**001、python-易错知识(一)**

**001、python的三种文件形式**

源代码： .py的文件(./–.py或者python –.py)

字节代码: 经过编译之后生成的扩展名为”pyc”的文件名;

import py\_compile

py\_compile.compile('Helloword.py')

注：Helloword.py为需要编译的文件

python Helloword2.pyc

优化代码：经过优化的源文件，扩展名为”.pyo”

python -O -m py\_compile hello.py

**002、运算符和表达式**

1.实数除法:”/”

整数除法:”//”

求余:%

x的y次方:x\*\*y

>>> 5.6/2

2.8

>>> 5.6//2

2.0

>>> 3\*\*2

9

>>> -20/8

-2.5

>>> -20//8

-3

2.逻辑运算符

逻辑与’and’

逻辑或 ‘or’

逻辑非 ‘not’

>>> 1<3 and 2>3

False

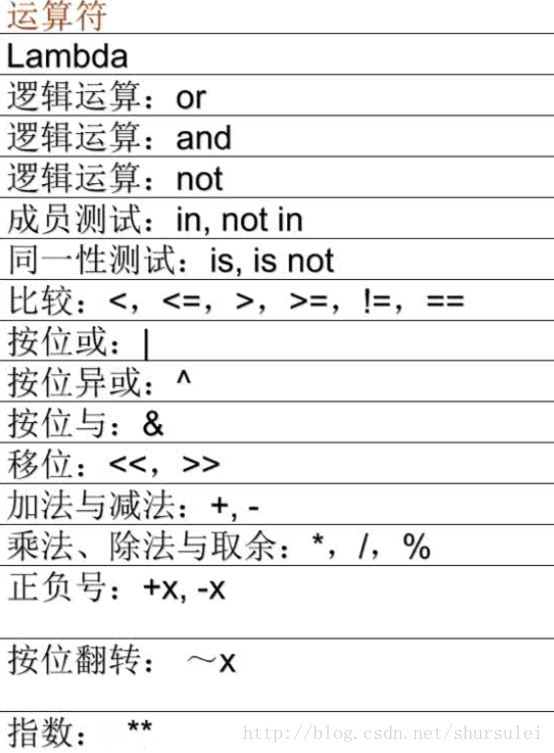
>>> not 1<3

False

>>> 1<3 or 2>4

True

优先级问题：下图由上到下，依次增大，优先级依次增高



**003、键盘的输入与输出**

python中包含两个raw\_input()和input();

但是python3中目前input()可以用

a=input()

print(a)

a=int(input("please input num1:"))

b=int(input("num2:"))

print('请输入你要转换的温度')

F=int(input())

C=5/9\*(F-32)

print(C)

**004、python的数据类型**

Number（数字）/String（字符串）/List（列表）/Tuple（元组）/Sets（集合）/Dictionary（字典）

***type()查看数据类型，python中的数据类型不需要提前声明***

1.Number类型

整数int :表示的范围为-2147483648到2147483647

在Python 3里，只有一种整数类型 int，表示为长整型，没有 python2 中的 Long。

float浮点型

>>> f1=12

>>> type(f1)

<class 'int'>

>>> f1=12.0

>>> type(f1)

<class 'float'>

复数类型:complex比如抛物线的运算

>>> type(num)

<class 'complex'>

2.String类型

1、表示形式(' '/" "/""" """)

>>> a=123

>>> star="123"*#字符串类型*

>>> str1='a'

>>> str2="b"

>>> str3="""v"""

没有区别，但是单引号的中间不能再用单引号，比如：“let's go” 此处只能用双引号，或者使用转义符:let's "go"

>>> say="let's \"go\""

>>> say

'let\'s "go"'

>>> print(say)

let's "go"

三重引号

>>> mail="""tom:

i

goof

ff

"""

控制字符串换行

>>> mail='tom:\n hello\n i am jvav'

>>> print(mail)

tom:

hello

i am jvav

>>> mail

'tom:\ni\ngoof\nff\n'

>>> print(mail)

tom:

i

goof

ff

序列类型数据:字符串，列表，数组

序列表示方法:() []

序列的两个主要特点是索引操作符(特定)和切片操作符(一部分)

索引可以是负数，从尾部开始计算

切片使用[],冒号用来分割

序列的基本操作

1.len() 求序列的长度

2.+ 连接两个序列

3.\* 序列重复出现

4.in 判断是否在序列中

5.max() 返回最大值

6.min() 返回最小值

>>> str1='abcdfrds'

>>> len(a)

9

>>> str1+'z'

'abcdfrdsz'

>>> 'z'\*2

'zz'

>>> 'x' in str1

False

>>> max(str1)

's'

>>> min(str1)

'a'

7.cmp(a,b)比较两个序列值是否相同，如果 x < y 返回 -1, 如果 x == y 返回 0, 如果 x > y 返回 1。

此处的cmp在python3中已经没有了。python3中需要导入operator模块

lt(a, b) 相当于 a < b

le(a,b) 相当于 a <= b

eq(a,b) 相当于 a == b

ne(a,b) 相当于 a != b

gt(a,b) 相当于 a > b

ge(a, b)相当于 a>= b

函数的返回值不是布尔哦。这点需要注意，还是跟cmp一样返回的是数值哦。

>>> import operator

>>> operator.eq(str1,str2)

False

字符串的处理方法：索引和切片

索引：

切片:[起始值:终点值:步长] 注意:终点值不包括结束点；索引还可以是负数，步长可以控制方向，可以重后往前取数

>>> a[0]

'a'

>>> a[0]+a[1]

'ab'

>>> a[0:5:3]

'ac'

>>> a[::2]

'avcfh'

>>> a[-2:-4:-1]*#步长可以控制方向，可以重后往前取数*

'gf

3.List类型：数据可变

列表可以放在元组中，元组中也可以存放列表

列表(list):是可变类型的数据

表现形式:[];其中可以是字符串，

列表的操作：

取值：切片和索引；list[]

添加:list.append()

删除:del(list[])

修改:list[]=x

查找:var in list

>>> a=["a",123,"456"]

>>> del(a[0])

>>> a

[123, '456']

>>> a.remove('123')*#删除第一个出现123的值*

4.元组类型（tuple）:数据不可以改变

1.元组(tuple)：元组和列表十分相似，但是元组和字符串一样是不可变的

>>> str='12345'

>>> id(str)

2397410817448

>>> str='12346'

>>> id(str)

2397410817616

在内存空间是改变的

2.元组的表示方式：()

好处：存储一系列的值，保存用户定义的函数，存储安全性较高的固定的值

3.创建元组

空元组，单一元素元组:(2,)需要加一个逗号，否则不是tuple类型，一般元组

>>> userinfo=('mike','20')

>>> print(userinfo)

('mike', '20')

>>> len(userinfo)

2

>>> userinfo[0]

'mike'

>>> t1=()

>>> t2=(2,)

4.元组的操作：和字符串操作(序列操作)一样；元组的值不可改变

>>> userinfo[1]+1

Traceback (most recent call last):

File "<pyshell#19>", line 1, in <module>

userinfo[1]+1

TypeError: Can't convert 'int' object to str implicitly

此处不可以更改

>>> a,b,c=(1,2,3)

>>> a,b,c

(1, 2, 3)

5.集合类型

6.字典类型(dict)

字典--映射

(1)映射的关系

>>> t=['name','age','gender']

>>> t2=['milo',30,'male']

>>> list(zip(t,t2))

[('name', 'milo'), ('age', 30), ('gender', 'male')]

>>> zip(t,t2)

<zip object at 0x0000026A85069D88>

python2.x中使用zip()，python3.x中使用的是list(zip())

在python 3.0中zip()是可迭代对象，使用时必须将其包含在一个list中，方便一次性显示出所有结果

(2)字典的表现形式: {}

(3)字典是python中唯一的映射类型(哈希表)

字典对象是可变的，但是字典的键必须使用不可变对象，并且一个字典中可以使用不同类型的键值

keys()或者values()返回键列表或者列表

items()返回包含键值对的元组。

>>> dic={0:0,1:1,2:2}

>>> dic[0]

0

>>> dic1={'name':'张三','age':'25','gender':'male'}

>>> dic1['name']

'张三'

(4)使用工厂的方法dict()生成字典

字典的遍历

>>> for k in dic1:

print(k)

age

gender

name

>>> for k in dic1:

dic1[k]

'25'

'male'

'张三'

(5)字典的更新和删除

更新：用键值访问更新;内建的update()方法可以将整个字典的内容拷贝到另一个字典中。

>>> dic1['tel']=10086

>>> dic1

{'age': '25', 'gender': 'male', 'tel': 10086, 'name': '张三'}*#无序*

>>> dic1

*#删除*

del(dic1['tel'])

dic1.pop('a')删除并且返回键为'a'的元素

dic1.clear()删除字典所有元素

del dic1删除整个字典

>>> dic1.pop('name')

'张三'

>>> dic1

{'age': '25', 'gender': 'male'}

(6)字典的内置函数：type(),str(),cmp

工厂函数dict():

>>> dict(x=1,y=2)

{'y': 2, 'x': 1}

>>> dic

{0: 0, 1: 1, 2: 2}

>>> dic.get(3)

获取不到序列为3的值

>>> dic.get(3,'error')

'error'

>>>



**002、python-易错知识(二)**

**001、python流程控制**

1.if-else

if condition\_1:

statement\_block\_1

elif condition\_2:

statement\_block\_2

else:

statement\_block\_3

使用or/ and/ not:与或非

2.for循环

for循环语法:

for var in sequence:

statement(s)

迭代序列指数

range():快速生成序列

range() 函数返回的是一个可迭代对象（类型是对象），而不是列表类型， 所以打印的时候不会打印列表。

list() 函数是对象迭代器，把对象转为一个列表。返回的变量类型为列表。

>>> range(100)

range(0, 100)

>>> list(range(10))#不包括结尾

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

range(i,j,[,步值])

python3中取消了xrange()函数

3.遍历(迭代)

for x in "abc":

print(x)

s="abc"

for x in range(len(s)):

print(s[x])

a

b

c

遍历字典

d={1:111,2:222,4:333}

for k,v in d.items():

print(k,v)

1 111

2 222

4 333

4.循环控制

import time

for x in range(300):

print(x)

time.sleep(1)

else:

print("ending")

ctrl+c结束进程

Traceback (most recent call last):

File "G:\Python\workspace\day02\bianli.py", line 4, in <module>

time.sleep(1)

KeyboardInterrupt

发现ending并没有执行

break/continue/pass(代码桩，占位)/exit()跳出整个过程

for x in range(1,11):

print(x)

if x==6:

break

else:

print("ending")

1

2

3

4

5

6

for x in range(1,11):

print(x)

if x==6:

break

else:

print("ending")

for x in range(1,11):

print("--------",x)

1

2

3

4

5

6

*-------- 1*

*-------- 2*

*-------- 3*

*-------- 4*

*-------- 5*

*-------- 6*

*-------- 7*

*-------- 8*

*-------- 9*

*-------- 10*

for x in range(1,11):

print(x)

if x==2:

print(5555555)

continue

if x==6:

break

else:

print("ending")

for x in range(1,11):

print('--------',x)

1

2

5555555

3

4

5

6

*-------- 1*

*-------- 2*

*-------- 3*

*-------- 4*

*-------- 5*

*-------- 6*

*-------- 7*

*-------- 8*

*-------- 9*

*-------- 10*

pass的代码

for x in range(1,11):

print(x)

if x==2:

pass

if x==6:

break

1

2

3

4

5

6

exit()跳出整个过程

for x in range(1,11):

print(x)

if x==2:

pass

if x==3:

exit()

if x==6:

break

5.while

while 1:

a=input("请输入x")

if a=='q':

break

print(a)

print("hello")

else:

print("ending---")

二.函数

1.函数的定义

函数：工具集/排序/极值

作用：降低编程的难度/代码重用

方式：def 函数名(参数列表):

函数体

>>> def add():

c=a+b

print(c)

>>> a=200

>>> b=300

>>> add()

500

def fun():

if True:

print("good")

fun()

**002、函数的参数（形参/实参）**

缺省参数(默认参数)

def machine(x,y="奶油"):

print("制作出一个",x,"元",y,"口味冰淇淋")

machine(5,"巧克力")

machine(5)*#这个值就是第一位*

machine("玫瑰")

machine(90,"玫瑰")

制作出一个 5 元 巧克力 口味冰淇淋

制作出一个 5 元 奶油 口味冰淇淋

制作出一个 玫瑰 元 奶油 口味冰淇淋

制作出一个 90 元 玫瑰 口味冰淇淋

**003、局部变量和全局变量**

**004、函数的return**

函数返回值：return 返回值

>>> def f(x,y):

print("welcome")

return x+y

>>> f(2,3)

welcome

函数在return之后结束

>>> def f():

return 5

return 6

>>> f()

5

**005、函数的其他功能(tuple-dict)**

默认参数(见上)；多类型传值

--向函数传元组和字典

------fun(\*args)

------fun(\*\*kwords)

单参数

>>> def f(x):

print(x)

>>> f(10)

10

>>> f("aaaa")

aaaa

>>> f([1,2,3,4,5])

[1, 2, 3, 4, 5]

>>> f((23,4,5,56))

(23, 4, 5, 56)

>>> f({1:111,12:234,13:233})

{1: 111, 12: 234, 13: 233}

多参数

传递元组(需要用到一个\*号)

>>> t=('name','milo')

>>> def f(x,y):

print("%s:%s" % (x,y))

>>> f(\*t)

name:milo

传递字典(需要用到两个\*号)*#无序传值，此处需要和字典的key值相同，否则出错*

>>> tt={'age':30,'name':'milo'}*#此处需设置默认值*

>>> def f(name='name',age=30):

print('name:%s'%name)

print('age:%s'%age)

>>> f(\*\*tt)

age:30

name:milo

格式化:

>>> print("%s:%s" %('name','milo'))

name:milo

-传值冗余(处理多余的实参/多个参数的接受)

--- def fun(\*args,\*\*kw)

>>> def f(x,\*args):*#此处args为元组*

print(x)

print(args)

>>> f(1)

1

()

>>> f(1,2,3)

1

(2, 3)

----- 利用字典映射

>>> def f(x,\*args,\*\*kwargs):

print(x)

print(args)

print(kwargs)

>>> f(1,2,3,4,5,6,7,8,9)

1

(2, 3, 4, 5, 6, 7, 8, 9)

{}

>>> f(x=1,y=2)*#存在映射关系*

1

()

{'y': 2}

>>> f(1,2,3,4,x=10,y=20)

Traceback (most recent call last):

File "<pyshell#35>", line 1, in <module>

f(1,2,3,4,x=10,y=20)

TypeError: f() got multiple values for argument 'x'

>>> f(1,2,3,4,z=10,y=20)

1

(2, 3, 4)

{'y': 20, 'z': 10}

**006、lambda**

lambda表达式:

--匿名函数

lambda函数是一种快速定义单行的最小函数，是从Lisp借用而来，可以用在

任何需要函数的地方。

----基本模型(冒号前面是参数)

>>> lambda x,y:x+y

<function <lambda> at 0x00000202B819DC80>*#此处返回的是一个对象*

>>> g=lambda x,y:x+y

>>> g(2,3)

5

----好处:精简代码;不需要考虑命名的问题;容易理解

----应用：

reduce:逐次操作list里的每项，接时的参数为2个，最后返回一个结果。

在Python 3里，reduce()函数已经被从全局名字空间里移除了，它现在被放置在fucntools模块里

用的话要 先引入from functools import reduce

求阶乘

*#编写一个文件(使用for循环)*

*#!/usr/bin/python3*

*# 通过用户输入数字计算阶乘*

*# 获取用户输入的数字*

num = int(input("请输入一个数字: "))

factorial = 1

*# 查看数字是负数，0 或 正数*

if num < 0:

print("抱歉，负数没有阶乘")

elif num == 0:

print("0 的阶乘为 1")

else:

for i in range(1,num + 1):

factorial = factorial\*i

print("%d 的阶乘为 %d" %(num,factorial))

>>> from functools import reduce

>>> sum=reduce(lambda x,y:x\*y,range(1,7))

>>> print(sum)

720

>>> from functools import reduce

>>> def myadd(x,y):

return x+y

>>> sum=reduce(myadd,(1,2,3))

>>> sum

6

**007、dict模拟实现switch的功能(函数调用)**

switch-dict(字典实现switch的功能)

-----switch语句：用于编写多个分支结构，但是python并没有switch语句

函数调用:

通过字典调用函数

{1:case1,2:case2}.get(x,lambda \*args,\*\*key:)()

if-elif-elif形式的计算器

from \_\_future\_\_ import division

def add(x,y):

return x+y

def sub(x,y):

return x-y

def cheng(x,y):

return x\*y

def div(x,y):

return x/y

def operator(x,o,y):

if o=='+':

print(add(x,y))

elif o=='-':

print(sub(x,y))

elif o=='\*':

print(cheng(x,y))

elif o=='/':

print(div(x,y))

else:

pass

operator(2,'+',4)

字典调用的形式:

*#字典中必须包含该值*

from \_\_future\_\_ import division

def add(x,y):

return x+y

def sub(x,y):

return x-y

def cheng(x,y):

return x\*y

def div(x,y):

return x/y

operator={"+":add,"-":sub,"\*":cheng,"/":div}

*#print(operator["+"](3,2))#调用函数,通过函数与字典的调用,5*

def f(x,o,y):

print(operator.get(o)(x,y))

f(3,"+",2)*#5*

[返回主页](https://www.cnblogs.com/cchenyang/)**003、本篇用于记录在写leetcode时遇到的python易错知识。**

**001、Python range() 函数用法:**

range(start, stop[, step])

start: 计数从 start 开始。默认是从 0 开始。例如range（5）等价于range（0， 5）;

stop: 计数到 stop 结束，**但不包括 stop**。例如：range（0， 5） 是[0, 1, 2, 3, 4]没有5

step：步长，默认为1。例如：range（0， 5） 等价于 range(0, 5, 1)

**002、python数组删除元素的三种方法:**

1)remove: 删除单个元素，删除首个符合条件的元素，按值删除

举例说明:

>>> str=[1,2,3,4,5,2,6]

>>> str.remove(2)

>>> str

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

2)pop:  删除单个或多个元素，按位删除(根据索引删除)

>>> str=[0,1,2,3,4,5,6]

>>> str.pop(1)   #pop删除时会返回被删除的元素

>>> str

[0, 2, 3, 4, 5, 6]

3)del：它是根据索引(元素所在位置)来删除

举例说明:

>>> str=[1,2,3,4,5,2,6]

>>> del str[1]

>>> str

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

del还可以删除指定范围内的值。

>>> str=[0,1,2,3,4,5,6]

>>> del str[2:4]  #删除从第2个元素开始，到第4个为止的元素(但是不包括尾部元素)

>>> str

[0, 1, 4, 5, 6]

注意：del是删除引用(变量)而不是删除对象(数据)，对象由自动垃圾回收机制（GC）删除。

**003、python自带库:collections**

collections是Python内建的一个集合模块，提供了许多有用的集合类。

Counter是collections库的函数，是一个简单的计数器，例如，统计字符出现的个数：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | from collections import Counter  c = Counter()  for ch in 'programming':      c[ch] = c[ch] + 1  print(c)  >>Counter({'r': 2, 'g': 2, 'm': 2, 'p': 1, 'o': 1, 'a': 1, 'i': 1, 'n': 1}) |

2019.9.3

**004、数组添加元素**

1)append

在数组末尾添加元素

list1 = ['Google', 'Runoob', 'Taobao']

list1.append('Baidu')

print ("更新后的列表 : ", list1)

更新后的列表 : ['Google', 'Runoob', 'Taobao', 'Baidu']

2)insert

在数组指定位置插入元素

list.insert(index, obj)

index -- 对象 obj 需要插入的索引位置。

obj -- 要插入列表中的对象。

aList = [123, 'xyz', 'zara', 'abc']

aList.insert( 3, 2009)

print "Final List : ", aList

Final List : [123, 'xyz', 'zara', 2009, 'abc']

 2019.9.9

**005、字典的get():**

字典的get()方法可用于解决字典对字典里键值的增加。

dict.get(key, default=None)

key -- 字典中要查找的键。

default -- 如果指定键的值不存在时，返回该默认值值。

dict1 = {}

dict1[0]=4

print(dict1)

>>{0: 4}

dict1[0] = dict1.get(0,0)+1

print(dict1)

>>{0: 5}

get()方法搜索0，若0不存在，则把0的值设为0；若0存在，则把0的值+1。

**2019.9.9**

**006、列表转字符串:**

res = [2,5,7,9]

result = "".join(str(i) for i in res)

print(result)

print(type(result))

>>2579

>><class 'str'>

**2019.9.14**

**007、python文件读写**

python 文件读写可用 open 和with open 两种方式。

# opem

try:

f = open('/path/to/file', 'r')

print(f.read())

finally:

if f:

f.close()

|  |  |
| --- | --- |
| 1  2  3  4 | # with open  with open("..\test.txt", 'w') as f:      f.write("hello")      f.read() |

with open 方法，无需close，每次进行操作完便会自动关闭。

# 文件读写方式

r : 只读

r+ : 读写 从头开些

w: 只写

w+: 读写 从头开写

a: 写 继续写入

a+: 读写 继续写入

# IO 常用method

tell()：告诉你文件内的当前位置

必须先导入OS模块

rename()：修改文件名

os.rename(current\_file\_name, new\_file\_name)

remove()：删除文件名

os.remove(file\_name)

mkdir()：创建目录

os.mkdir("newdir")

chdir()：改变当前目录

os.chdir("newdir")

**004、Python易错知识点**

**001、\_\_name\_\_**

class Person:

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

pass

def getAge(self):

print \_\_name\_\_

p = Person()

p.getAge()

以上代码作为脚本文件运行时（而不是作为模块被引用），将输出\_\_main\_\_

**002、sys.argv**

以这种方式运行脚本

python my.py v1 v2

from sys import argv如何获得v2的参数值?

：argv[2]

sys.argv是传递给python脚本的命令行参数【字符串】列表

argv[0]为该脚本自身路径，其余为命令行参数

**003、浅复制和深复制：**

浅复制是指只拷贝父对象，不会拷贝对象的内部的子对象，内部子对象的引用是共享的。

深复制则完全独立于原对象

import copy

a = [1, 2, 3, 4, ['a', 'b']]

b = a

c = copy.copy(a)

d = copy.deepcopy(a)

a.append(5)

a[4].append('c')

a == [1,2, 3, 4, ['a', 'b', 'c'], 5]

b == [1,2, 3, 4, ['a', 'b', 'c'], 5]

c == [1,2, 3, 4, ['a', 'b', 'c']]

d == [1,2, 3, 4, ['a', 'b']]

**004、字符串和字符概念问题：**

python中不存在char，字符串也不以“\0”结尾

以”\0”结尾是C/C++中存在。

**005、逻辑运算**

a = 'a'

print a > 'b' or 'c'

:c

**006、字典**

dict2 = { 3 : 5 }

dict4 = {(1,2,3): “uestc”}

错误：

dict3 = {[1,2,3]: “uestc”}

字典的键必须是唯一且不可变的，而list对象是可变的，所以不能作为键值来生成字典。

**007、装饰器函数**

def dec(f):

n = 3

def wrapper(\*args,\*\*kw):

return f(\*args,\*\*kw) \* n

return wrapper

@dec

def foo(n):

return n \* 2

foo(2)

:12

装饰器本身是一个函数，目的是在不改变待装饰函数代码的情况下，额外增加功能，装饰器的返回值是已装饰的函数对象。

所以上面的foo（2）等价于：

foo = dec(foo)

foo(2)

执行过程其实为：

foo(2) = wrapper(2) = foo(2)\*3 = 2\*2\*3 = 12

**008、异常捕捉顺序问题**

a = 1

try:

a += 1

expect:

a += 1

else:

a += 1

finally:

a += 1

print a

：4

try:的语句出现异常才会执行except后的语句，

如果正常，则执行完try后执行else。

另外，finally语句不管有无异常都会执行。

**009、内存管理**

对： 变量不必事先声明

错： 变量无须先创建和赋值而直接使用

对： 变量无须指定类型

对：可以使用del释放资源

Python 是弱类型脚本语言，变量就是变量，没有特定类型，因此不需要声明。

但**每个变量在使用前都必须赋值**，变量赋值以后该变量才会被创建。

用 del 语句可以释放已创建的变量（已占用的资源）。

**010、\*args 和 \*\*kwargs**

\*args：（表示的就是将实参中按照位置传值，多出来的值都给args，且以元祖的方式呈现）

def foo(x,y=1,\*args):

print(x)

print(y)

print(args)

foo(1,2,3,4,5)#其中的x为1，y=1的值被2重置了，3,4,5都给了args

\*\*kwargs：（表示的就是形参中按照关键字传值把多余的传值以字典的方式呈现）

def foo(x,\*args,\*\*kwargs):

print(x)

print(args)

print(kwargs)

foo(1,2,3,4,y=1,a=2,b=3,c=4)#将1传给了x，将2,3,4以元组方式传给了args，y=1,a=2,b=3,c=4以字典的方式给了kwargs

1

(2, 3, 4)

{'y': 1, 'a': 2, 'b': 3, 'c': 4}

位置参数、*args、kwargs三者的顺序必须是位置参数、*args、**kwargs**

**位置参数、默认参数、**kwargs三者的顺序必须是位置参数、默认参数、\*\*kwargs，

**011、python 浮点数问题：**

1.2 - 1.0

Out[160]: 0.19999999999999996

因为Python 中浮点数的运算存在误差

**012、变量有效范围：**

x = 1

def change(a):

x += 1

print x

change(x)

报错：因为x在未赋值前调用。

正确写法：用 global 声明变量 x，使其成为全局变量

x = 1

def change(a):

global x

x += 1

print x

change(x)

正确写法2：将函数内的 x 参数改为 a

x = 1

def change(a):

a += 1

print a

change(x)

**013、运行层面问题：**

从运行层面上来看，从四个选项选出不同的一个：

JAVA

Python

objectC

C#

Python 只有它是动态语言

动态语言的定义：动态编程语言 是高级程序设计语言 的一个类别，在计算机科学领域已被广泛应用。它是一类 在运行时可以改变其结构的语言：例如新的函数、对象、甚至代码可以被引进，已有的函数可以被删除或是其他结构上的变化。众所周知的 [ECMAScript](http://zh.wikipedia.org/wiki/ECMAScript) （ [JavaScript](http://zh.wikipedia.org/wiki/JavaScript) ）便是一个动态语言，除此之外如 [PHP](http://zh.wikipedia.org/wiki/PHP) 、 [Ruby](http://zh.wikipedia.org/wiki/Ruby) 、 [Python](http://zh.wikipedia.org/wiki/Python) 等也都属于动态语言，而 [C](http://zh.wikipedia.org/wiki/C%E8%AF%AD%E8%A8%80) 、 [C++](http://zh.wikipedia.org/wiki/C%2B%2B) 等语言则不属于动态语言。

**014、方法命名方式**

Python中单下划线\_foo与双下划线\_\_foo与\_\_foo\_\_的成员，下列说法正确的是?

对：

\_foo 不能直接用于’from module import \*’

\_\_foo解析器用\_classname\_\_foo来代替这个名字，以区别和其他类相同的命名

\_\_foo\_\_代表python里特殊方法专用的标识

错：

\_\_foo 可以直接用于’from module import \*’

python中主要存在四种命名方式：

1、object ：公用方法

2、\_object ：半保护

被看作是“protect”，意思是只有类对象和子类对象自己能访问到这些变量， 在模块或类外不可以使用，不能用’from module import \*’导入。

\_\_object 是为了避免与子类的方法名称冲突， 对于该标识符描述的方法，父类的方法不能轻易地被子类的方法覆盖，他们的名字实际上是\_classname\_\_methodname。

3、\_\_ object：全私有，全保护

私有成员“private”，意思是只有类对象自己能访问，连子类对象也不能访

问到这个数据，不能用’from module import \*’导入。

4、\_\_ object\_\_ ：内建方法，用户不要这样定义

**015、\_\_new\_\_和\_\_init\_\_的区别**

\_\_new\_\_是一个静态方法，而\_\_init\_\_是一个实例方法

\_\_new\_\_方法会返回一个创建的实例，而\_\_init\_\_什么都不返回

只有在\_\_new\_\_返回一个cls的实例时，后面的\_\_init\_\_才能被调用

当创建一个新实例时调用\_\_new\_\_，初始化一个实例时用\_\_init\_\_

**016、解释型语言的特性**

python是解释性语言

解释性语言和编译性语言的定义：

计算机不能直接理解高级语言，只能直接理解机器语言，所以必须要把高级语言翻译成机器语言，计算机才能执行高级语言编写的程序。

翻译的方式有两种，一个是编译，一个是解释。两种方式只是翻译的时间不同。

解释性语言的定义：

解释性语言的程序不需要编译，在运行程序的时候才翻译，每个语句都是执行的时候才翻译。这样解释性语言每执行一次就需要逐行翻译一次，效率比较低。

现代解释性语言通常把源程序编译成中间代码，然后用解释器把中间代码一条条翻译成目标机器代码，一条条执行。

编译性语言的定义：

编译性语言写的程序在被执行之前，需要一个专门的编译过程，把程序编译成为机器语言的文件，比如exe文件，以后要运行的话就不用重新翻译了，直接使用编译的结果就行了（exe文件），因为翻译只做了一次，运行时不需要翻译，所以编译型语言的程序执行效率高。

**017、Python的映射类型**

字典是 Python 语言中唯一的映射类型。

**018、xrange()**

[i\*\*i for i in xrange(3)]

[1, 1, 4]

0\*\*0=1 , 1\*\*1=1, 2\*\*2=4

**019、协程**

下列对协程的理解错误的是?

一个线程可以运行多个协程：对

协程的调度由所在程序自身控制：对

Linux中线程的调度由操作系统控制：对

Linux中协程的调度由操作系统控制：错

协程是一种用户态的轻量级线程，协程的调度完全由用户控制。

协程拥有自己的寄存器上下文和栈。

协程调度切换时，将寄存器上下文和栈保存到其他地方，在切回来的时候，恢复先前保存的寄存器上下文和栈，直接操作栈则基本没有内核切换的开销，可以不加锁的访问全局变量，所以上下文的切换非常快。

**020、编解码**

有一段python的编码程序如下：urllib.quote(line.decode("gbk").encode("utf-16")),请问经过该编码的字符串的解码顺序是：

url解码 utf16 gbk

题中，先将line解码为gbk编码格式，在编码为utf-16,在进行url编码

**021、元组**

若 a = (1, 2, 3)，下列哪些操作是合法的?

a[1:-1]

a\*3

a[2] = 4 错误

list(a)

元组（tuple）不可变