
【干货收藏】Python面试指南大全

【干货收藏】Python面试指南大全


[已重置] [已重置]

关注他

217 人赞了该文章

收拾了一下自己学习Python过程中的笔记，将Python面试过程中经常涉及到的一些问题整理出来。没有总结到的知识点，欢迎大家在评论里提出来，本文长期更新。

1、Python基本语法

1、@staticmethod 和 @classmethod

Python中有三种方法，实例方法、类方法(@classmethod)、静态方法(@staticmethod)。

类方法的第一个参数是cls，表示该类的一个实例，静态方法基本上和一个全局函数相同

```
class A(object):
def foo(self, x):
print("executing foo(%s,%s)" % (self, x))
print('self:', self)
    @classmethod
def class_foo(cls, x):
print("executing class_foo(%s,%s)" % (cls, x))
print('cls:', cls)
    @staticmethod
def static_foo(x):
print("executing static_foo(%s)" % x)
a = A()
print(a.foo(1))
print(a.class_foo(1))
print(a.static_foo(1))
```

2、迭代器和生成器

迭代器：是访问集合元素的一种方式，从集合的第一个元素开始访问，直到所有元素被访问结束。其优点是不需要事先准备好整个迭代过程中的所有元素，仅在迭代到某个元素时才开始计算该元素。适合遍历比较巨大的集合。__iter__()：方法返回迭代器本身， __next__()：方法用于返回容器中下一个元素或数据。

生成器：带有yield的函数不再是一个普通函数，而是一个生成器。当函数被调用时，返回一个生成器对象。不像一般函数在生成值后退出，生成器函数在生成值后会自动挂起并暂停他们的执行状态。


```
'''迭代器'''
print('for x in iter([1, 2, 3, 4, 5]):')
for x in iter([1, 2, 3, 4, 5]):
print(x)

'''生成器'''
def myyield(n):
while n>0:
print("开始生成...:")
yield n
```

```
for i in myyield(4):
print("遍历得到的值:",i)
```

3、闭包

闭包可以实现先将一个参数传递给一个函数，而并不立即执行，以达到延迟求值的目的。满足以下三个条件：必须有一个内嵌函数；内嵌函数必须引用外部函数中变量；外部函数返回值必须是内嵌函数。

```
def delay_fun(x, y):
def caculator():
return x+y
return caculator

print('返回一个求和的函数，并不求和')
msum = delay_fun(3,4)
print('调用并求和:')
print(msum())
```

4、*args 和 **kwargs

这两个是Python中的可变参数，用于接受参数的传递。*args表示任何多个无名参数，它是一个元组，**kwargs表示关键字参数，它是一个字典。同时使用*args和**kwargs时，必须*args在**kwargs之前。

5、鸭子类型：

在鸭子类型中，关注的不是对象的类型本身，而是他是如何使用的。例如，在不使用鸭子类型的语言中，我们可以编写一个函数，它接受一个类型为鸭的对象，并调用它的走和叫方法。在使用鸭子类型的语言中，这样的一个函数可以接受一个任意类型的对象，并调用它的走和叫方法。

```
class duck():
def walk(self):
print('I am duck,I can walk...')
def swim(self):
print('I am duck,I can swim...')
def call(self):
print('I am duck,I can call...')

duck1=duck()
duck1.walk()
# I am duck,I can walk...
duck1.call()      # I am duck,I can call...
```

6、@property 和 @setter

@property负责把一个方法变成属性调用。在对实例操作时，不暴露接口，而是通过getter和setter方法实现。


```
class Student(object):
    @property
def score(self):
return self._score

    @score.setter
def score(self, value):
if not isinstance(value, int):
raise ValueError('score m
if value<0 or value>100:
```

```
s = Student()
s.score = 60
print(s.score)
s.score = 999
print(s.score)
```

7、多进程和多线程

进程：是资源分配的最小单位，创建和销毁开销较大；

线程：是CPU调度的最小单位，开销小，切换速度快；

操作系统将CPU时间片分配给多个线程，每个线程在指定放到时间片内完成。操作系统不断从一个线程切换到另一个线程执行，宏观上看就好像是多个线程一起执行。

Python中由于全局锁 (GIL) 的存在导致，同一时间只有一个获得GIL的线程在跑，其他线程则处于等待状态，这导致了多线程只是在做分时切换，并不能利用多核。

多线程与多进程的区别：（1）多进程中同一个变量各自有一份拷贝在每个进程中，互不影响；（2）多线程中，所有变量都由所有线程共享，任何一个变量都可被任何一个线程修改。线程之间共享数据的最大危险在于多个线程同时更改一个变量，把内容改乱。

```
from  multiprocessing  import  Pool            #多进程
from  multiprocessing.dummpy  import  Pool     #多线程
```

8、类变量和实例变量

普通的变量（非类的变量），在被赋值后即变量存在。类的变量在class里def外，通过变量名能被赋值，在def里通过类对象可被赋值

```
class Apple(object):
name = 'apple'

p1 = Apple()
p2 = Apple()
p1.name = 'orange'
print(p1.name)
print(p2.name)
```

9、装饰器

装饰器是一个工厂函数，接受一个函数作为参数，然后返回一个新函数，其闭包中包含被装饰的函数。有了装饰器，可以提取大量函数中与本身功能无关的类似代码（这块在Flask中用于定义路由的@app.route，就是一个很好的例子），达到代码重用的目的。可应用于插入日志、性能测试、事务处理等方面。


```
def deco(func):
def warpper(*args, **kwargs):
print('start')
func(*args, **kwargs)
print('end')
return warpper

@deco
def myfunc(parameter):
print("run with %s" % parameter)

myfunc("something")
```

1、MySQL基本语法

增：创建数据表

```
USE database
CREATE  TABLE  example(id  INT,
name  VARCHAR(20),
sex  BOOLEAN);
```

删：

```
ALTER  TABLE  表名 DROP  属性名;    # 删除字段
DROP  TABLE   表名;       # 删除表
```

改：

```
ALTER  TABLE  旧表名 RENAME  新表名;    # 修改表名
ALTER  TABLE  表名 MODIFY  属性名  数据类型;    # 修改字段数据类型
```

查：

```
SELECT  * FROM 表名 WHERE  id=1;   # 条件查询
SELECT  * FROM 表名 WHERE  字段名 BETWEEN 条件一 AND  条件二 # 范围查询
SELECT  COUNT(*) FROM  表名;   # 查询表共有多少条记录
```

触发器：是由INSERT、UPDATE和DELETE等事件来触发某种特定操作，满足触发条件时，数据库系统会执行触发器中定义的语句，这样可以保证某些操作之间的一致性。

```
CREATE  TRIGGER  触发器名称  BEFORE|AFTER   触发事件
ON  表名称 FOR  EACH ROW
BEGIN
执行语句
END
```

3、算法

1、快排

算法：先从数列中取出一个数作为基准；然后将比该数大的数放到右边，比该数小的数放到左边；再对左右区间重复上一步骤。

```
def qsort(seq):
if seq==[]:
return []
else:
pivot=seq[0]
lesser=qsort([x for x in seq[1:] if x<pivot])
greater=qsort([x for x in seq[1:] if x>=pivot])
return lesser+[pivot]+greater

if __name__=='__main__':
seq=[5,6,78,9,0,-1,2,3,-65,12]
print(qsort(seq))
```

2、冒泡

```
def bubbleSort(nums):
for i in range(len(nums)-1):    # 这个循环负责设置冒泡排序进行的次数
for j in range(len(nums)-i-1):  # j 为列表下标
if nums[j] > nums[j+1]:
nums[j], nums[j+1] = nums[j+1], nums[j]
return nums

nums = [5,2,45,6,8,3,1]
print(bubbleSort(nums))
```

4、网络

1、post 和 get方法区别

GET：浏览器告知服务器，只获取页面上的信息，请求的参数加到url后面；

POST：浏览器告知服务器，想在URL上发布新的信息，并且服务器必须确保数据已经存储且仅存储一次。这是html表单发送数据到服务器的方法。提交的数据放到data或body中，不能放到url中。

2、Cookie 和 Session

Cookie：存储在客户端，用于跟踪会话，保存用户偏好设置和用户名密码等，不安全；

Session：存储在服务器端，用于跟踪会话，安全。

作者：赵宏田

出处：Python爬虫实战

知乎专栏：Python爬虫实战

最近很多人私信问我问题，平常知乎评论看到不多，如果没有及时回复，大家也可以加小编微信：tszhihu，进知乎大数据分析挖掘交流群，可以跟各位老师互相交流。谢谢。

编辑于 2017-02-13

Python 面试 数据分析

文章被以下专栏收录

知乎 Python开发者社区 首发于 **Python开发者社区** 关注

[已重置] 发表于Pytho... [已重置] 发表于Pytho... [已重置] 发表于Pytho...

15 条评论 ⇌ 切换为时间排序

写下你的评论...

霍华德 2 年前
感觉自己学了假python
1

wonder041 2 年前
发python代码起码得把缩进弄上吧
13

fish scar 2 年前
我也觉得自己学了假的python，闭包，工厂模式，这不是js才有的吗？啥，python也有？！
赞

月半小厨 回复 fish scar 2 年前
这个确实有
赞

c0de 回复 fish scar 2 年前
动态语言的特性，swift也是有的
赞

北竞王 2 年前
都是比较基础的内容，再看一遍又理解深刻了一些
赞

c0de 2 年前
说的是挺好的，但是没缩进呀~
1

「已注销」 2 年前
感觉我学了假python 0.0 不过最近一次面试问了浅拷贝、鸭子类型、迭代器还有匿名函数
2

rafisher 2 年前
懂一半
赞

大叔 2 年前
呵呵㊗
赞

花满楼 1 年前
😂一直被虐，突然看到这篇文章，感觉春招还是很有希望的。
赞

KKKKKBBBBB

▲ 赞同 217 ▼ 15 条评论 分享 收藏 ···

▲ 赞同 217
分享

Zean 1 年前

看了楼主的帖子，我感觉我还要走好长的路啊。

赞

Thomc 1 年前

学的真的是假python，基础太差了

赞

Z先生 1 年前

其实这些都会也找不到工作，这是基础的基础吧

2