python面试题搜集（四）： 最常见的 35 个 Python 面试题及答案（2018 版）

\1. Python 面试问题及答案

作为一个 Python 新手，你必须熟悉基础知识。在本文中我们将讨论一些 Python 面试的基础问题和高级问题以及答案，以帮助你完成面试。包括 Python 开发问题、编程问题、数据结构问题、和 Python 脚本问题。让我们来深入研究这些问题

Python 面试问题

Q.1. Python 的特点和优点是什么?

Python 可以作为编程的入门语言，因为他具备以下特质：

\1. 解释性

\2. 动态特性

\3. 面向对象

\4. 语法简洁

\5. 开源

\6. 丰富的社区资源

实际上 Python 的优点远不止这些，更详细的介绍可以阅读 Introduction to Python（ https://data-flair.training/blogs/python-tutorial/）

Q.2. 深拷贝和浅拷贝的区别是什么?

深拷贝是将对象本身复制给另一个对象。这意味着如果对对象的副本进行更改时不会影响原对象。在 Python 中，我们使用 deepcopy（）函数进行深拷贝，使用方法如下：

深拷贝-Python 面试问题及答案

浅拷贝是将对象的引用复制给另一个对象。因此，如果我们在副本中进行更改，则会影响原对象。使用 copy（）函数进行浅拷贝，使用方法如下：

浅拷贝—Python 面试问题及答案

Q.3. 列表和元祖有什么不同?

主要区别在于列表是可变的，元祖是不可变的。看下面的例子：

```
>>> mylist=[1,3,3]
>>> mylist[1]=2
>>> mytuple=(1,3,3)
>>> mytuple[1]=2
Traceback (most recent call last):
File "<pyshell#97>", line 1, in <module>
mytuple[1]=2
```

会出现以下错误提示：

TypeError: ‘tuple’ object does not support item assignment

关于列表和元祖的更多内容参考 Tuples vs Lists（https://data-flair.training/blogs/python-tuples-vs-lists/）

\2. Python 面试基础题

Q.4 到 Q.20 是新手经常会被问到的一些 Python 基础题，有经验的人也可以参考这些问题来复习这些概念。

Q.4. 解释 Python 中的三元表达式

与 C++不同, 在 Python 中我们不需要使用 ? 符号，而是使用如下语法：

[on true] if [expression]else [on false]

如果 [expression] 为真, 则 [on true] 部分被执行。如果表示为假则 [on false] 部分被执行

下面是例子:

2

Hi

Q.5. Python 中如何实现多线程?

线程是轻量级的进程，多线程允许一次执行多个线程。众所周知，Python 是一种多线程语言，它有一个多线程包。

GIL（全局解释器锁）确保一次执行单个线程。一个线程保存 GIL 并在将其传递给下一个线程之前执行一些操作，这就产生了并行执行的错觉。但实际上，只是线程轮流在 CPU 上。当然，所有传递都会增加执行的开销。

Q.6. 解释继承

一个类继承自另一个类，也可以说是一个孩子类/派生类/子类，继承自父类/基类/超类，同时获取所有的类成员（属性和方法）。

继承使我们可以重用代码，并且还可以更方便地创建和维护代码。Python 支持以下类型的继承：

\1. 单继承- 一个子类类继承自单个基类

\2. 多重继承- 一个子类继承自多个基类

\3. 多级继承- 一个子类继承自一个基类，而基类继承自另一个基类

\4. 分层继承- 多个子类继承自同一个基类

\5. 混合继承- 两种或两种以上继承类型的组合

关于继承的更多内容参考 Python Inheritance（https://data-flair.training/blogs/python-inheritance/）

Q.7. 什么是 Flask?

Flask 是一个使用 Python 编写的轻量级 Web 应用框架，使用 BSD 授权。其 WSGI 工具箱采用 Werkzeug，模板引擎则使用 Jinja2。除了 Werkzeug 和 Jinja2 以外几乎不依赖任何外部库。因为 Flask 被称为轻量级框架。

Flask 的会话会话使用签名 cookie 来允许用户查看和修改会话内容。它会记录从一个请求到另一个请求的信息。但如果要修改会话，则必须有密钥 Flask.secret_key。

我们将在后续的课程中进一步讨论 Flask。

Q.8. 如何在 Python 中管理内存?

Python 用一个私有堆内存空间来放置所有对象和数据结构，我们无法访问它。由解释器来管理它。不过使用一些核心 API，我们可以访问一些 Python 内存管理工具控制内存分配。

Q.9. 解释 Python 中的 help() 函数和 dir() 函数。

help() 函数返回帮助文档和参数说明：

运行结果如下:

Help on function copy in module copy

copy(x)

Shallow copy operation on arbitrary Python objects.

See the module』s **doc** string for more info.

dir() 函数返回对象中的所有成员 (任何类型)

```
1.  >>> dir(copy.copy)
```

```
['__annotations__', '__call__', '__class__', '__closure__', '__code__', '__defaults__',
'__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__get__', '__getattribute__', '__globals__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__kwdefaults__', '__le__', '__lt__', '__module__', '__name__',
'__ne__', '__new__', '__qualname__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__']
```

Q.10. 当退出 Python 时是否释放所有内存分配?

答案是否定的。那些具有对象循环引用或者全局命名空间引用的变量，在 Python 退出是往往不会被释放

另外不会释放 C 库保留的部分内容。

Q.11. 什么是猴子补丁?

在运行时动态修改类和模块

```
>>> class A:
    def func(self):
        print("Hi")
>>> def monkey(self):
print "Hi, monkey"
>>> m.A.func = monkey
>>> a = m.A()
>>> a.func()
```

Hi, monkey

Q.12. 什么是 Python 字典?

字典是我在 C++和 Java 中没有见过的数据结构，它拥有键-值对

3

字典是可变的，我们也可以用推导式的方式创建它.

{25: 5, 16: 4, 9: 3, 4: 2, 1: 1}

要了解更多字典的内容请点击 Python Dictionaries（ https://data-flair.training/blogs/python-dictionaries/)

Q.13. 能否解释一下 *args 和 **kwargs?

如果我们不知道将多少个参数传递给函数，比如当我们想传递一个列表或一个元组值时，就可以使用*args。

3

2

1

4

7

当我们不知道将会传入多少关键字参数时，使用**kwargs 会收集关键字参数。

a.1

b.2

c.7

使用 args 和 kwargs 作为参数名只是举例，可以任意替换。

对于 Python 的基础题任何疑问，请在评论区提问。

Q.14. 编程实现计算文件中的大写字母数

```
>>> import os
>>> os.chdir('C:\\Users\\lifei\\Desktop')
>>> with open('Today.txt') as today:
    count=0
    for i in today.read():
        if i.isupper():
            count+=1
    print(count)
```

26

Q.15. 什么是负索引?

我们先创建如下列表:

与正索引不同，负索引是从右边开始检索。

6

同样可以用于列表的切片:

[3, 4, 5, 6, 7]

Q.16. 如何随机打乱列表中元素，要求不引用额外的内存空间?

我们用 random 包中的 shuffle() 函数来实现。

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

Q.17. 解释 Python 中的 join() 和 split() 函数

join() 函数可以将指定的字符添加到字符串中。

'1,2,3,4,5'

split() 函数可以用指定的字符分割字符串

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

Q.18. Python 区分大小写吗?

验证 Python 是否区分大小写的方法是测试 myname 和 Myname 在程序中是不是算同一个标识符。观察以下代码的返回结果:

Myname

NameError: name 'Myname' is not defined

如你所见，这里出现了 NameError，所以 Python 是区分大小的语言。

Q.19. Python 中标识符的命名规则?

Python 中的标识符可以是任意长度，但必须遵循以下命名规则:

\1. 只能以下划线或者 A-Z/a-z 中的字母开头。

\2. 其余部分只能使用 A-Z/a-z/0-9。

\3. Python 标识符区分大小写。

\4. 关键字不能作为标识符。Python 有以下这些关键字：

and	def	False	import	not	True
as	del	finally	in	or	try
assert	elif	for	is	pass	while
break	else	from	lambda	print	with
class	except	global	None	raise	yield
continue	exec	if	nonlocal	return	

Q.20. 如何删除字符串中的前置空格

前置空格是第一个非空格字符前的所有空格，使用 lstrip() 函数来删除.

‘Ayushi ‘

如图这个字符串既包含前置空格也包含后置空格. 调用 lstrip() 函数去除了前置空格。如果想去除后置空格，使用 rstrip() 函数。

‘ Ayushi’

以上是面向 Python 新手的基础题部分。

\3. Python 面试进阶题

Q. 21 至 Q. 35 是针对有经验者的一些 Python 面试问题及其答案和示例。

Q.21. 如何将字符串转换为小写?

使用 lower() 函数

‘ayushi’

转换为大写用 upper() 函数

‘AYUSHI’

要检查字符串是否为全大写或全小写，使用 isupper() 和 islower() 函数

```
>>> 'AyuShi'.isupper()
```

False

```
>>> 'AYUSHI'.isupper()
```

True

```
>>> 'ayushi'.islower()
```

True

```
>>> '@yu$hi'.islower()
```

True

```
>>> '@YU$HI'.isupper()
```

True

像 @ 和$这样的字符即满足大写也满足小写。

istitle() 可以检查字符串是否是标题格式。

True

Q.22. Python 中的 pass 语句有什么作用?

我们在写代码时，有时可能只写了函数声明而没想好函数怎么写，但为了保证语法检查的正确必须输入一些东西。在这种情况下，我们使用 pass 语句。

类似的 break 语句可以跳出循环。

0

1

2

continue 语句可以跳到下一轮循环。

0

1

2

4

5

6

Q.23. 请解释 Python 中的闭包?

如果在一个内部函数里。对在外部作用域（但不是在全局作用域）的变量进行引用，那么内部函数就是一个闭包。

7

闭包的详细解释请点击 Closures in Python。（https://data-flair.training/blogs/python-closure/）

Q.24. 解释 Python 中的//，%和**运算符

//运算符执行地板除法，返回结果的整数部分 (向下取整)。

3

用/符号除法结果为 3.5。

符号表示取幂. ab 返回 a 的 b 次方

1024

% 是取模符号。返回除法后的余数。

6

0.5

对于 Python 进阶面试问题和答案有任何疑问请在评论区提问。

Q.24. Python 中有多少种运算符，解释算术运算符。

这类面试问题可以判断你的 Python 功底，可以举一些实例来回答这类问题。

在 Python 中我们有 7 中运算符:算术运算符、关系 (比较) 运算符、赋值运算符、逻辑运算符、位运算符、成员运算符、身份运算符。

\1. 加号 (+) 将两个对象的值相加。

15

\2. 减号 (-) 将第一个对象的值减去第二个对象的值。

-1

\3. 乘号 (*) 将两个对象的值相乘。

56

\4. 除号 (/) 将第一个对象的值除以第二个对象的值。

0.875

1.0

关于地板除法、取模和取幂，请参考上一个问题。

Q.25. 解释 Python 中的关系运算符。

关系运算符用来比较两个对象。

\1. 判断小于 (<)：如果符号左边的值比右边小则返回 True。

False

\2. 判断大于 (>)：如果符号左边的值比右边大则返回 True。

True

出现上面的错误结果是因为 Python 的浮点运算存在一些 Bug。

\3. 判断小于等于 (<=)：如果符号左边的值小于或等于右边则返回 True。

True

\4. 大判断于等于 (>=)：如果符号左边的值大于或等于右边则返回 True。

True

\5. 判断等于 (==) 如果符号两边的值相等则返回 True。

True

\6. 判断不等于 (!=) 如果符号两边的值不等则返回 True。

True

True

Q.26. 解释 Python 中的赋值和算数运算符?

这是面试中的常见题目。我们将算数运算符和赋值符号放在一起介绍。

```
>>> a=7
>>> a+=1
>>> a
```

8

```
>>> a-=1
>>> a
```

7

```
>>> a*=2
>>> a
```

14

```
>>> a/=2
>>> a
```

7.0

```
>>> a**=2
>>> a
```

49.0

```
>>> a//=3
>>> a
```

16.0

```
>>> a%=4
>>> a
```

0.0

Q.27. 解释 Python 中的逻辑运算符

Python 中有三个逻辑运算符：and、or、not

```
>>> False and True
```

False

```
>>> 7<7 or True
```

True

```
>>> not 2==2
```

False

Q.28. 解释 Python 中的成员运算符

使用 in 和 not in 运算符我们可以判断某个值是否在成员中。

```
>>> 'me' in 'disappointment'
```

True

```
>>> 'us' not in 'disappointment'
```

True

Q.29. 解释 Python 中的身份运算符

这是非常常见的 Python 面试题，用下面的示例来回答.

is 和 not is 运算符可以判断两个对象是否相同

```
>>> 10 is '10'
```

False

```
>>> True is not False
```

True

Q.30. 解释 Python 中的位运算符

此运算符按二进制位对值进行操作。

\1. 与 (&) 返回按位与结果

2

\2. 或 (|) 返回按位或结果

3

\3. 异或 (^) 返回按位异或结果

1

\4. 取反 (~) 返回按位取反结果

-3

\5. 左移位 (<<) 将符号左边数的二进制左移右边数位

4

1 的二级制 001 左移 2 位变成 100 也即十进制的 4

\6. 右移位 (>>)

1

想了解关于位运算符的更多内容请点击 Operators in Python（https://data-flair.training/blogs/python-operators/）

Q.31. 如何在 Python 使用多进制数字?

除十进制以外，在 Python 中还可以使用二进制、八进制、十六进制。

\1. 二进制数有 0 和 1 组成，我们使用 0b 或 0B 前缀表示二进制数

10

使用 bin() 函数可以将数字转换为二进制

'0b1111'

\2. 八进制数由数字 0-7 组成，使用前缀 0o 或 0O 表示 8 进制数

'0o10'

\3. 十六进数由数字 0-15 组成，使用前缀 0x 或者 0X 表示 16 进制数

'0x10'

'0xf'

Q.32. 如何获取字典中的所有键?

使用 keys() 来获取字典中的所有键

Q.33. 问什么标识符不建议使用下划线开头?

因为在 Python 中以下划线开头的变量为私有变量，如果你不想让变量私有，就不要使用下划线开头。

Q.34. 如何声明多个变量并赋值?

有两种方式:

Q.35. 什么是元组的解封装?

首先我们来介绍元组封装:

(3, 4, 5)

将 3，4，5 封装到元组 mytuple 中。

现在我们要将这些值解封装到变量 x，y，z 中

12

以上是 Python 高级面试问题和答案，新手也可以参考这些问题以获得进阶的 Python 知识。

\4. 结束语

本篇文章介绍了一些重要的 Python 面试问题和答案，后续我们还会增加。在你面试之前应该熟练掌握这些。如有想添加的问题欢迎随时评论。