**https://taizilongxu.gitbooks.io/stackoverflow-about-python/content/1/README.html**

**Question 1**

What is Python really? You can (and are encouraged) make comparisons to other technologies in your answer

1. **- Python is an interpreted language.** That means that, unlike languages like C and its variants, Python **does not need to be compiled before it is run**. Other interpreted languages include **PHP and Ruby.**
2. Python is dynamically typed; this means that you don't need to state the types of variables.不需要声明变量
3. Python is well suited to **object orientated programming in that it allows the definition of classes along with composition and inheritance**. Python does not have access specifiers (like C++'s public, private)
4. Writing Python code is quick but running it is often slower than compiled languages.  Python allows the inclusion of C based extensions so bottlenecks can be optimised away and often are. The numpy package is a good example of this, it's really quite quick because a lot of the number crunching it does isn't actually done by Python
5. Python finds use in many spheres - web applications, automation, scientific modelling, big data applications and many more. It's also often used as "glue" code to get other languages and components to play nice.
6. [Python makes difficult things easy](https://xkcd.com/353/) so programmers can focus on overriding algorithms and structures rather than nitty-gritty low level details.
7. It supports functional and structured programming methods as well as OOP.
8. It can be used as a scripting language or can be compiled to byte-code for building large applications.
9. It provides very high-level dynamic data types and supports dynamic type checking.
10. It supports automatic garbage collection.
11. It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

## Question 2

Fill in the missing code:打印出一个目录下所有的目录，和os.walk类似

def print\_directory\_contents(sPath):

import os

for sChild in os.listdir(sPath):

sChildPath = os.path.join(sPath,sChild)

if os.path.isdir(sChildPath):

print\_directory\_contents(sChildPath)

else:

print(sChildPath)

## Question 3

A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))

A1 = range(10)

A2 = sorted([i for i in A1 if i in A0])

A3 = sorted([A0[s] for s in A0])

A4 = [i for i in A1 if i in A3]

A5 = {i:i\*i for i in A1}

A6 = [[i,i\*i] for i in A1]

A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4} # the order may vary

A1 = range(0, 10) # or [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] in python 2

A2 = []

A3 = [1, 3, 2, 5, 4]

A4 = [1, 2, 3, 4, 5]

A5 = {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

A6 = [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

## Question 4

Python and multi-threading. Is it a good idea? List some ways to get some Python code to run in a parallel way.

实际上不允许多线程。It has a [multi-threading package](https://docs.python.org/2/library/threading.html) but if you want to multi-thread to speed your code up, then it's usually not a good idea to use it. Python has a construct called the Global Interpreter Lock (GIL). **全局解释器锁** **[全局](http://cn.bing.com/dict/search?q=%E5%85%A8%E5%B1%80&FORM=BDVSP6&mkt=zh-cn)[解释](http://cn.bing.com/dict/search?q=%E8%A7%A3%E9%87%8A&FORM=BDVSP6&mkt=zh-cn)****[器](http://cn.bing.com/dict/search?q=%E5%99%A8&FORM=BDVSP6&mkt=zh-cn)****[锁](http://cn.bing.com/dict/search?q=%E9%94%81&FORM=BDVSP6&mkt=zh-cn)**[用于](http://cn.bing.com/dict/search?q=%E7%94%A8%E4%BA%8E&FORM=BDVSP6&mkt=zh-cn)[保护](http://cn.bing.com/dict/search?q=%E4%BF%9D%E6%8A%A4&FORM=BDVSP6&mkt=zh-cn)[当前](http://cn.bing.com/dict/search?q=%E5%BD%93%E5%89%8D&FORM=BDVSP6&mkt=zh-cn)[线程](http://cn.bing.com/dict/search?q=%E7%BA%BF%E7%A8%8B&FORM=BDVSP6&mkt=zh-cn)[状态](http://cn.bing.com/dict/search?q=%E7%8A%B6%E6%80%81&FORM=BDVSP6&mkt=zh-cn)[的](http://cn.bing.com/dict/search?q=%E7%9A%84&FORM=BDVSP6&mkt=zh-cn)[指针](http://cn.bing.com/dict/search?q=%E6%8C%87%E9%92%88&FORM=BDVSP6&mkt=zh-cn) The GIL makes sure that only one of your 'threads' can execute at any one time. A thread acquires the GIL, does a little work, then passes the GIL onto the next thread.

There are reasons to use Python's threading package. If you want to run some things simultaneously, and efficiency is not a concern, then it's totally fine and convenient. Or if you are running code that needs to wait for something (like some IO) then it could make a lot of sense. But the threading library wont let you use extra CPU cores.

## Question 6

What does this code output:

def f(x,l=[]):

for i in range(x):

l.append(i\*i)

print(l)

f(2)

f(3,[3,2,1])

f(3)

[0, 1]

[3, 2, 1, 0, 1, 4]

[0, 1, 0, 1, 4]

## Question 7

What is monkey patching and is it ever a good idea?

### Answer

Monkey patching is changing the behaviour of a function or object after it has already been defined. For example:

import datetime

datetime.datetime.now = lambda: datetime.datetime(2012, 12, 12)

Most of the time it's a pretty terrible idea - it is usually best if things act in a well-defined way. One reason to monkey patch would be in testing. The [mock](https://pypi.python.org/pypi/mock) package is very useful to this end.

## Question 8

What does this stuff mean: \*args, \*\*kwargs? And why would we use it?

### Answer

Use \*args when we aren't sure how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple of arguments to a function. 不确定有几个arguments要pass给function，或者要pass一个list或者元组时

 \*\*kwargs is used when we dont know how many keyword arguments will be passed to a function, or it can be used to pass the values of a dictionary as keyword arguments. 不确定多少个keyword arguments或者pass字典中的值们

* \*args：位置型参数，使用 args 将位置型参数转换为 tuple；
* \*\*kwargs：关键字型参数，使用 kwargs 将关键字型参数转换为 dict；

## Question 9

What do these mean to you: @classmethod, @staticmethod, @property?

### Actual Answer

The decorators @classmethod, @staticmethod and @property are used on functions defined within classes. Here is how they behave:

## Question 10

Consider the following code, what will it output?

class A(object):

def go(self):

print("go A go!")

def stop(self):

print("stop A stop!")

def pause(self):

raise Exception("Not Implemented")

class B(A):

def go(self):

super(B, self).go()

print("go B go!")

class C(A):

def go(self):

super(C, self).go()

print("go C go!")

def stop(self):

super(C, self).stop()

print("stop C stop!")

class D(B,C):

def go(self):

super(D, self).go()

print("go D go!")

def stop(self):

super(D, self).stop()

print("stop D stop!")

def pause(self):

print("wait D wait!")

class E(B,C): pass

a = A()

b = B()

c = C()

d = D()

e = E()

# specify output from here onwards

a.go()

b.go()

c.go()

d.go()

e.go()

a.stop()

b.stop()

c.stop()

d.stop()

e.stop()

a.pause()

b.pause()

c.pause()

d.pause()

e.pause()

### Answer

The output is specified in the comments in the segment below:

a.go()

# go A go!

b.go()

# go A go!

# go B go!

c.go()

# go A go!

# go C go!

d.go()

# go A go!

# go C go!

# go B go!

# go D go!

e.go()

# go A go!

# go C go!

# go B go!

a.stop()

# stop A stop!

b.stop()

# stop A stop!

c.stop()

# stop A stop!

# stop C stop!

d.stop()

# stop A stop!

# stop C stop!

# stop D stop!

e.stop()

# stop A stop!

a.pause()

# ... Exception: Not Implemented

b.pause()

# ... Exception: Not Implemented

c.pause()

# ... Exception: Not Implemented

d.pause()

# wait D wait!

e.pause()

# ...Exception: Not Implemented

## Question 12

Describe Python's garbage collection mechanism in brief.

### Answer

1. Python对于每个object 都有几个引用指向其，如果引用数为0，指向的object就不存在了，释放内存
2. 引用循环 也会发生。 垃圾回收周期性查找引用循环并且清理。比如说一旦有两个objects o1和o2，that o1.x == o2 and o2.x == o1. If o1 and o2 are not referenced by anything else then they shouldn't be live. But each of them has a reference count of 1.
3. 另外还有一些可以加速垃圾回收。比如最新创建的objects最容易死掉。 As objects are created, the garbage collector assigns them to generations. Each object gets one generation, and younger generations are dealt with first.

Python GC主要使用引用计数（reference counting）来跟踪和回收垃圾。在引用计数的基础上，通过“标记-清除”（mark and sweep）解决容器对象可能产生的循环引用问题，通过“分代回收”（generation collection）以空间换时间的方法提高垃圾回收效率。

## Question 13

Place the following functions below in order of their efficiency. They all take in a list of numbers between 0 and 1. The list can be quite long. An example input list would be[random.random() for i in range(100000)]. How would you prove that your answer is correct?

输入是很长的一个list，判断下面三个方法的效率

def f1(lIn):

l1 = sorted(lIn)

l2 = [i for i in l1 if i<0.5]

return [i\*i for i in l2]

f1:先将整个list排序，找到小于0.5的所有数，每个平方值返回

def f2(lIn):

l1 = [i for i in lIn if i<0.5]

l2 = sorted(l1)

return [i\*i for i in l2]

f2：先找到所有的小于0.5的数，再排序，再将每个都平方

def f3(lIn):

l1 = [i\*i for i in lIn]

l2 = sorted(l1)

return [i for i in l1 if i<(0.5\*0.5)]

f3:先将每个都平方，再排序，再找到小于0.5平方的值

Most to least efficient: f2,> f1,> f3. F2效率最高

可以用profiling package

import cProfile

lIn = [random.random() for i in range(100000)]

cProfile.run('f1(lIn)')

cProfile.run('f2(lIn)')

cProfile.run('f3(lIn)')

<http://www.zhihu.com/question/33398583?sort=created>

<http://python.jobbole.com/86100/>

<http://python.jobbole.com/84153/>

<http://python.jobbole.com/85873/>

<http://python.jobbole.com/86257/>

<https://taizilongxu.gitbooks.io/stackoverflow-about-python/content/3/README.html>

<http://stackoverflow.com/questions/231767/what-does-the-yield-keyword-do-in-python>

<https://taizilongxu.gitbooks.io/stackoverflow-about-python/content/1/README.html>

<http://python.jobbole.com/86257/>

Unicode应运而生。Unicode把所有语言都统一到一套编码里，这样就不会再有乱码问题了。

Unicode标准也在不断发展，但最常用的是用两个字节表示一个字符（如果要用到非常偏僻的字符，就需要4个字节）。现代操作系统和大多数编程语言都直接支持Unicode。

现在，捋一捋ASCII编码和Unicode编码的区别：ASCII编码是1个字节，而Unicode编码通常是2个字节。

节约的精神，又出现了把Unicode编码转化为“可变长编码”的UTF-8编码。UTF-8编码把一个Unicode字符根据不同的数字大小编码成1-6个字节，常用的英文字母被编码成1个字节，汉字通常是3个字节，只有很生僻的字符才会被编码成4-6个字节。

在计算机内存中，统一使用Unicode编码，当需要保存到硬盘或者需要传输的时候，就转换为UTF-8编码。

用记事本编辑的时候，从文件读取的UTF-8字符被转换为Unicode字符到内存里，编辑完成后，保存的时候再把Unicode转换为UTF-8保存到文件：