110 道 Python 面试笔试题超强汇总

110 道 Python 面试笔试题汇总

一行代码实现 1-100 之和

```
sum(range(0, 101))  # 5050
```

如何在一个函数内部修改全局变量

```
num = 5
def func():
    global num
    num = 4
func()
print(num)  # 4
```

列出 5 个常用 Python 标准库?

```
os：提供了不少与操作系统相关联的函数
sys：通常用于命令行参数
re：正则匹配
math：数学运算
datetime：处理日期时间
```

如何合并两个字典?

```
    name = {'name': 'Gage'}
    age = {'age': 25}
    name.update(age)
    print(name)  # {'name': 'Gage', 'age': 25}
```

谈下 Python 的 GIL?

> GIL 是 Python 的全局解释器锁，同一进程中假如有多个线程运行，一个线程在运行 Python 程序的时候会占用 Python 解释器（加了一把锁即 GIL），使该进程内的其他线程无法运行，等该线程运行完后其他线程才能运行。如果线程运行过程中

> 遇到耗时操作，则解释器锁解开，使其他线程运行。所以在多线程中，线程的运行仍是有先后顺序的，并不是同时进行。

多进程中因为每个进程都能被系统分配资源，相当于每个进程有了一个 Python 解释器，所以多进程可以实现多个进程的同时运行，缺点是进程系统资源开销大。

Python 实现列表去重的方法？

```
num_list = [1, 3, 1, 5, 3, 6, 1]
print([num for num in set(num_list)])  # [1, 3, 5, 6]
```

fun(args,kwargs)中的args, kwargs 什么意思？

> 如果你有其他语言基础的话，你应该听说过重载的概念，对，Python 为了避免这种繁琐的情况发生，引入了 args 和 kwargs；args 用来接受非键值对的数据，即元组类型，而 kwargs 则用来接受键值对数据，即字典类型。

Python2 和 Python3 的 range（100）的区别？

> Python2 返回列表，Python3 返回迭代器，节约内存。

生成一个 16 位的随机字符串？

```
import string
print(''.join((random.choice(string.printable)) for i in
range(16)))   # X{|op?_gSM-ra%N\
```

一句话解释什么样的语言能够用装饰器？

> 函数可以作为参数传递

Python 内建数据类型有哪些？

> 整型--int
> 布尔型--bool
> 字符串--str
> 列表--list
> 元组--tuple
> 字典--dict

简述面向对象中**new**和**init**区别？

1、__new__至少要有一个参数 cls，代表当前类，此参数在实例化时由Python解释器自动识别。

2、__new__必须要有返回值，返回实例化出来的实例，这点在自己实现__new__时要特别注意，可以 **return** 父类（通过 **super**(当前类名, cls)）__new__出来的实例，或者直接是 **object** 的**__new__**出来的实例。

3、__init__有一个参数 self，就是这个__new__返回的实例，__init__在__new__的基础上可以完成一些其它初始化的动作，__init__不需要返回值。

4、如果__new__创建的是当前类的实例，会自动调用__init__函数，通过 **return** 语句里面调用的__new__函数的第一个参数是 cls 来保证是当前类实例，如果是其他类的类名，；那么实际创建返回的就是其他类的实例，其实就不会调用当前类的__init__函数，也不会调用其他类的__init__函数。

简述 with 方法打开处理文件帮我我们做了什么?

打开文件在进行读写的时候可能会出现一些异常状况，如果按照常规的f.open 写法，我们需要 try,except,finally，做异常判断，并且文件最终不管遇到什么情况，都要执行 finally f.close() 关闭文件，with 方法帮我们实现了 finally 中 f.close。

列表[1,2,3,4,5]，请使用 map() 函数输出[1,4,9,16,25]，并使用列表推导式提取出大于 10 的数，最终输出 [16,25]?

```
num_list = [1, 2, 3, 4, 5]
print([x for x in list(map(lambda x: x * x, num_list)) if x > 10])
# [16,25]
```

python 中生成随机整数、随机小数、0-1 之间小数方法?

```
import random
print(random.randint(1, 10))  # 随机整数
print(random.random())  # 0-1随机小数
print(random.uniform(2, 6)) # 指定范围[2-6]随机小数
```

避免转义给字符串加哪个字母表示原始字符串?

```
b'input\n' # bytes字节符，打印以b开头。
输出：
b'input\n'
---------------------------------------
r'input\n' # 非转义原生字符，经处理'\n'变成了'\\'和'n'。也就是\n表示的是两个字符，而不是换行。
输出：
'input\\n'
---------------------------------------
u'input\n' # unicode编码字符，python3默认字符串编码方式。
输出：
'input\n'
```

Python ，用正则匹配出标签里面的内容（“Python”），其中 class 的类名是不确定的。

```
import re
s = '<div class="nam">Python</div>'
print(re.findall(r'<div class=".*">(.*?)</div>', s)) # ['Python']
```

Python 中断言方法举例?

```
age = 10
assert 0 < age < 10
--------------------
Traceback (most recent call last):
  File "F:/MxOnline/110/exam.py", line 69, in <module>
    assert 0 < age < 10
AssertionError
```

dict 中 fromkeys 的用法

```
keys = ('info',)
print(dict.fromkeys(keys, ['Gage', '25', 'man'])) # {'info': ['Gage', '25', 'man']}
```

请用正则表达式输出汉字

```
import re
a = "not 404 found 中国 2018 我爱你"
r1 = '[a-zA-Z0-9’!"#$%&\'()*+,-./:;<=>?@，。?★、…【】《》？“”‘’！[\\]^_`{|}~]+\s?'
print(re.sub(r1, '', a))  # 中国 我爱你
```

Python2 和 Python3 区别？列举 5 个

```
1.去除了<>，全部改用!=
2.xrange() 改名为range()
3.内存操作cStringIO改为StringIO
4.加入nonlocal 作用：可以引用外层非全局变量
5.zip()、map()和filter()都返回迭代器，而不是生成器，更加节约内存
```

列出 Python 中可变数据类型和不可变数据类型，为什么?

1、可变数据类型：list、dict、set
2、不可变数据类型：int/float、str、tuple
3、原理：可变数据类型即公用一个内存空间地址，不可变数据类型即每产生一个对象就会产生一个内存地址

dict 的内部实现？

> 在 Python 中，字典是通过哈希表实现的。也就是说，字典是一个数组，而数组的索引是键经过哈希函数处理后得到的。哈希函数的目的是使键均匀地分布在数组中。由于不同的键可能具有相同的哈希值，即可能出现冲突，高级的哈希函数能够使冲突数目最小化。

s = “ajldjlajfdljfddd”，去重并从小到大排序输出”adfjl”？

```
s1 = "ajldjlajfdljfddd"
print(''.join(sorted(set(s1))))  # adfjl
```

用 lambda 函数实现两个数相乘？

```
mul = lambda x, y: x*y
print(mul(2, 4))  # 8
```

字典根据键从小到大排序？

```
    info = {'name': 'Gage', 'age': 25, 'sex': 'man'}
    print(sorted(info.items(), key=lambda x: x[0])) # [('age',
25), ('name', 'Gage'), ('sex', 'man')]
```

Python 获取当前日期？

```
    import time
    import datetime
    print(datetime.datetime.now())
    print(time.strftime('%Y-%m-%d %H:%M:%S')) # 2019-03-13
11:33:56
```

获取请求头的参数？

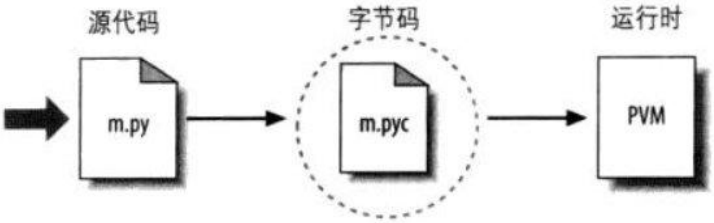
```
    from urllib.parse import urlparse, parse_qs
    s2 = "/get_feed_list?
version_name=5.0.9.0&device_id=12242channel_name=google"
    def spiltline(value):
        url = {'site': urlparse(value).path}
```

```
        url.update(parse_qs(urlparse(value).query))
        return url
    ----------------------------------------
    {'site': '/get_feed_list', 'version_name': ['5.0.9.0'],
'device_id': ['12242channel_name=google']}
```

例举五条 PEP8 规范

不要在行尾加分号，也不要用分号将两条命令放在同一行
不要使用反斜杠连接行
不要在返回语句或条件语句中使用括号
顶级定义之间空2行，方法定义之间空1行，顶级定义之间空两行
如果一个类不继承自其它类，就显式的从object继承

Python 语言的运行机制



Fibonacci 数列

```
def fab(n):
    a, b = 0, 1
    while n:
        yield b
        a, b = b, a+b
        n -= 1
```

Python 三目运算

```
# 若果 a>b 成立  就输出  a-b  否则 a+b
h = a-b if a>b else a+b
```

单例模式

```
class Single(object):
    __isstance = None
    __first_init = False
    def __new__(cls, *args, **kwargs):
        if not cls.__isstance:
            cls.__isstance = object.__new__(cls)
        return cls.__isstance
    def __init__(self, name):
```

```
        if not self.__first_init:
            self.name = name
            Singleton.__first_init = True
```

正则匹配优先级

运算符	描述
\	转义符
(), (?:), (?=), []	圆括号和方括号
*, +, ?, {n}, {n,}, {n,m}	限定符
^, $, \任何元字符、任何字符	定位点和序列（即：位置和顺序）
|	替换，"或"操作 字符具有高于替换运算符的优先级，使得"m|food"匹配"m"或"food"。若要匹配"mood"或"food"，请使用括号创建子表达式，从而产生"(m|f)ood"。

递归

```
def digui(n):
    if n == 1:
        return 1
    else:
        return (n * digui(n-1)
```

统计字符串每个单词出现的次数

```
from collections import Counter
s3 = "kjalfj;ldsjafl;hdsllfdhg;lahfbl;hl;ahlf;h"
print(Counter(s3))
```

正则 re.complie 作用

> re.compile 是将正则表达式编译成一个对象，加快速度，并重复使用

filter 方法求出列表所有奇数并构造新列表，a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
    a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    print(list(filter(lambda x: x % 2, a)))
    ---------------------
    Counter({'l': 9, ';': 6, 'h': 6, 'f': 5, 'a': 4, 'j': 3, 'd':
3, 's': 2, 'k': 1, 'g': 1, 'b': 1})
```

列表推导式求列表所有奇数并构造新列表，a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
print([x for x in a if x % 2])
```

a=（1，）b=(1)，c=(“1”) 分别是什么类型的数据?

```
print(type((1, ))) # tuple
print(type((1))) # int
print(type(("1"))) # str
```

两个列表[1,5,7,9]和[2,2,6,8]合并为[1,2,2,3,6,7,8,9]

```
l1 = [1, 5, 7, 9]
l2 = [2, 2, 6, 8]
l1.extend(l2)
```

用 python 删除文件和用 linux 命令删除文件方法

```
python：os.remove(文件名)
linux：rm 文件名
```

logging 模块的使用?

```
import logging
logging.basicConfig(level = logging.INFO,format = '%
(asctime)s - %(name)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

logger.info("Start print log")
logger.debug("Do something")
logger.warning("Something maybe fail.")
logger.info("Finish")
```

写一段自定义异常代码

```
#自定义异常用raise抛出异常
def fn():
    try:
        for i in range(5):
            if i>2:
                raise Exception("数字大于2了")
    except Exception as ret:
        print(ret)
fn()    # 数字大于2了
```

正则表达式匹配中，(.) 和 (.?) 匹配区别?

```
#（.*）是贪婪匹配，会把满足正则的尽可能多的往后匹配
#（.*?）是非贪婪匹配，会把满足正则的尽可能少匹配
s = "<a>哈哈</a><a>呵呵</a>"
import re
res1 = re.findall("<a>(.*)</a>", s)
print("贪婪匹配", res1)
res2 = re.findall("<a>(.*?)</a>", s)
print("非贪婪匹配", res2)
-------------------------
输出：
贪婪匹配 ['哈哈</a><a>呵呵']
非贪婪匹配 ['哈哈', '呵呵']
```

[[1,2],[3,4],[5,6]]一行代码展开该列表，得出[1,2,3,4,5,6]

```
a=[[1,2],[3,4],[5,6]]
print([j for i in a for j in i])
```

x=”abc”,y=”def”,z=[“d”,”e”,”f”]，分别求出 x.join(y) 和 x.join(z) 返回的结果

```
#join()括号里面的是可迭代对象，x插入可迭代对象中间，形成字符串，结果一致
x="abc"
y="def"
z=["d","e","f"]
a=x.join(y)
b=x.join(z)
print(a)
print(b)
均输出：
dabceabcf
```

举例说明异常模块中 try except else finally 的相关意义

```
try..except..else没有捕获到异常，执行else语句
try..except..finally不管是否捕获到异常，都执行finally语句
```

python 中交换两个数值

```
a,b=1,2
a,b=b,a
```

举例说明 zip() 函数用法

```
list1 = [1, 2, 3, 5]
list2 = [4, 5, 6]
zipped = zip(list1, list2)
print(list(zipped))  # [(1, 4), (2, 5), (3, 6)]
# print(list(zip(*zipped)))  # [(1, 2, 3), (4, 5, 6)]
```

a=”张明 98分”，用 re.sub，将 98 替换为 100

```
import re
a="张明 98分"
ret=re.sub(r"\d+","100",a)
print(ret)
```

a=”hello”和b=”你好”编码成 bytes 类型

```
a=b"hello"
b="你好".encode()
print(a,b)
print(type(a),type(b))
```

[1,2,3]+[4,5,6]的结果是多少?

```
print([1,2,3]+[4,5,6]) # [1, 2, 3, 4, 5, 6]
```

提高 python 运行效率的方法

1、使用生成器，因为可以节约大量内存
2、循环代码优化，避免过多重复代码的执行
3、核心模块用Cython　PyPy等，提高效率
4、多进程、多线程、协程
5、多个if elif条件判断，可以把最有可能先发生的条件放到前面写，这样可以减少程序判断的次数，提高效率

遇到 bug 如何处理

1、细节上的错误，通过print（）打印，能执行到print（）说明一般上面的代码没有问题，分段检测程序是否有问题，如果是js的话可以alert或console.log
2、如果涉及一些第三方框架，会去查官方文档或者一些技术博客。
3、对于bug的管理与归类总结，一般测试将测试出的bug用teambin等bug管理工具进行记录，然后我们会一条一条进行修改，修改的过程也是理解业务逻辑和提高自己编程逻辑缜密性的方法，我也都会收藏做一些笔记记录。
4、导包问题、城市定位多音字造成的显示错误问题

list=[2,3,5,4,9,6]，从小到大排序，不许用 sort，输出[2,3,4,5,6,9]

```
    def quicksort(list):
        if len(list)<2:
            return list
        else:
            midpivot = list[0]
            lessbeforemidpivot = [i for i in list[1:] if
i<=midpivot]
            biggerafterpivot = [i for i in list[1:] if i >
midpivot]
            finallylist = quicksort(lessbeforemidpivot)+
[midpivot]+quicksort(biggerafterpivot)
            return finallylist

    print quicksort([2,3,5,4,9,6])
```

两数相除保留两位小数

```
print(round(5/3, 2)) # 1.67
```

正则匹配，匹配日期 2018-03-20

```
    import re
    print(re.findall('((?:(?:[2468][048]00|[13579][26]00|[1-
9]\d0[48]|[1-9]\d[2468][048]|[1-9]\d[13579][26])/(?:0?2/(?:0[1-
9]|0?[1-9](?=\D)|[12]\d)))|(?:(?:[12]\d{3})/(?:(?:0?2/(?:0[1-
9]|0?[1-9](?=\D)|1\d|2[0-8]))|(?:0?[3578]/(?:0[1-9]|0?[1-9](?
=\D)|[12]\d|3[01]))|(?:0?[469]/(?:0[1-9]|0?[1-9](?=\D)|
[12]\d|30))|(?:1[02]/(?:0[1-9]|0?[1-9](?=\D)|[12]\d|3[01]))|
(?:11/(?:0[1-9]|0?[1-9](?=\D)|[12]\d|30))|(?:0?1/(?:0[1-9]|0?[1-
9](?=\D)|[12]\d|3[01])))))', 'Date: 2018/03/20'))
```

使用 pop 和 del 删除字典中的”name”字段，dic={“name”:”zs”,”age”:18}

```
dic = {"name": "zs", "age": 18}
dic.pop('name')
del dic['age']
print(dic) # {}
```

简述多线程、多进程

进程是资源（CPU、内存等）分配的基本单位，它是程序执行时的一个实例。
线程是程序执行时的最小单位，它是进程的一个执行流。
进程有自己的独立地址空间，每启动一个进程，系统就会为它分配地址空间，建立数据表来维护代码段、堆栈段和数据段，这种操作非常昂贵
线程是共享进程中的数据的，使用相同的地址空间，因此CPU切换一个线程的花费远比进程要小很多，同时创建一个线程的开销也比进程要小很多

简述 any() 和 all() 方法

```
# all如果存在0 Null False返回False,否则返回True;any与之相反
print(all([0, 1, 2, 3]))
print(all([1, 2, 3]))
```

IOError、AttributeError、ImportError、IndentationError、IndexError、KeyError、SyntaxError、NameError 分别代表什么异常

IOError：输入输出异常
AttributeError：试图访问一个对象没有的属性
ImportError：无法引入模块或包，基本是路径问题
IndentationError：语法错误，代码没有正确的对齐
IndexError：下标索引超出序列边界
KeyError:试图访问你字典里不存在的键
SyntaxError:Python代码逻辑语法出错，不能执行
NameError:使用一个还未赋予对象的变量

Python 中 copy 和 deepcopy 区别

```
# copy
l1 = [1, 2, [3, 4]]
l2 = copy.copy(l1)
l1.append(5)
l1[2].append(5)  # 子对象 改变
print(l1)
print(l2)
--------------
[1, 2, [3, 4, 5], 5]
[1, 2, [3, 4, 5]]
# deepcopy
l1 = [1, 2, [3, 4]]
l2 = copy.deepcopy(l1)
l1.append(5)
l1[2].append(5)
print(l1)
print(l2)
--------------
[1, 2, [3, 4, 5], 5]
[1, 2, [3, 4]]
```

列出几种魔法方法并简要介绍用途

__init__:对象初始化方法
__new__:创建对象时候执行的方法，单列模式会用到
__str__:当使用print输出对象的时候，只要自己定义了__str__(self)方法，那么就

会打印从在这个方法中return的数据
__del__:删除对象执行的方法

上下文管理器 with...as 的实现

```
class Close():
    def __init__(self, obj):
        self.obj = obj

    def __enter__(self):
        return self.obj  # 返回作为as目标

    def __exit__(self, exc_type, exc_val, exc_tb):
        try:
            self.obj.close()
        except AttributeError:
            print(exc_type)
```

python arg.py 1 2 命令行启动程序并传参，print(sys.argv) 会输出什么数据?

```
['arg.py', '1', '2']
```

请将[i for i in range(3)]改成生成器

```
class iter():
    def __init__(self, data):
        self.data = data
        self.loop = -1
    def __iter__(self):
        return self
    def __next__(self):
        if self.loop >= self.data:
            raise StopIteration
        self.loop += 1
        return self.loop
```

字符串转化大小写?

```
str="HHaa"
print(str.upper())
print(str.lower())
```

请说明 sort 和 sorted 对列表排序的区别

1.sort()与sorted()的不同在于，sort是在原位重新排列列表，而sorted()是产生一个新的列表。sorted(L)返回一个排序后的L，不改变原始的L；L.sort()是对原始的L进

行操作，调用后原始的L会改变，没有返回值；所以a = a.sort()是错的啦！a = sorted(a)才对。
2.sorted()适用于任何可迭代容器，list.sort()仅支持list（本身就是list的一个方法）
3.基于以上两点，sorted使用频率比list.sort()更高些，所以Python中更高级的排序技巧便通过sorted()来演示

对 foo = [-5,8,0,4,9,-4,-20,-2,8,2,-4] 进行排序，使用 lambda 函数从小到大排序

```
foo = [-5,8,0,4,9,-4,-20,-2,8,2,-4]
res=sorted(foo,key=lambda x:x)
print(res)
```

在 70 题的基础上将正数从小到大，负数从大到小

```
foo = [-5,8,0,4,9,-4,-20,-2,8,2,-4]
res=sorted(foo,key=lambda x:(x<0,abs(x)))
print(res)
```

Python 传参数是传值还是传址?

> Python 中函数参数是引用传递（注意不是值传递）。对于不可变类型（数值型、字符串、元组），因变量不能修改，所以运算不会影响到变量自身；而对于可变类型（列表字典）来说，函数体运算可能会更改传入的参数变量。

w、w+、r、r+、rb、rb+ 文件打开模式区别

r	以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。
w	打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
a	打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
rb	以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。
wb	以二进制格式打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
ab	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
r+	打开一个文件用于读写。文件指针将会放在文件的开头。
w+	打开一个文件用于读写。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
a+	打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写。
rb+	以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头。
wb+	以二进制格式打开一个文件用于读写。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
ab+	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写。

int(“1.4”)、int(1.4)的输出结果?

```
print(int("1.4"))    # 异常
print(int(1.4)) # 1
```

Python 垃圾回收机制?

1.引用计数

```
import sys
# 请在Python解释器下运行   为 2   创建一次 调用一次
str1 = 'hello world'
print(sys.getrefcount(str1))
```

2.分代技术

Python默认定义了三代对象集合，索引数越大，对象存活时间越长

Python中使用了某些启发式算法（heuristics）来加速垃圾回收。例如，越晚创建的对象更有可能被回收。对象被创建之后，垃圾回收器会分配它们所属的代（generation）。每个对象都会被分配一个代，而被分配更年轻代的对象是优先被处理的。

3.引用循环

垃圾回收器会定时寻找这个循环，并将其回收。举个例子，假设有两个对象o1和o2，而且符合o1.x == o2和o2.x == o1这两个条件。如果o1和o2没有其他代码引

Python 字典和 json 字符串相互转化方法

```
import json
dic = {"name":"zs"}
res = json.dumps(dic)
print(res,type(res))
ret = json.loads(res)
print(ret,type(ret))
```

Python 正则中 search 和 match 的区别

> match() 从第一个字符开始找, 如果第一个字符就不匹配就返回 None, 不继续匹配. 用于判断字符串开头或整个字符串是否匹配,速度快。
> search() 会整个字符串查找,直到找到一个匹配。

Python 中读取 Excel 文件的方法?

```
import pandas
read_excel = pandas.read_excel("test.xlsx")
print(read_excel)
```

输入日期，判断这一天是这一年的第几天?

```
import datetime
def dayofyear():
    year = input("请输入年份: ")
    month = input("请输入月份: ")
    day = input("请输入天: ")
    date1 =
datetime.date(year=int(year),month=int(month),day=int(day))
    date2 = datetime.date(year=int(year),month=1,day=1)
    return (date1-date2).days+1
```

什么是 lambda 函数？有什么好处?

lambda 函数是一个可以接收任意多个参数(包括可选参数)并且返回单个表达式值的匿名函数

好处：
1、**lambda** 函数比较轻便，即用即删除，很适合需要完成一项功能，但是此功能只在此一处使用，
连名字都很随意的情况下；
2、匿名函数，一般用来给 filter， map 这样的函数式编程服务；
3、作为回调函数，传递给某些应用，比如消息处理

求两个列表的交集、差集、并集?

```
a= [1,2,3,4]
b= [4,3,5,6]
jj1=[i for i in a if i in b]    #在a中的i，并且也在b中，就是交集
jj2=list(set(a).intersection(set(b)))
bj1=list(set(a).union(set(b)))
cj1=list(set(b).difference(set(a)))
cj2=list(set(a).difference(set(b)))
print("交集" ,jj1)
print("交集",jj2)
print("并集",bj1)
print("差集" ,cj1)
print("差集" ,cj2)
```

什么是负索引?

与正索引不同，负索引是从右边开始检索

正则匹配不是以4和7结尾的手机号?

```
import re
tels=["13100001234","18912344321","10086","18800007777"]
for tel in tels:
    ret=re.match("1\d{9}[0-3,5-6,8-9]",tel)
    if ret:
        print("结果是：",ret.group())
    else:
        print("%s不是想要的手机号" % tel)
```

用两种方法去空格?

```
str="hello world ha ha"
res=str.replace(" ","")
print(res)

list=str.split(" ")
res="".join(list)
print(res)
```

均输出：

```
helloworldhaha
```

统计字符串中某字符出现次数?

```
str="张三 哈哈 张三 呵呵 张三"
res=str.count("张三")
print(res)
```

正则表达式匹配 URL

```
    import re
    pattern = re.compile(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-
_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')    # 匹配模式

    string = 'Its after 12 noon, do you know where your rooftops
are? https://blog.gaozhe.top '
    url = re.findall(pattern,string)
    print(url)
```

正则匹配以 163.com 结尾的邮箱?

```
    import re
    email_list= ["sdgaozhe@163.com","xiaoWang@163.comheihei",
".com.602556194g@qq.com" ]
    for email in email_list:
        ret = re.match("[\w]{4,20}@163\.com$",email)
        if ret:
            print("%s 是符合规定的邮件地址，匹配后结果是:%s" %
(email,ret.group()))
        else:
            print("%s 不符合要求" % email)
```

s="info:xiaoZhang 33 shandong",用正则切分字符串输出 ['info', 'xiaoZhang', '33', 'shandong']

```
    import re
    s="info:xiaoZhang 33 shandong"
    res=re.split(r":| ",s)          #|表示或，根据冒号或者空格切分
    print(res)
```

两个有序列表，l1,l2，对这两个列表进行合并不可使用 extend

```
    def loop_merge_sort(l1,l2):
        tmp = []
        while len(l1)>0 and len(l2)>0:
            if l1[0] <l2[0]:
                tmp.append(l1[0])
                del l1[0]
            else:
                tmp.append(l2[0])
                del l2[0]
```

代码描述列表推导式、字典推导式、生成器?

```
import random
td_list=[i for i in range(10)]
print("列表推导式", td_list, type(td_list))
ge_list = (i for i in range(10))
print("生成器", ge_list)
dic = {k:random.randint(4, 9)for k in ["a", "b", "c", "d"]}
print("字典推导式",dic,type(dic))
```

根据键对字典排序，不可使用zip?

```
dic = {"name":"zs","sex":"man" ,"city":"bj"}
print(dic.items())
b= sorted(dic.items(),key= lambda x:x[0])
print("根据键排序",b)
new_dic = {i[0]:i[1] for i in b}
print("字典推导式构造新字典",new_dic)
```

阅读一下代码他们的输出结果是什么?

```
def multi():
    return [lambda x : i*x for i in range(4)]
print([m(3) for m in multi()])  # [9,9,9,9]
```

代码实现 Python 的线程同步

```
import threading
import time

def thread():
    time.sleep(2)
    print('---子线程结束---')

def main():
    t1 = threading.Thread(target=thread)
    t1.start()
    print('---主线程--结束')

if __name__ =='__main__':
    main()
#执行结果
---主线程--结束
---子线程结束---
```

简述 read、readline、readlines 的区别?

```
read 读取整个文件
readline 读取下一行,使用生成器方法
readlines 读取整个文件到一个迭代器以供我们遍历
```

a = ” hehheh “，去除收尾空格?

```
a="  hehheh  "
print(a.strip())
```

yield 用法

```
yield 就是保存当前程序执行状态。你用 for 循环的时候，每次取一个元素的时候就会计算一次。用 yield 的函数叫 generator,和 iterator 一样，它的好处是不用一次计算所有元素，而是用一次算一次，可以节省很多空间，generator 每次计算需要上一次计算结果，所以用 yield,否则一 return，上次计算结果就没了。
```

字符串 “123” 转换成 123，不使用内置 API，例如 int()

```
def atoi(s):
    num = 0
    for v in s:
        t = "%s * 1" % v
        n = eval(t)
        num = num * 10 + n
    return num
```

is 和 == 的区别

```
# is 比较的是内存地址    == 比较内容和数据类型

a = [1, 2, 3]
b = a
print(a is b)
print(a == b)

c = copy.deepcopy(a)
print(a is c)
print(a == c)
-------------
True
True
False
True
```

有没有一个工具可以帮助查找 python 的 bug 和进行静态的代码分析?

> PyChecker 是一个 python 代码的静态分析工具，它可以帮助查找 python 代码的 bug，会对代码的复杂度和格式提出警告
> Pylint 是另外一个工具可以进行 codingstandard 检查

文件递归

```
def print_directory_contents(sPath):
"""
这个函数接收文件夹的名称作为输入参数
返回该文件夹中文件的路径
以及其包含文件夹中文件的路径
"""
import os
for s_child in os.listdir(s_path):
    s_child_path = os.path.join(s_path, s_child)
    if os.path.isdir(s_child_path):
        print_directory_contents(s_child_path)
    else:
        print(s_child_path)
```

Python 如何 copy 一个文件?

> shutil 模块有一个 copyfile 函数可以实现文件拷贝

打乱一个排好序的 list 对象 alist?

```
import random
alist = [1,2,3,4,5]
random.shuffle(alist)
print(alist)
```

对 s=”hello”进行反转

```
s="hello"
print(s[::-1])
```

Python 中单下划线和双下划线使用

__foo__:一种约定,Python内部的名字,用来区别其他用户自定义的命名,以防冲突，就是例如__init__(),__del__(),__call__()这些特殊方法

_foo:一种约定,用来指定变量私有.程序员用来指定私有变量的一种方式.不能用from module import * 导入，其他方面和公有一样访问;

__foo:这个有真正的意义:解析器用_classname__foo来代替这个名字,以区别和其他类

相同的命名,它无法直接像公有成员一样随便访问,通过对象名._类名__xxx这样的方式可以访问.

反转一个整数

```
class Solution(object):
    def reverse(self,x):
        if -10<x<10:
            return x
        str_x = str(x)
        if str_x[0] !="-":
            str_x = str_x[::-1]
            x = int(str_x)
        else:
            str_x = str_x[1:][::-1]
            x = int(str_x)
            x = -x
        return x if -2147483648<x<2147483647 else 0
if __name__ == '__main__':
    s = Solution()
    reverse_int = s.reverse(-120)
    print(reverse_int)
```

代码描述静态方法 (staticmethod)，类方法(classmethod) 和实例方法

```
def foo(x):
    print "executing foo(%s)"%(x)

class A(object):
    def foo(self,x):
        print "executing foo(%s,%s)"%(self,x)

    @classmethod
    def class_foo(cls,x):
        print "executing class_foo(%s,%s)"%(cls,x)

    @staticmethod
    def static_foo(x):
        print "executing static_foo(%s)"%x

a=A()
```

新式类和旧式类的区别?

a. 在python里凡是继承了object的类，都是新式类
b. Python3里只有新式类
c. Python2里面继承object的是新式类，没有写父类的是经典类
d. 经典类目前在Python里基本没有应用

请写出一个在函数执行后输出日志的装饰器

```
def do_log(func):
    @wraps(func)
    def wrapper(*args, **kw):
        if func.__name__ == "debug":
            msg = "debug {}".format(args[0])
        elif func.__name__ == "info":
            msg = "info {}".format(args[0])
        else:
            msg =  "unknown {}".format(args[0])
        return func(msg, **kw)
    return wrapper

@do_log
def debug(msg):
    print(msg)

@do_log
def info(msg):
    print(msg)

if __name__ == "__main__":
    debug("123")
    info("abc")
```

请解释一下协程的优点

子程序切换不是线程切换，而是由程序自身控制
没有线程切换的开销，和多线程比，线程数量越多，协程的性能优势就越明显
不需要多线程的锁机制，因为只有一个线程，也不存在同时写变量冲突，在协程中控制共享资源不加锁

闭包必须满足那几点

1.必须有一个内嵌函数
2.内嵌函数必须引用外部函数中的变量
3.外部函数的返回值必须是内嵌函数

简历制作与面试技巧

简历中的常见错误

信息过多，缺乏重点

XXX

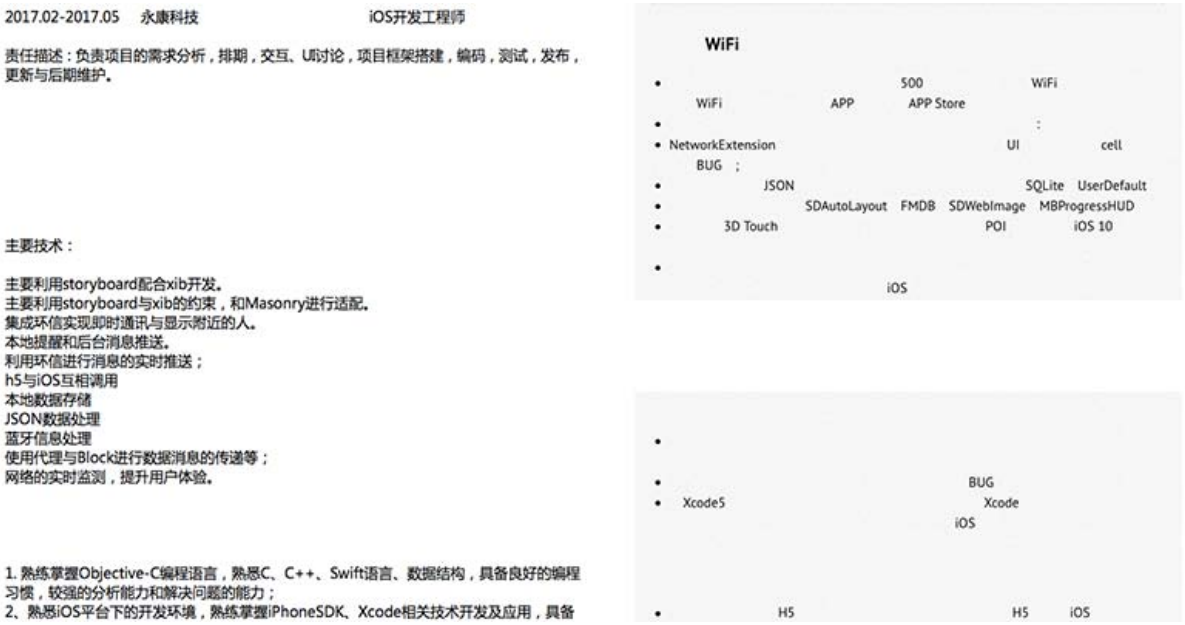
xxx学校 | xxx专业

1. 熟练掌握Objective-C编程语言，熟悉C、C++、Swift语言、数据结构，具备良好的编程习惯，较强的分析能力和解决问题的能力；
2、熟悉iOS平台下的开发环境，熟练掌握iPhoneSDK、Xcode相关技术开发及应用，具备独立完成项目开发的能力；
3、熟练掌握app发布流程，并成功发布多款app；
4、熟练掌握常用的面向对象设计模式，如MVC、代理模式、观察者模式、单例模式等等；
5、熟悉iOS平台的内存管理机制，能够使用ARC与MRC模式开发；
6、熟练掌握Runtime消息机制，熟悉RunLoop机制提高代码运行效率；
7、熟练掌握控制器的生命周期、App的生命周期，运用于项目开发之中；
8、熟练掌握各种UI控件，以及能够自定义控件，精通UI界面的搭建，熟悉Xib及StoryBoard的使用；
9、熟练掌握GET、POST等HTTP/ HTTPS协议请求方式，能够结合AFNetworking、NSURLConnection/ NSURLSession完成数据请求；
10、熟练掌握TCP/IP、UDP的Socket通信的原理以及使用AsyncSocket实现套接字的应用
11、熟练掌握原生的代码约束和AutoLayout约束，使用Masonry、SDAutoLayout实现屏幕自适应；
12、熟练掌握断点调试和View Hierarchy调试；
13、熟悉掌握本地数据存储技术；
14、熟练掌握app发布流程，并成功发布多款app；
15、书籍掌握蓝牙信息处理技术；
16、使用代理与Block进行数据消息的传递等；
17、熟悉七牛推流SDK、连麦SDK、PLPlayer，并使用集成了一款直播app；
18、熟悉原生地图、高德地图和百度地图的集成和开发，能够按项目需求完成定位以及地图指引功能；
19、熟悉QQ、微信、新浪微博等第三方登陆、分享的集成和开发；
20、熟悉支付宝支付、微信支付、银联支付等第三方支付的集成和开发，并应用于实际开发中；

无意义的描述

> XXX 平台
> 根据项目任务要求完成爬虫模块，前端展示模块，完成数据整理与入库功能。
>
> XXX 平台
> 构建 IP 代理池，使用 Scrapy 开发异步爬虫系统，优化爬虫策略和防屏蔽规则，提升 200% 网页抓取速度。
> 使用 Vue 框架完成后台管理系统，实现自定义分页，第三方登录等 6个 主要功能。
> 负责数据的清洗与存储到 MySQL 数据库，使用数据库索引减少 50% 数据查询时间。

排版杂乱，错别字多



2017.02-2017.05　永康科技　　iOS开发工程师

责任描述：负责项目的需求分析，排期，交互、UI讨论，项目框架搭建，编码，测试，发布，更新与后期维护。

主要技术：

主要利用storyboard配合xib开发。
主要利用storyboard与xib的约束，和Masonry进行适配。
集成环信实现即时通讯与显示附近的人。
本地提醒和后台消息推送。
利用环信进行消息的实时推送；
h5与iOS互相调用
本地数据存储
JSON数据处理
蓝牙信息处理
使用代理与Block进行数据消息的传递等；
网络的实时监测，提升用户体验。

1. 熟练掌握Objective-C编程语言，熟悉C、C++、Swift语言、数据结构，具备良好的编程习惯，较强的分析能力和解决问题的能力；
2、熟悉iOS平台下的开发环境，熟练掌握iPhoneSDK、Xcode相关技术开发及应用，具备

WiFi

- 500 WiFi
  WiFi APP APP Store
- :
- NetworkExtension UI cell
  BUG ；
- JSON SQLite UserDefault
- SDAutoLayout FMDB SDWebImage MBProgressHUD
- 3D Touch POI iOS 10
- iOS

- BUG
- Xcode5 Xcode
  iOS

- H5 H5 iOS

如何写一份更好的简历

基本信息	工作经历	项目经验	教育情况	其他
姓名	公司名称	项目名称	学校名称	兴趣爱好
邮箱	岗位	项目介绍	专业	志愿者工作
电话	主要职责	主要职责	就读年份	
Github/博客	在职时间	项目时间	获奖情况	
个人简介	项目经验			
技能列表				

综合起来

杨勇

两年Python爬虫系统开发经验，熟悉**Scrapy框架**，作为主力工程师参与设计与开发过多个项目，负责系统核心模块的开发，自动化测试与部署。**熟悉HTTP协议、TCP/IP协议，正则表达式，XPATH**的用法，了解**Redis，MySQL数据库**与Linux系统的常见机制与原理。

基本信息

contact@jobder.net

133-5555-6666

github.com/Windsooon

求职意向：爬虫工程师

政治面貌：群众

教育经历

中山大学
计算机科学 | 2013年- 2017年

操作系统(85分/专业排名18/100)，数据结构(90分/专业排名10/100)

2015-2016学年获得美国大学生数学建模竞赛一等奖

志愿工作

协助组织翻译Flask，Requests第一版本文档，翻译多篇技术文章。

Github上的开源项目Cherry获得200个Star。

Django

Vue

Docker

Linux

团队能力

工作经历

独角科技有限公司（旗下产品EngineGo）

2014年6月-2016年6月 | 高级Python工程师

- 作为组长负责设计和开发**分布式网络爬虫系统**，优化爬虫策略和防屏蔽规则，提升网页抓取的效率和质量。
- 根据行业需求分析设计方案可行性，对项目代码进行测试优化，**协助持续集成与自动化部署**，提高系统可用性。
- 负责公司技术文档的编写以及维护，定期review团队的代码，与团队共同学习以及成长。

EngineGo爬虫系统

- 与产品经理保持沟通，使用**Scrapy框架**对爬虫模块进行重构，提高**200%爬虫速度**并减少服务器**20%CPU**负载。改进爬虫策略，降低**40%**被屏蔽的请求数。
- 作为主要工程师设计以及开发**物业模块，活动模块，实现报名，即时通知**等10个功能。
- 使用**Docker**对项目进行拆分重新架构，减少业务模块之间的资源耦合，实现**持续集成与自动化部署**。

未来科技有限公司

2012年6月-2014年6月 | Python工程师

- 使用Python爬虫帮助公司进行电商数据的**数据采集，文本分析与文本标注**。并将就分析结果与开发人员讨论产品方向。
- 带领3人团队完成后端系统的难点分析与架构设计工作，承担核心功能代码编写，开发与维护系统核心模块。使用selenium等工具对产品进行多方面测试，保证代码的可靠，安全。

内部后台管理系统

- 基于**Flask**实现RESTful风格的后台管理系统，使用**MySQL**作为存储数据，配置主从热备，使用**Celery**实现异步邮件推送。
- 使用**Redis**存储定时任务和缓存结果，选择**RabbitMQ**做消息队列服务，开发，测试**登录注册，请假审批**等7个模块。
- 采用**Python，Shell**等脚步语言公司多个办公项目的自动化，大幅提高公司内部项目效率。

面试技巧

好的自我介绍决定了面试的 80%

不管你相不相信，你适不适合这份工作，HR 在你自我介绍的阶段，已经基本决定了。很多人在自我介绍时会犯一个错误，那就是把自己的学校情况，工作经历，兴趣爱好笼统的丢给 HR，让他自己判断你适不适合这份工作。如果你这么做了，恭喜你，你已经进入了 HR 心中的“平庸组”名单。那么，自我介绍到底应该说些什么呢？你应该斩钉截铁地告诉他：为什么这份工作非要你来做才适合！所以，每一次面试我都会告诉面试官：“我看到在招聘启事上，这份工作需要 blabla，这和我之前的工作经历中 blabla 非常类似。”

你对我们公司了解多少？

这是让很多面试者头疼的一个问题，难点就在于，大家说的都差不多，你很难在众多面试者中脱颖而出。如果你按照某度的搜索结果，只是说出这家公司的行业地位，规模，企业文化等，那这个问题就算是白问了。因为，面试官其实想问的是：为什么你非得选择我们公司？我下面要说的就是：如何通过半小时的准备，给面试官留下一个难以磨灭的印象。

这个方法叫做：概况+细节+情绪。概况不要多说，因为这点并不决定你和其他面试者的差异。你只要说出公司的地位、总部、规模等信息即可。接下来，你就要开始说细节了，这很重要！非常简单但给人印象很深的做法是，你看几个该企业的宣传片，然后用绘声绘色、深受感染的语气描绘其中的情节，比如：“我感触特别深的就是贵公司的数据审核产品，他极大了解决了人工审核慢而且容易出错的问题，具有很好的商业前景”。这样做有什么好处？就是能在最短的时间内，调动起面试官的全部情绪。注意到其中好玩的地方了吗？面试官的情绪越高涨，对面试者留下的印象也就越深。

离职原因——处处是陷阱

离职原因是一个很重要的问题，因为它考验的不但是你的工作能力和性格，更考验你的情商和智商。不讨巧的离职原因包括：工作业绩差，沟通能力差，老板傻逼，看同事不顺眼，和公司有纠纷……所以，尽量挑一些主观上无法避免的原因，比如：公司的产品质量出了问题，公司面临破产，部门被合并，亲人有重大变故，公司的氛围和你想要的相去甚远（顺便夸夸新东家）。最重要的一点是，强调现在是你最想要稳定的时期。换句话说，就是前面那些都是浮云！

你的缺点是什么？

缺点真的是一个非常不好答的问题，但是只要掌握了以下这个原则，这道问题也只是小菜一碟，那就是：避重就轻。什么是重？性格方面的问题，人际方面的问题，工作能力方面的原因。如果你说：“我的缺点就是耐心太差”，“我的缺点就是沟通能力有待提高”，那你真的是一个大傻帽。什么是轻？举点例子：我方向感不太好，不善于理财（金融岗位除外）之类的。有人会问了，我说了这些缺点，面试官会不会觉得我很虚伪？那我告诉你，只要你的虚伪不至于让他想吐（比如“我最大的缺点就是太追求完美”），那虚伪绝对要比傻乎乎的坦诚好。

你有什么想问我的吗？

一般问到这个问题，整个面试就要结束了，但是不要掉以轻心，因为最后这个问题决定了面试官对你的最终印象。

所以这个问题背后的潜台词是什么呢？那就是：你还想了解一些什么，帮助你更好地留在这个公司？换言之，就是你有多想留在这个公司？
如果你说“没有”，那么面试官说不定心里咯噔一下：原来你对这个职位兴趣也就这点啊……这个问题其实给了你表忠心的机会，你可以很认真地问她：“那如果我来到了这个公司，那每天的日常大概会是什么样的？”或者“这个公司的氛围是什么样的？”（暗示你来这里工作的强烈欲望）。

总结

好了，面试攻略就分享到这里，最后送给大家一句话：找工作就是，胆大心细脸皮厚！如果你觉得这篇免费 Chat 对你有帮助，那就转发一下吧！

给大家介绍一下Mr. Chang的学习方法。学习的三种阶段：入门、掌握、精通。

- 入门篇 – 读文章 对这个知识有一定的了解，了解大概的发展趋势以及使用复杂程-度，达到基本可以使用程度。
- 掌握篇 – 看网上的教程 对这个知识使用并了解一些出现bug解决方案，扩展使用途径。
- 精通篇 – 对知识体系结构已有完整认知 需要读别人写的书 加上自己亲自打代码时间 完全了解这个知识发展渊源。

不论你是科班毕业还是非科班毕业，但要相信努力一定有收获，不要抱怨，趁年轻，多学习，你一定会成为 ta 心目中的哪个大英雄！

送给所有有梦想的我们一句话： 相信自己，力量在心中！！

大家可以关注《Python数据结构》公众号后回复 110 获取源代码。