# **python面试题超纲20道**

### What is python .. (“dot dot”) notation syntax?

Python的 .. (点 点) 是什么语法？

f = 1..\_\_truediv\_\_ # or 1..\_\_div\_\_ for python 2print(f(8)) # prints 0.125

  真的超纲了喂。..是什么鬼啊摔。真的不是问\*\*args是什么语法么？那我们用python试试好了。

>>> f = 1.>>> f1.0>>> f.\_\_floordiv\_\_

<method-wrapper '\_\_floordiv\_\_' of float object at 0x7f9fb4dc1a20>

  咦，你是不是有看出来什么了。再看一个例子。

>>> 1..\_\_add\_\_(2.)3.0

  大家都看出来了吧，..**本身并不是什么操作符更不是什么语法**。其中，**第一个点是浮点值的一部分**，**第二个点是点操作符来访问对象属性和方法**。  
  太简单，下一题。

### Why is x\*\*4.0 faster than x\*\*4 in Python 3?

为什么在Python3中 x**4.0 比 x**4运行的快？

$ python -m timeit "for x in range(100):" " x\*\*4.0"

10000 loops, best of 3: 24.2 usec per loop

$ python -m timeit "for x in range(100):" " x\*\*4"

10000 loops, best of 3: 30.6 usec per loop

  我先解释下为什么会有这个问题，先不要讨论提问者的电脑怎么这么慢好么。我拿**python2**试了一下，结果是这样。

$ python -m timeit "for x in range(100):" " x\*\*4.0"

10000 loops, best of 3: 15.6 usec per loop

$ python -m timeit "for x in range(100):" " x\*\*4"

10000 loops, best of 3: 4.59 usec per loop

  可以明显看出**python2中int型计算明显快于float**。嗯...其实我已经回答完了不知道大家有没有发现。这个问题是由python2与python3差别引起的。  
  先指出答案核心：**python3的4是**PyLongObject**，pytohn2的4是**int**。**

* pytohn3与python2中的float对象依然是那个原生类。
* python3中的int对象实例化一个支持任意长度的成熟类，叫PyLongObject。
* python2中的**整数分为int与long**，比如：

# Python 2type(4) # <type 'int'>type(4L) # <type 'long'>

  以上的区别导致python3中整数的运算更加繁琐复杂，因为你需要用PyLongObject对象的值来执行它的ob\_digit阵数组。(参考：**Understanding memory allocation for large integers in Python for more on PyLongObjects.**)

### Given a string of a million numbers (Pi for example), write a function/program that returns all repeating 3 digit numbers and number of repetition greater than 1

给定一个长度为100万的数字（比如π），写代码计算出所有连续的三个数字，且要求该连续的三个数字至少重复出现过一次。

# For example: if the string was: 123412345123456 then the function/program would return:

# 123 - 3 times# 234 - 3 times# 345 - 2 times

  提问的人最后还加了一句能不能实现**时间复杂度为常数级**解决方案？这很明显的**面试题**嘛，假如面试时候你遇到这个你会怎么回答？  
  首先，**在O(1)情况下无法处理任意大小的数据结构**，在这种情况下，**最好的希望是O(n)**，其中n是字符串的长度，也就是**线性时间复杂度**。  
  但是这里如果每次输入定长100w，从**技术的角度来说实现O(1)是可行的（以为数据源定长嘛就相当于n=100w，这属于咬文嚼字了没意思）**，但是我相信这绝不会是这题重点。还是先给出python实现代码好了：

inpStr = '123412345123456'

# O(1) array creation.

freq = [0] \* 1000

# O(n) string processing.for val in [int(inpStr[pos:pos+3]) for pos in range(len(inpStr) - 2)]:

freq[val] += 1

# O(1) output of relevant array values.print ([(num, freq[num]) for num in range(1000) if freq[num] > 1])

  代码不复杂，相信大家都看的懂，我们继续讨论速度的问题。上述代码可见实现O(n)是没什么问题的，但是如果再提速呢？  
  方法也有，**运用归并思想，将输入值切片，多线程运行。**像这样

# 123412345123456# 拆成

vv

123412 vv

123451

5123456

  运用归并的思想去解决问题，但问题来了，因为**pytohn有GIL的存在，所以用python实现该思想可能成本很大**。当然你可以用其他语言实现。我觉得面试时候给出python代码提出归并思想就比较好了。

### Why in python 0, 0 == (0, 0) equals (0, False)

为什么在Python中表达式 0, 0 == (0, False) 的结果是 (0, False)

(0, 0) == 0, 0 # results in a two element tuple: (False, 0)

0, 0 == (0, 0) # results in a two element tuple: (0, False)

(0, 0) == (0, 0) # results in a boolean True # But:

a = 0, 0

b = (0, 0)

a == b # results in a boolean True

  这题很简单啊，我换种写法,标明优先级大家就懂了。

(((0, 0) == 0), 0) # results in a two element tuple: (False, 0)

(0, (0 == (0, 0))) # results in a two element tuple: (0, False)

((0, 0) == (0, 0)) # results in a boolean True # ALSO:

a = 0, 0

b = (0, 0)

a == b # results in a boolean True

**逗号分隔符与相等运算符的优先级是不同的。**

### Does Python optimize away a variable that's only used as a return value?

Python会把一个只用做返回值的变量优化掉吗？

# Case 1:def func():

a = 42

return a

# Case 2:def func2():

return 42

  这个问题非常有趣，我是从来没往这方面想过。我对其底层可能理解不多大家轻点喷，我把解决问题的过程帮大家写出来，大家可以看着思考一下：

# 用dis打印运行过程

from dis import dis# Case 1:

dis(func)

2 0 LOAD\_CONST 1 (42)

2 STORE\_FAST 0 (a)

3 4 LOAD\_FAST 0 (a)

6 RETURN\_VALUE

# Case 2:

dis(func2)

2 0 LOAD\_CONST 1 (42)

2 RETURN\_VALUE

作者：Tony带不带水  
链接：https://www.jianshu.com/p/8d66c5b99642  
來源：简书  
简书著作权归作者所有，任何形式的转载都请联系作者获得授权并注明出处。