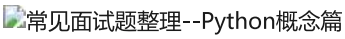

常见面试题整理--Python概念篇

路人甲
喜欢用数据讲故事，微信公众号：一个程序员的日常，合作请加微信：904727147
关注他

双十一你们的手还健在吗？ 还在的话来刷一波题？

希望此文可以长期更新并作为一篇Python的面试宝典。每一道题目都附有详细解答，以及更加详细的回答链接。此篇是概念篇，下一篇会更新面试题代码篇。

```
'''
                /题目目录/

    （一）、这两个参数是什么意思：*args，**kwargs？
    （二）、谈一谈Python的装饰器（decorator）
    （三）、简要描述Python的垃圾回收机制（garbage collection）
    （四）：Python多线程（multi-threading）。这是个好主意吗？
    （五）、 说明os,sys模块不同，并列举常用的模块方法？
    （六）、什么是lambda表达式？它有什么好处？
    （七）、Python中pass语句的作用是什么？
    （八）、Python是如何进行类型转换的？
    （九）、Python里面如何拷贝一个对象？
    （十）、__new__和__init__的区别。
    （十一）、Python中单下划线和双下划线分别是什么？
    （十二）、说一说Python自省。
    ......
'''
```


（一）、这两个参数是什么意思：*args，**kwargs？我们为什么要使用它们？

答：如果我们不确定往一个函数中传入多少参数，或者我们希望以元组（tuple）或者列表（list）的形式传参数的时候，我们可以使用*args（单星号）。如果我们不知道往函数中传递多少个关键词参数或者想传入字典的值作为关键词参数的时候我们可以使用**kwargs（双星号），args、kwargs两个标识符是约定俗成的用法。

另一种答法：当函数的参数前面有一个星号*号的时候表示这是一个可变的位置参数，两个星号**表示这个是一个可变的关键词参数。星号*把序列或者集合解包（unpack）成位置参数，两个星号**把字典解包成关键词参数。

代码辅助理解：

```
tempDict = {'s':3, 'm':4, 'c':5}

def testFunc(*args,**kwargs):
    print args,kwargs

testFunc() #() {}
testFunc(*tempList) #(1, 2, 3) {}
testFunc(*tempTuple) #(2,3,4) {}
testFunc(*tempDict) #('s','m','c'),{}
testFunc(**tempDict) #(),{'s':3,'m':4,'c':5}
testFunc(*tempList,**tempDict) #(1,2,3) {'s':3,'m':4,'c':5}
testFunc(0) #(0,) {}
testFunc(0,*tempList) #(0,1,2,3) {}
testFunc(0,**tempDict) #(0,) {'s': 3, 'm': 4, 'c': 5}
testFunc(0,*tempList,tempName = 'bye',**tempDict)
#(0, 1, 1, 2, 3) {'s': 3, 'm': 4, 'c': 5, 'tempName': 'bye'}
```

（二）、谈一谈Python的装饰器（decorator）

装饰器本质上是一个Python函数，它可以让其它函数在不作任何变动的情况下增加额外功能，装饰器的返回值也是一个函数对象。它经常用于有切面需求的场景。比如：插入日志、性能测试、事务处理、缓存、权限校验等。有了装饰器我们就可以抽离出大量的与函数功能无关的雷同代码进行重用。

有关于具体的装饰器的用法看这里：装饰器 - 廖雪峰的官方网站

```
import functools
def log(text):
    if isinstance(text, basestring):
        def decorator(func):
            functools.wraps(func)
            def wrapper(*args,**kwargs):
                func(*args,**kwargs)
                print '%s %s'%(text,func.__name__)
            print '%s %s'%('start',func.__name__)
            return wrapper
        return decorator
    else:
        functools.wraps(func)
        def wrapper(*args,**kwargs):
            print 'Call %s:'%(func.__name__)
            return func(*args,**kwargs)
        return wrapper

@log('end')
def now():
    print '2016-11-10'
now()
```

（三）、简要描述Python的垃圾回收机制（garbage collection）

Python中的垃圾回收是以**引用计数**为主，**标记-清除**和**分代收集**为辅。

- 引用计数：Python在内存中存储每个对象的引用计数，如果计数变成0，该对象就会消失，分配给该对象的内存就会释放出来。
- 标记-清除：一些容器对象，比如list、dict、tuple，instance等可能会出现引用循环，对于这些循环，垃圾回收器会定时回收这些循环（对象之间通过引用（指针）连在一起，构成一个有向图，对象构成这个有向图的节点，而引用关系构成这个有向图的边）。
- 分代收集：Python把内存根据
们所属的代。每个对象都会被

如果你想要深入了解Python的GC机制，点击这里：[转载]Python垃圾回收机制--完美讲解!

（四）、Python多线程（multi-threading）。这是个好主意吗?

Python并不支持真正意义上的多线程，Python提供了多线程包。Python中有一个叫Global Interpreter Lock（GIL）的东西，它能确保你的代码中永远只有一个线程在执行。经过GIL的处理，会增加执行的开销。这就意味着如果你先要提高代码执行效率，使用threading不是一个明智的选择，当然如果你的代码是IO密集型，多线程可以明显提高效率，相反如果你的代码是CPU密集型的这种情况下多线程大部分是鸡肋。

想要深入详细了解多线程，点击这里：详解Python中的多线程编程_python

想了解一下IO密集和CPU密集可以点击这里：CPU-bound(计算密集型) 和I/O bound(I/O密集型)

（五）、 说明**os,sys**模块不同，并列举常用的模块方法?

官方文档:

- os模板提供了一种方便的使用操作系统函数的方法
- sys模板可供访问由解释器使用或维护的变量和与解释器交互的函数

另一种回答:

os模块负责程序与操作系统的交互，提供了访问操作系统底层的接口。sys模块负责程序与Python解释器的交互，提供了一系列的函数和变量用户操作Python运行时的环境。

一些常用的方法:

一些常用的用法示例：

想要了解更详细的使用请访问：os和sys模块 - 君醉

（六）、什么是lambda表达式？它有什么好处？

简单来说，lambda表达式通常是当你需要使用一个函数，但是又不想费脑袋去命名一个函数的时候使用，也就是通常所说的匿名函数。

lambda表达式一般的形式是：关键词lambda后面紧接一个或多个参数，紧接一个冒号“：”，紧接一个表达式。lambda表达式是一个表达式不是一个语句。

想更加详细的了解Python中的Lamdba表达式可以点击这里：Lambda 表达式有何用处？如何使用？ - Python

（七）、Python中pass语句的作用是什么？

pass语句不会执行任何操作，一般作为占位符或者创建占位程序

（八）、Python是如何进行类型转换的？

Python提供了将变量或值从一种类型转换为另一种类型的内置方法。

（九）、Python里面如何拷贝一个对象？

Python中对象之间的赋值是按引用传递的，如果要拷贝对象需要使用标准模板中的copy

- copy.copy：浅拷贝，只拷贝父对象，不拷贝父对象的子对象。
- copy.deepcop y：深拷贝，拷贝父对象和子对象。

（十）、__new__和__init__的区别。

- __init__为初始化方法，__new__方法是真正的构造函数。
- __new__是实例创建之前被调用，它的任务是创建并返回该实例，是静态方法
- __init__是实例创建之后被调用的，然后设置对象属性的一些初始值。

总结：__new__方法在__init__方法之前被调用，并且__new__方法的返回值将传递给__init__方法作为第一个参数，最后__init__给这个实例设置一些参数。

想要更加详细的了解这两个方法，请点击：Python中的__new__及其用法

（十一）、Python中单下划线和双下划线分别是什么？

- __name__：一种约定，Python内部的名字，用来与用户自定义的名字区分开，防止冲突
- _name：一种约定，用来指定变量私有
- __name：解释器用_classname__name来代替这个名字用以区别和其他类相同的命名

想要更加详细的了解这两者的区

自省就是面向对象的语言所写的程序在运行时，所能知道对象的类型。简单一句话就是运行时能够获得对象的类型。比如：**type()、dir()、getattr()、hasattr()、isinstance()**

想要完整的理解Python自省，请点击：Python自省（反射）指南

有关于元类以及单例模式会在后面文章中做详细的解释说明。

本文参考文献资料：

「1」七、PYTHON 一些基础面试题目总结

「2」很全的 Python 面试题

「3」Python自省（反射）指南

「4」Python学习笔记（十二）：lambda表达式与函数式编程

「5」Python面试必须要看的15个问题

学习编程，欢迎关注专栏：学习编程 - 知乎专栏

编辑于 2017-05-23

Python 编程 程序员

文章被以下专栏收录

推荐阅读

36 条评论 ⇌ 切换为时间排序

写下你的评论...

精选评论（1）

Change State 2 年前
第一个图最后一句，0,1,1,2,3怎么会有两个1
4 查看回复

评论（36）

一个饭团 2 年前
在CPython中存在GIL，但在PyPy中没有。所以我觉得Multiple threading要看解释器吧。
4

tohilary 2 年前
这个字体是?
赞

Change State 2 年前
第一个图最后一句，0,1,1,2,3怎么会有两个1
4

路人甲 路人甲 (作者) 回复 Change State 2 年前
完了，写错了。感谢提醒。
赞

李松泽 2 年前
很实用，谢谢
赞

长脖子树 2 年前
赞一个
赞

MeeLu 2 年前
居然大部分都会了。。
赞

ReidHolmes 2 年前
Python还是陌生了一点
赞

钟文森 2 年前
简单了点。答主还没提到元类，协程，进程间通信等内容，这些很多面试也会问的
赞

陈瑞兴

柳傾 2 年前
I/O 密集型用多线程不一定好，另外在网络方面（如爬虫）用多线程还是有优势的。
赞

路人甲 路人甲 (作者) 回复 柳傾 2 年前
是的，爬虫方面多线程比较有优势。
赞

Kies 2 年前
都是好基础的啊
2

Evan 2 年前
够赞👍
赞

嘿嘿嘿 2 年前
请问有木有c++的面试题啊😂
赞

纸目 2 年前
掘金好像特小众吧
赞

sean Zhi 2 年前
那个装饰器的代码，应该是@functools.wraps，wraps本身就是一个装饰器
1

丑龙 2 年前
先收藏日后欣赏
赞

我就是我 2 年前
mark it
赞

高杰 2 年前
简直天书，我还说要转行码农呢
1

知否 2 年前
能出一个java的吗？求
1

1 2 下一页