一、input()与raw\_input()的区别；

1、这两个均是 python 的内建函数，通过读取控制台的输入与用户实现交互。但他们的功能不尽相同；

2、input()函数支持用户输入数字或者表达式,不支持输入字符串，返回的是数字类型的数值。raw\_input()函数捕获的是用户的原始输入,返回为字符串。

**3、当输入为纯数字时**  
\* input返回的是数值类型，如int，float  
\* raw\_inpout返回的是字符串类型，string类型  
  
**输入字符串为表达式**  
input会计算在字符串中的数字表达式，而raw\_input不会。  
如输入 “1+ 1”：  
\* input会得到整数2  
\* raw\_input会得到字符串”1 + 1”

**Python urllib2模块**

**1、最基本的使用方法实例：**

import urllib2

request = urllib2.Request("http://www.baidu.com")

response = urllib2.urlopen(request)

print response.read()

**2、请求中包含需要发送的数据时的使用实例：**

import urllib

import urllib2

values = {"username":"1016903103@qq.com","password":"XXXX"}

data = urllib.urlencode(values)

url = "https://passport.csdn.net/account/login?from=http://my.csdn.net/my/mycsdn"

request = urllib2.Request(url,data)

response = urllib2.urlopen(request)

print response.read()

注：

urllib.urlencode(query[, doseq])：将dict或者包含两个元素的元组列表转换成url参数。

例如 字典{'name': 'dark-bull', 'age': 200}将被转换为"name=dark-bull&age=200"

urlopen返回 一个类文件对象，他提供了如下方法：

read() , readline() , readlines() , fileno() , close() ：这些方法的使用方式与文件对象完全一样;

info()：返回一个httplib.HTTPMessage 对象，表示远程服务器返回的头信息；

getcode()：返回Http状态码。如果是http请求，200表示请求成功完成;404表示网址未找到；

geturl()：返回请求的url；

**3、添加headers到http请求的方法：**

import urllib

import urllib2

url = 'http://www.pythontab.com'

user\_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'

values = {'name' : 'Michael Foord',

'location' : 'pythontab',

'language' : 'Python' }

headers = { 'User-Agent' : user\_agent }

data = urllib.urlencode(values)

req = urllib2.Request(url, data, headers)

response = urllib2.urlopen(req)

the\_page = response.read()

**方法二：**

import urllib2

request = urllib2.Request(uri)

request.**add\_header**('User-Agent', 'fake-client')

response = urllib2.urlopen(request)

**python os模块**

1. os.remove() 删除文件

例：

Import os;

os.remove(“D:\\test.txt”) //删除D盘中的test.txt文件

1. os.rename() 重命名文件或文件夹

例：

Import os;

os.rename(“D:\\test.txt”, “D:\\test1.txt”)

注：若不带文件前的路径，只写文件名的话，默认为python安装路径

3、其它

os.listdir() 列出指定目录下所有文件

os.chdir() 改变当前工作目录

os.getcwd() 获取当前文件路径

os.mkdir() 新建目录

os.rmdir() 删除空目录(删除非空目录, 使用shutil.rmtree())

os.makedirs() 创建多级目录

os.removedirs() 删除多级目录

os.stat(file) 获取文件属性

os.chmod(file) 修改文件权限

os.utime(file) 修改文件时间戳

os.name(file) 获取操作系统标识

os.system() 执行操作系统命令

os.execvp() 启动一个新进程

os.fork() 获取父进程ID，在子进程返回中返回0

os.execvp() 执行外部程序脚本（Uinx）

os.spawn() 执行外部程序脚本（Windows）

os.access(path, mode) 判断文件权限(详细参考cnblogs)

**python datetime,date,time模块**

datetime.date.today() 本地日期对象,(用str函数可得到它的字面表示(2014-03-24))

datetime.date.isoformat(obj) 当前[年-月-日]字符串表示(2014-03-24)

datetime.date.fromtimestamp() 返回一个日期对象，参数是时间戳,返回 [年-月-日]

datetime.date.weekday(obj) 返回一个日期对象的星期数,周一是0

datetime.date.isoweekday(obj) 返回一个日期对象的星期数,周一是1

datetime.date.isocalendar(obj) 把日期对象返回一个带有年月日的元组

datetime对象：

datetime.datetime.today() 返回一个包含本地时间(含微秒数)的datetime对象 2014-03-24 23:31:50.419000

datetime.datetime.now([tz]) 返回指定时区的datetime对象 2014-03-24 23:31:50.419000

datetime.datetime.utcnow() 返回一个零时区的datetime对象

datetime.fromtimestamp(timestamp[,tz]) 按时间戳返回一个datetime对象，可指定时区,可用于strftime转换为日期表示

datetime.utcfromtimestamp(timestamp) 按时间戳返回一个UTC-datetime对象

datetime.datetime.strptime('2014-03-16 12:21:21',"%Y-%m-%d %H:%M:%S") 将字符串转为datetime对象

datetime.datetime.strftime(datetime.datetime.now(), '%Y%m%d %H%M%S') 将datetime对象转换为str表示形式

datetime.date.today().timetuple() 转换为时间戳datetime元组对象，可用于转换时间戳

datetime.datetime.now().timetuple()

time.mktime(timetupleobj) 将datetime元组对象转为时间戳

time.time() 当前时间戳

time.localtime

time.gmtime

**python random模块**

random.random() 产生0-1的随机浮点数

random.uniform(a, b) 产生指定范围内的随机浮点数

random.randint(a, b) 产生指定范围内的随机整数

random.randrange([start], stop[, step]) 从一个指定步长的集合中产生随机数

random.choice(sequence) 从序列中产生一个随机数

random.shuffle(x[, random]) 将一个列表中的元素打乱

random.sample(sequence, k) 从序列中随机获取指定长度的片断

code cademy学习笔记：

1、定义一个变量，并赋值

my\_int =4

定义一个字符变量：

my\_name=”liuyp”

2、定义一个布尔类型变量，并赋值：

my\_bool= True

3、定义一个函数：

Def spam():

“””this is a comments”””

print “hello”

spam()

4、注释符号 ：#

# this is a comments

5、运算：

a = 10+20

b=5-3

c=2\*4

d=100/2

指数运算：e=2\*\*3

余数运算：f=5%4

6、'This isn't flying, this is falling with style!'直接在使用不正确，需加转义符：

'This isn\'t flying, this is falling with style!'

7、将fifth\_letter变量的值设置为“MONTY”的第5个字母

fifth\_letter = "MONTY"[4]

8、计算字符串的长度函数len()

parrot = "Norwegian Blue"

print len(parrot)

将字符串全部小写化函数lower()

parrot = "Norwegian Blue"

print parrot.lower()

将字符串全部大写函数upper()

parrot = "norwegian blue"

print parrot.upper()

将非字符串转换为字符串函数str()

pi = 3.14

print str(pi)

9、输出多个字符串组合

print "Spam "+"and "+"eggs"

print "The value of pi is around " + 3.14会出错，正确写法：

print "The value of pi is around " + str(3.14)

10、打印 包含字符串和变量时：

string\_1 = "Camelot"

string\_2 = "place"

print "Let's not go to %s. 'Tis a silly %s." % (string\_1, string\_2)

打印结果：

Let's not go to Camelot. 'Tis a silly place.

11、打印实例：

name = raw\_input("What is your name?")

quest = raw\_input("What is your quest?")

color = raw\_input("What is your favorite color?")

print "Ah, so your name is %s, your quest is %s, " \

"and your favorite color is %s." % (name, quest, color)

12、

from datetime import datetime

now = datetime.now()

print now

print now.year

print now.month

print now.day

print '%s/%s/%s' % (now.year, now.month, now.day)

print '%s:%s:%s' % (now.hour, now.minute, now.second)

print '%s/%s/%s %s:%s:%s' % (now.year,now.month,now.day,now.hour, now.minute, now.second)

13、not、and和or运算符的先后顺序

1. not is evaluated first;
2. and is evaluated next;
3. or is evaluated last.

14、

导入引用模块的方法：

import math

from math import sqrt

from math import \*

15、lists的用法：

lists是一种数据类型，将一组信息数据，存放在一个变量中

zoo\_animals = ["pangolin", "cassowary", "sloth", "monkey"];

# One animal is missing!

if len(zoo\_animals) > 3:

print "The first animal at the zoo is the " + zoo\_animals[0]

print "The second animal at the zoo is the " + zoo\_animals[1]

print "The third animal at the zoo is the " + zoo\_animals[2]

print "The fourth animal at the zoo is the " + zoo\_animals[3]

为lists添加一项：

list.append(“abc”)

lists中包含项目的个数：

len(lists)

删除lists中的数据

lists.remove("dagger")

letters = ['a', 'b', 'c', 'd', 'e']

slice = letters[1:3]

print slice

slice为['b', 'c']

注意：letters[1:3]指的是从letters中的letters[1]（包含letters[1]）开始至letters[3]（但不包含letters[3]）结束

animals = "catdogfrog"

# The first three characters of animals

cat = animals[:3] 从animals[0]开始至animals[3]（不包含animals[3]）

# The fourth through sixth characters

dog = animals[3:6]

# From the seventh character to the end

frog = animals[6:] 从animals[6]开始到结尾

lists.index(“abc”)

lists.insert(index,”ccc”)

实例：

my\_list = [1,9,3,8,5,7]

for number in my\_list:

print 2\*number

实例：

start\_list = [5, 3, 1, 2, 4]

square\_list = []

# Your code here!

for number in start\_list:

square\_list.append(number\*\*2)

square\_list.sort()

print square\_list

删除lists中数据的方法：

n = [1, 3, 5]

n.pop(1) 将列表中index为1的数据删除，即去除3

n.remove(1) 将列表中值为“1”的数据删除，即去除1

del(n[1]) 将列表中index为1的数据删除，即去除3

将两个lists合并为一个list：

a = [1, 2, 3]

b = [4, 5, 6]

print a + b

# prints [1, 2, 3, 4, 5, 6]

lists中嵌套lists实例：

list\_of\_lists = [[1, 2, 3], [4, 5, 6]]

for lst in list\_of\_lists:

for item in lst:

print item

运行结果：

1

2

3

4

5

6

16、

数据字典：

d = {'key1' : 1, 'key2' : 2, 'key3' : 3}

和lists 有点像，但数据字段调用值通过key，而不是通过索引

Key可以是string或者number

residents = {'Puffin' : 104, 'Sloth' : 105, 'Burmese Python' : 106}

print residents['Puffin'] # Prints Puffin's room number

为字典添加数据：

dict\_name[new\_key] = new\_value

删除字典的数据：

del dict\_name[key\_name]

数据字典里面可以包含lists

实例：

inventory = {

'gold' : 500,

'pouch' : ['flint', 'twine', 'gemstone'], # Assigned a new list to 'pouch' key

'backpack' : ['xylophone','dagger', 'bedroll','bread loaf']

}

# Adding a key 'burlap bag' and assigning a list to it

inventory['burlap bag'] = ['apple', 'small ruby', 'three-toed sloth']

# Sorting the list found under the key 'pouch'

inventory['pouch'].sort()

# Your code here

inventory['pocket']=['seashell','strange berry','lint']

inventory['backpack'].sort()

inventory['backpack'].remove('dagger')

inventory['gold']+=50

打印字典中的所有值：

webster = {

"Aardvark" : "A star of a popular children's cartoon show.",

"Baa" : "The sound a goat makes.",

"Carpet": "Goes on the floor.",

"Dab": "A small amount."

}

# Add your code below!

for key in webster:

print webster[key]

实例：

prices={

"banana":4,

"apple":2,

"orange":1.5,

"pear":3

}

stock={

"banana":6,

"apple":0,

"orange":32,

"pear":15

}

for key in prices:

print key

print "price: %s" %prices[key]

print "stock: %s" %stock[key]

实例：

字典的key与列表的字段相同的运用实例

shopping\_list = ["banana", "orange", "apple"]

stock = {

"banana": 6,

"apple": 0,

"orange": 32,

"pear": 15

}

prices = {

"banana": 4,

"apple": 2,

"orange": 1.5,

"pear": 3

}

# Write your code below!

def compute\_bill(food):

total = 0

for item in food:

if stock[item]>0:

total+=prices[item]

stock[item]-=1

return total

实例: 列表里面可以包含字典，字典里面可以包含列表

lloyd = {

"name": "Lloyd",

"homework": [90.0, 97.0, 75.0, 92.0],

"quizzes": [88.0, 40.0, 94.0],

"tests": [75.0, 90.0]

}

alice = {

"name": "Alice",

"homework": [100.0, 92.0, 98.0, 100.0],

"quizzes": [82.0, 83.0, 91.0],

"tests": [89.0, 97.0]

}

tyler = {

"name": "Tyler",

"homework": [0.0, 87.0, 75.0, 22.0],

"quizzes": [0.0, 75.0, 78.0],

"tests": [100.0, 100.0]

}

students = [lloyd, alice, tyler]

for st in students:

print st["name"]

print st["homework"]

print st["quizzes"]

print st["tests"]

实例：

lloyd = {

"name": "Lloyd",

"homework": [90.0, 97.0, 75.0, 92.0],

"quizzes": [88.0, 40.0, 94.0],

"tests": [75.0, 90.0]

}

alice = {

"name": "Alice",

"homework": [100.0, 92.0, 98.0, 100.0],

"quizzes": [82.0, 83.0, 91.0],

"tests": [89.0, 97.0]

}

tyler = {

"name": "Tyler",

"homework": [0.0, 87.0, 75.0, 22.0],

"quizzes": [0.0, 75.0, 78.0],

"tests": [100.0, 100.0]

}

# Add your function below!

def average(numbers):

total = sum(numbers)

total = float(total)

av = total/len(numbers)

return av

def get\_average(student):

homework = average(student["homework"])

quizzes = average(student["quizzes"])

tests = average(student["tests"])

average1=0.1\*homework+0.3\*quizzes+0.6\*tests

return average1

def get\_letter\_grade(score):

if score >=90:

return "A"

elif 80<=score<90:

return "B"

elif 70<=score<80:

return "C"

elif 60<=score<70:

return "D"

else:

return "F"

print get\_letter\_grade(get\_average(lloyd))

def get\_class\_average(class\_list):

results=[]

for s in class\_list:

a = get\_average(s)

results.append(a)

return average(results)

classAverage=get\_class\_average([lloyd,alice,tyler])

print classAverage

print get\_letter\_grade(classAverage)

17、range

The range function has three different versions:

1. range(stop)
2. range(start, stop)
3. range(start, stop, step)

In all cases, the range() function returns a list of numbers from start up to (but not including) stop. Each item increases by step.

If omitted, start defaults to 0 and step defaults to 1.

实践1、

编写一个函数：将输入的整数的各数字相加。如输入1234，输出为1+2+3+4的值：10

def digit\_sum(n):

sum1= sum(map(int,str(n)))

return sum1

实践2、

编写一个函数：此函数判断输入值是否为整数（包含7.0,-7，也认为是整数）：

def is\_int(x):

a=str(x)

if type(x)==int:

return True

else:

s=a.split(".")

if s[1]==str(0):

return True

else:

return False

实践3、

字符串反转函数：def reverse(text):

str=""

l=len(text)-1

while l>=0:

str+=text[l]

print str

l-=1

return str

实践4、去除字符串里的元音字母：

def anti\_vowel(text):

newT=""

v=['a','e','i','o','u','A','E','I','O','U']

for i in range(len(text)):

if text[i] not in v:

newT+=text[i]

return newT

def median(s):

sorted(s)

if len(s)>1 and len(s)%2!=0:

m=s[(len(s)-1)/2]

elif len(s)%2==0:

m=(s[len(s)/2]+s[(len(s)/2)-1]/2

return m

0到50之间的偶数：

evens\_to\_50 = [i for i in range(51) if i % 2 == 0]

print evens\_to\_50

匿名函数：

lambda x: x % 3 == 0

应用：

my\_list = range(16)

filter(lambda x: x % 3 == 0, my\_list)

过滤掉字符串中的“X”：

garbled = "IXXX aXXmX aXXXnXoXXXXXtXhXeXXXXrX sXXXXeXcXXXrXeXt mXXeXsXXXsXaXXXXXXgXeX!XX"

message=filter(lambda x: x!="X",garbled)

print message

list comprehension应用：

squares = [x \*\* 2 for x in range(5)]

threes\_and\_fives=[i for i in range(1,16) if i%3==0 or i%5==0]

##### python内置模块

###### urllib 、urllib2

urllib. Urlopen(“http://XXXX”)

urlopen返回对象提供方法：

-read() , readline() ,readlines() , fileno() , close() ：这些方法的使用方式与文件对象完全一样

-info()：返回一个httplib.HTTPMessage对象，表示远程服务器返回的头信息

-getcode()：返回Http状态码。如果是http请求，200请求成功完成;404网址未找到

-geturl()：返回请求的url

**Python的urllib和urllib2模块都做与请求URL相关的操作，但他们提供不同的功能。他们两个最显着的差异如下：**

* **urllib2可以接受一个Request对象，并以此可以来设置一个URL的headers，但是urllib只接收一个URL。这意味着，你不能伪装你的用户代理字符串等。**
* **urllib模块可以提供进行urlencode的方法，该方法用于GET查询字符串的生成，urllib2的不具有这样的功能。这就是urllib与urllib2经常在一起使用的原因**

目前的大部分http请求都是通过urllib2来访问的

GET方法

例如百度：

百度是通过http://www.baidu.com/s?wd=XXX 来进行查询的，这样我们需要将{‘wd’:’xxx’}这个字典进行urlencode

#coding:utf-8

import urllib

import urllib2

url = 'http://www.baidu.com/s'

values = {'wd':'D\_in'}

data = urllib.urlencode(values)

print data

url2 = url+'?'+data

response = urllib2.urlopen(url2)

the\_page = response.read()

print the\_page

POST方法

import urllib

import urllib2

url = 'http://www.someserver.com/cgi-bin/register.cgi'

user\_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)' //将user\_agent写入头信息

values = {'name' : 'who','password':'123456'} //post数据

headers = { 'User-Agent' : user\_agent }

data = urllib.urlencode(values) //对post数据进行url编码

req = urllib2.Request(url, data, headers)

response = urllib2.urlopen(req)

the\_page = response.read()

进行电子档案管理系统接口测试实践时的问题：

测试接口为创建档案接口：

问题：

添加cookies、headers和data数据（格式为字典数据格式），调用requests的post接口，运行时，总是运行失败；

headers={**"User-Agent"**:User\_Agent1,**"Content-Type"**: **"multipart/form-data; boundary=-----------------------------7e121c42073c"**,**"Referer"**:**"http://192.168.1.20:8080/main/index.html"**,**"Accept-Language"**:**"zh-CN"**,**"X-Requested-With"**: **"XMLHttpRequest"**}

cookies={**"JSESSIONID"**:**"467D4B5545F5CEDB67D14735A9C56319"**,**"userId"**:**"liuyp"**, **"digest"**:**"Basic%20bGl1eXA6MTIzNDU2"**,**"userName"**:**"%u5218%u4E9A%u5E73"**,**"deptNo"**:**"whgjj"**}

后去掉data数据，并修改为get方式，可以成功运行，证明data数据有误。

通过fiddler发现，此post操作的content-type为*multipart/form-data*格式，上网搜索得知，该类型的数据上传时，不能用数据字典的方式来上传，需要用files格式来上传；后修改上传的数据为files格式：

createDAnfiles={  
 **'categoryName'**:(None,**'公积金分类中心-业务类-业务类公积金贷款'**),  
 **'categoryId'**:(None,**'YWL-GJJDK'**),  
 **'contentType'**:(None,**'TYPE1'**),  
 **'contentName'**: (None, **'55505'**),  
 **'deptNo'**: (None, **'wchgjj'**),  
 **'paperYear'**: (None, **'2017'**),  
 **'securityLevel'**: (None, **'公开'**),  
 **'retentionPeriod'**: (None, **'10年'**)  
}

再次运行脚本，仍运行失败；

后查看经理编写的哈行性能测试脚本中，发现脚本中没有添加headers，于是将自己脚本中的headers去掉，运行脚本，成功执行；问题得到解决。

后查看headers中，定义了**boundary=-----------------------------7e121c42073c**，因为boundary为随机生成的，所以此次若定义了，那么上传的数据便不会被正确分隔，导致数据错误；去掉headers中的content-type设置项，脚本也是可以成功执行的。

上面的脚本在创建档案时，没有添加附件，接下来，创建带附件的档案，在createDAnfiles中，添加附件信息：

**'Fileattachment'**:(**'new.txt'**,open(**'D:\new.txt'**,**'rb'**),**'txt'**)

运行脚本，执行失败；

后将文件路径修改为双反斜杠：

**'Fileattachment'**:(**'new.txt'**,open(**'D:\\new.txt'**,**'rb'**),**'txt'**)

运行脚本，成功执行。

创建用户接口时的问题：

利用fiddler工具查看创建用户的接口，其数据格式为**application/json**

当直接将post的数据设置为数据字典格式，如：

**userData={"employeeNo":"www","employeeName":"www","depts":"whgjj","email":"","password":"123456","roles":[],"gender":"","status":"","description":""……}**

在post方法中直接传入该数据：

**request.post(url, userData,cookies)**

执行脚本，运行失败。

将userData转化为json格式数据，再进行上传：

**requests.post(createUserURL,json.JSONEncoder().encode(createUserData),headers=headers,cookies=cookies)**

运行脚本，可以成功执行