-else**

如何判断一个值是否在某个阈值内，1<x<10或者x in range(1,10)

door = raw\_input("> ") 这样表示输出>提示符给用户，并将用户输入的数值赋给door

if door == "1":

elif door == "2":

else:

**ex32 列表**

C语言里面叫数组，Python里面叫列表

C语言里叫二维数组，Python里叫二元列表

A = [1,2,3,4]

B = ['a','b','c','d']

C = [1,'b',3,'d']

以上三种方式都支持，也就是说，列表里面可以是整型、字符型、甚至同时包含整型和字符型。读取列表的时候，整型用%d，字符型用%s，不知道可以用%r

elements = [] 新建空列表

for i in range(0,6) 循环6次，含首不含尾，0-5

elements.append(i) 将i追加至elements数组末端

elements = range(0,6) 输出整个列表，从[0]至5[]

range(start, stop[, step]) 如果不填step，默认是1，若step=2，意思是0 2 4 6 8 这样循环

二元列表a = [[1,2,3],[4,5,6]]

为什么for-loop可以使用未定义的变量？循环开始时这个变量就被定义了，每循环一次重新定义一次

**ex33 while循环**

尽量少用while-loop，大多数时候for-loop是更好的选择

**ex34 读取列表元素**

基数是从0开始，与列表下标相匹配的；序数是从1开始，与第几第几相匹配的

**ex35 分支和函数**

用str.isdigit()来判断str是不是一个数值（很奇怪，pydoc isdigit看不到帮助文档）

exit(0)表示程序正常退出，1或者其他数值表示异常退出

**ex36**

最好的debug方式是在怀疑的地方用print输出，而不是去使用debugger

程序最好写一点运行一点，调通了再加功能

在python里面，除非死循环，不然尽量不要用while循环，而用for循环

**ex37**

【很多看不懂，以后再回来看】

关键字

and 与 可用于布尔判断里面，A and B 满足A且满足B

del 与python的垃圾回收机制gc模块有关

from 从……中 例如 import argv from sys

not 否 布尔运算

while 当…… 例如while循环

as 一般和with一起使用，用于异常处理

https://www.cnblogs.com/DswCnblog/p/6126588.html

elif 例如if……elif……else

global 全局的 用于定义全局变量？不是不用定义变量么？

or 或者 布尔运算

with

assert ？

else 例如if……elif……else

if 例如if……elif……else

pass ？

yield ？

break 终止 例如for循环里面，如果满足什么条件就终止，退出循环 ？

except 异常处理？

import 导入 例如 import argv from sys

print 输出

class 面向对象的类

exec ？

in 在……中 for i in range(0,10)

raise

continue 继续 例如for循环里面，如果满足什么条件就继续？改一下之前那个很多房间的游戏

finally ？

is 有A is equal B这种用法么

return 返回 比如函数的返回值

def 用于定义函数

for 用于for循环

lamda ？

try 异常处理？

数据类型

True 是

False 否

None 空？

strings 字符串？

numbers 所有数字？

floats 浮点型？

lists 列表？

字符串转义序列

\\ 输出\

\' 输出'

\" 输出"

\a 响铃符

\b

\f

\n 换行符

\r

\t 制表符

\v

字符串格式化

%d 整型

%i ？

%o 八进制

%u 无符号？

%x 十六进制

%X 十六进制

%e 科学计数法

%E 科学计数法

%f 浮点型

%F 浮点型

%g ？

%G ？

%c 字符型

%r 无论输出是什么，原样输出

%s 字符串

%% 输出%

操作符号

+ 加

- 减

\* 乘

\*\* ？

/ 除

//

%求余

< 小于

> 大于

<= 小于等于

>= 大于等于

== 等于

!= 不等于

<> ？

() ？

[] 向下取整？

{} ？

@ ？

,

:

.

= 赋值

;

+= 加等于

-= 减等于

\*= 乘等于

/= 除等于

//= ？

%= 求余等于

\*\*= ？

**ex38 ex39字典**

print ‘-’\*10 可以输出10个-

Dict = {'A': 'a', 'B': 'b', 'C': 'c'}这种叫字典，前面是key，后面是value

可用Dict ['D'] = 'd'进行赋值

可用Dict.items()方法返回遍历的(key, value)列表

for k,v in d.items():

print "%s %s" %(k,v)

可用get()方法从字典中获取指定key值对应的value，若无此key值则返回None

Dict.get(key[,default None]) 第二个参数可以不填，可以填None，也可以填任意字符串

在命令行里用pydoc dict查看字典的帮助文档

字典的内容是无序的，因为是按照hash排列的，collections.OrderedDict 是按照输入次序进行排序的（按输入顺序，而不是正序或者逆序）

d1 = collections.OrderedDict()

有序字典和无序字典相比，只要内容一样，就认为是相等

有序字典和有序字典相比，即使内容一样，只要次序不一样，也认为不想等

**ex40 类**

【模块和字典】

字典：key是apple，value是 I AM APPLES！

mystuff = {'apple' : "I AM APPLES！"}

print mystuff['apple']

module 模块

模块是包含函数和变量的python文件

可以import这个文件

然后使用‘.’操作符访问模块里的函数和变量

模块和字典的使用方式类似

mystuff['apple'] # get apple from dict

mystuff.apple() # get apple from the module

mystuff.tangerine # same thing, it's just a variable

也就是说，python里面有这么一个通用模式：

1.拿一个类似key=value风格的数据容器

2.通过key的名称获取其中的value

对字典来说，key是一个字符串，获得值的取法是[key]

对模块来说，key是函数或者变量的名称，获得值的取法是.key

【模块和类】

class mystuff(object):

def \_\_init\_\_(self):

self.tangerine = "And now a thousand years between"

def Apple(self):

print "I AM CLASSY APPLES!"

区别：

一个类可以创建很多个对象，这些对象互不干扰。但是模块的话，整个代码里就只有一份，会互相干扰。

【对象和import】

thing = mystuff()

thing.Apple()

print thing.tangerine

综上，有3种方式可以从某个东西里面获取内容：

# 字典

mystuff['apples']

# 模块

mystuff.apples()

print mystuff.tangerine

# 类

thing = MyStuff()

thing.apples()

print thing.tangerine

实例：

def \_\_init\_\_(self, lyrics):

self.lyrics = lyrics

#为什么要加self，为了明确是对象的lyrics属性，还是一个叫lyrics的局部变量

**ex43**

书里代码印得不完整，网上找了一份

https://blog.csdn.net/weixin\_41586246/article/details/79301627

原作者说报错，我实际也报错，没有解决。看不懂

strip()移除字符串头尾

1. Python strip() 方法用于移除字符串头尾指定的字符（默认为空格或换行符）或字符序列。

注意：该方法只能删除开头或是结尾的字符，不能删除中间部分的字符。

2. strip()方法语法：str.strip([chars]);

参数chars -- 移除字符串头尾指定的字符序列。

返回值 返回移除字符串头尾指定的字符生成的新字符串。

shuffle()将序列的所有元素随机排序

1. 描述 shuffle() 方法将序列的所有元素随机排序。

2. 以下是 shuffle() 方法的语法:

import random

random.shuffle (lst )

注意：shuffle()是不能直接访问的，需要导入 random 模块，然后通过 random 静态对象调用该方法。

参数 lst -- 可以是一个列表。

返回值 该函数没有返回值。

**ex44 继承Inheritance和合成Composition**

大部分使用继承的场景都可以用合成替代，而多级继承更是要不惜一切避免