Python3 基础练习题精选

作者：上海-悠悠

更多 Python 自动化扫码关注**微信公众号：yoyoketang (扫二维码关注)**

个人博客地址：http://www.cnblogs.com/yoyoketang/

QQ 交流群：730246532

联系本人微信： 283340479

目录

第 1 章 字符串练习题

网易云课程-Python3 基础练习题精选

全部练习题视频课程，在网易云平台可以购买学习

https://study.163.com/course/courseMain.htm?courseId=1211387804&share=2&shareId=480000002230338

适用人群

1.python 零基础的同学

2.已经有一些基础语法，遇到练习题（看得懂）不会做

3.python 基础知识不牢固的同学

课程概述

课程简介：

1.本课程是 2021 年 3 月录制

2.以 python3.6 版本讲解常见的 python 练习题和面试题，

3.每个题目会详细讲解用到的 1-2 个 python 知识点。

4.课程目的在于基础知识的熟练掌握。

1.1 交换

已知 a 的值为"hello"，b 的值为"world"，如何交换 a 和 b 的值?

得到 a 的值为"world"，b 的值为"hello"

普通解决思路：中间变量 temp

```
a = "hello"
b = "world"
temp = a
a = b
b = temp
print(a)
print(b)
```

另外一种实现方式，可以用 python 里面支持的语法实现，Pythonic 写法

```
a = "hello"
b = "world"
a, b = b, a
print(a, b)
```

1.2 回文

回文的定义："回文" 就是正读倒读都一样的。

如奇数个："98789"，这个数字正读是"98789" 倒读也是"98789"。

偶数个数字"3223"也是回文数。

字母 "abcba" 也是回文。

判断一个字符串是否是回文字符串，是打印 True， 不是打印 False

解决思路：不用关注奇数个还是偶数个字符，直接反转字符串，判断跟反转前是不是相等的

```
a = "abcba"
# 1.切片 [::-1]
print(a == a[::-1])


# 2.reversed()  # 迭代器
print(reversed(a))  # reversed object
print("".join(reversed(a)))    # 得到字符串
print(a == "".join(reversed(a)))
```

reversed：将其元素从后向前颠倒构建成一个新的**迭代器**

reversed的英文解释

Return a reverse iterator. seq must be an object which has a __reversed__() method or supports the sequence protocol (the __len__() method and the __getitem__() method with integer arguments starting at 0).

reversed()函数的输入时任意一个序列，返回一份倒序后的序列副本。通常用于for循环需要倒序循环时。

eg:

```
seq = [1, 2, 3, 4, 5, 6, 7, 8]
for item in reversed(seq):
    print(item, end=" ")
```

结果：

8 7 6 5 4 3 2 1

reversed()的功能：翻转对象

- 翻转函数reversed()调用参数类中的__reversed__()函数。
- 函数功能是反转一个序列对象，将其元素从后向前颠倒构建成一个新的迭代器

1.3 字符串切割

已知一个字符串为 "hello_world_yoyo"，如何得到一个队列 ["hello","world","yoyo"]

split() 方法有 2 个参数

- 切割的字符
- 切割次数，默认切割全部

Python3 split()方法

Python3 字符串

描述

split() 通过指定分隔符对字符串进行切片，如果第二个参数 num 有指定值，则分割为 num+1 个子字符串。

语法

split() 方法语法：

```
str.split(str="", num=string.count(str))
```

参数

- str -- 分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等。
- num -- 分割次数。默认为 -1, 即分隔所有。

返回值

返回分割后的字符串列表。

```
a = "hello_world_yoyo"

print(a.split("_"))
```

让用户输入任意的用户名与密码，然后将他输入的用户名与 密码打印出来，如用户输入

abc/123 ，则打印

您输入的用户名是: abc

密码是:123

```
b = input("请输入用户名和密码（格式 user/password）:")

c = b.split("/")

print("您输入的用户名是: ", c[0])

print("密码是: ", c[1])
```

1.4 拼接字符串

有个列表 ["hello", "world", "yoyo"]如何把把列表里面的字符串联起来，

得到字符串 "hello_world_yoyo"

使用. join()方法拼接字符

语法： 'sep'.join(iterable)

参数说明

sep：分隔符。可以为空

iterable：可迭代对象，要连接的元素序列、字符串、元组、字典

上面的语法即：以 sep 作为分隔符，将 seq 所有的元素合并成一个新的字符串

返回值：返回一个以分隔符 sep 连接各个元素后生成的字符串

```
a = ["hello", "world", "yoyo"]


print("_".join(a))
```

1.5 替换字符

把字符串 s 中的每个空格替换成"%20"

输入：s = "We are happy."

输出："We%20are%20happy."

replace(old, new [, max])

把 将字符串中的 old 替换成 new,如果 max 指定，则替换不超过 max 次。

```
s = "We are happy."
print(s.replace(" ", "%20"))

# 只替换第 1 个空格
print(s.replace(" ", "%20", 1))
```

1.6 九九乘法表

打印 99 乘法表

%-2s 是表示左对齐，占 2 个字符位置

```
# a = range(10)  # 可迭代对象

for i in range(1, 10):
    # print(i)  # 行数
    for j in range(1, i+1):
        # print("列数: ", j)
        print("%s * %s = %-2s" % (j, i, j*i), end="   ")
    print()    # print 本身就会换行
```

1.7 字符下标

找出单词 "welcome" 在 字符串"Hello, welcome to my world." 中出现的位置，找不到返回-1

从下标 0 开始索引

str.index() 方法查找指定值的首次出现。如果找不到该值，index() 方法将引发异常。

语法 str.index(value, start, end)

```
# 语法 string.index(value, start, end)
txt = "Hello, welcome to my world."
x = txt.index("e")
print(x)  # 1

y = txt.index("welcome")
print(y)    # 7

# 如果只在位置 5 和 10 之间搜索时，字母 "e" 首次出现位置
z = txt.index("e", 5, 10)
print(z)  # 8
```

index() 方法与 find() 方法几乎相同，唯一的区别是，如果找不到该值，则 find() 方法将返回 -1

考虑找不到的情况，于是可以优化

```
"""
找出单词 "welcome" 在字符串"Hello, welcome to my world."
中出现的位置,找不到返回-1
"""
txt = "Hello, welcome to my world."
if "welcome" in txt:
    # 找到返回 index
    print(txt.index("welcome"))
else:
    # 找不到返回 -1
    print("-1")
```

1.8 统计字符出现的次数

统计字符串“Hello, welcome to my world.” 中字母 w 出现的次数

统计单词 my 出现的次数

count() 方法用于统计字符串里某个字符出现的次数。可选参数为在字符串搜索的开始与结束位置。

count()方法语法：str.count(sub, start= 0,end=len(string))

参数：

sub -- 搜索的子字符串

start -- 字符串开始搜索的位置。默认为第一个字符,第一个字符索引值为 0。

end -- 字符串中结束搜索的位置。字符中第一个字符的索引为 0。默认为字符串的最后一个位置。

```
a = "Hello, welcome to my world."


print(a.count("w"))
print(a.count("my"))
```

1.9 统计每个字符出现的次数

题目:输入一个字符串 str，输出第 m 个只出现过 n 次的字符，如在字符串 gbgkkdehh 中，找出第 2 个只出现 1 次的字符，输出结果：d

解决思路:

利用 collections 库的 Counter 方法统计字符串每个单词出现的次数

Counter 方法对字符串\列表\元祖\字典进行计数,返回一个字典类型的数据,键是元素,值是元素出现的次数

```
from collections import Counter


a = "gbgkkdehh"
m = 2    # 第 m 个
n = 1    # 出现次数 n
```

```
b = Counter(a)
print(b)
print(dict(b))


# 找出所有的出现 n 的字符
s = []
for i, j in dict(b).items():
    if j == n:   # 出现 n 次
        s.append(i)
print(s)
print(s[m-1])      # 第 m 个索引是 m-1
```

使用列表推倒式

```
from collections import Counter


a = "gbgkkdehh"
m = 2    # 第 m 个
n = 1    # 出现次数 n
# 方法 2 列表推导式
s1 = [i for i, j in dict(Counter(a)).items() if j == n]
print(s1[m-1])
```

1.10 判断字符 a 含 b

判断字符串 a="welcome to my world" 是否包含单词 b="world"

包含返回 True，不包含返回 False

解题思路：

find(str, beg=0, end=len(string))

检测 str 是否包含在字符串中，如果指定范围 beg 和 end ，则检查是否包含在指定范围内，如果包含返回开始的索引值，否则返回-1

index(str, beg=0, end=len(string))

跟 find()方法一样，只不过如果 str 不在字符串中会报一个异常。

```
a = "welcome to my world"
# 存在返回 index 位置，不存在返回-1
print(a.find("world"))
if a.find("world") == -1:
    print(False)
else:
    print(True)
# 三元表达式
print(False if a.find("world") == -1 else True)
```

逻辑运算符 a in b

```
a = "welcome to my world"

b = "world"


print(b in a)
```

1.11 查找字符首次出现位置

输出指定字符串 A 在字符串 B 中第一次出现的位置,如果 B 中不包含 A,则输出-1

从 0 开始计数

A = "hello"

B = "hi how are you hello world, hello yoyo !"

find(str, beg=0, end=len(string))

检测 str 是否包含在字符串中，如果指定范围 beg 和 end ，则检查是否包含在指定范围内，如果包含返回开始的索引值，否则返回-1

```
A = "hello"

B = "hi how are you hello world, hello yoyo !"


print(B.find(A))
```

1.12 查找字符串最后一次出现位置

输出指定字符串 A 在字符串 B 中最后出现的位置,如果 B 中不包含 A,则输出-1

从 0 开始计数

A = "hello"

B = "hi how are you hello world, hello yoyo !"

rfind() 返回字符串最后一次出现的位置，如果没有匹配项则返回 -1

rfind()方法语法：str.rfind(str, beg=0 end=len(string))

参数

str -- 查找的字符串

beg -- 开始查找的位置，默认为 0

end -- 结束查找位置，默认为字符串的长度。

返回值

返回字符串最后一次出现的位置，如果没有匹配项则返回-1。

```
A = "hello"
B = "hi how are you hello world, hello yoyo !"


print(B.rfind(A))
```

1.13 判断奇数偶数

给定一个数 a，判断一个数字是否为奇数或偶数

Python算术运算符

以下假设变量： a=10，b=20：

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 30
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除 - x除以y	b / a 输出结果 2
%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回x的y次幂	a**b 为10的20次方， 输出结果 100000000000000000000

//	取整除 - 返回商的整数部分（**向下取整**）	`>>> 9//2`
		`4`
		`>>> -9//2`
		`-5`

```
a = 12

# 1.if 判断%取余数

if a % 2 == 0:

    print("a 是偶数")

else:

    print("a 是奇数")


# 2.三元表达式

print("a 是偶数" if a % 2 == 0 else "a 是奇数")
```

1.14 判断一个姓名是否姓王

输入一个姓名，判断是否姓王

startswith() 方法用于检查字符串是否是以指定子字符串开头，如果是则返回 True，否则返回 False。如果参数 beg 和 end 指定值，则在指定范围内检查。

startswith()方法语法：str.startswith(str, beg=0,end=len(string));

参数

str -- 检测的字符串。

strbeg -- 可选参数用于设置字符串检测的起始位置。

strend -- 可选参数用于设置字符串检测的结束位置。

返回值

如果检测到字符串则返回 True，否则返回 False。

```
b = input("输入姓名: ")
# 判断字符串以 xx 开始
if str(b).startswith("王"):
    print("姓王")
else:
    print("不姓王")
```

endswith() 方法用于判断字符串是否以指定后缀结尾，如果以指定后缀结尾返回 True，否则返回 False。可选参数"start"与"end"为检索字符串的开始与结束位置。

endswith()方法语法：str.endswith(suffix[, start[, end]])

参数

suffix -- 该参数可以是一个字符串或者是一个元素。

start -- 字符串中的开始位置。

end -- 字符中结束位置。

返回值

如果字符串含有指定的后缀返回 True，否则返回 False。

```
b = input("输入姓名: ")


# 判断字符串以 xx 结尾
if str(b).endswith("三"):
    print("姓名以三结尾")
else:
    print("姓名不以三结尾")
```

1.15 判断是不是纯数字

如何判断一个字符串是不是纯数字组成

isdigit() 方法检测字符串是否只由数字组成。如果字符串只包含数字则返回 True 否则返回 False.

```
a = "123456"   # Only digit in this string
print(a.isdigit())


b = "this is string example....wow!!!"
print(b.isdigit())
```

int() 函数用于将一个字符串或数字转换为整型。如果字符串不是数字类型抛异常

参数

x -- 字符串或数字。

base -- 进制数，默认十进制。

返回值

返回整型数据。

```
a = "123"
b = "ab123"


# try 判断是否抛异常
try:
    int(a)
    print(True)
except:
    print(False)
```

1.16 字符串大小写转换

将字符串 a = "This is string example....wow!" 全部转成大写

字符串 b = "Welcome To My World" 全部转成小写

upper() 方法将字符串中的小写字母转为大写字母。

```
a = "This is string example....wow!"

print(a.upper())
```

lower() 方法转换字符串中所有大写字符为小写。

```
b = "Welcome To My World"

print(b.lower())
```

1.17 字符串去掉首尾空格

将字符串 a = " welcome to my world "首尾空格去掉

strip() 方法用于移除字符串头尾指定的字符（默认为空格）或字符序列。

strip()方法语法：str.strip([chars]);

参数

chars -- 移除字符串头尾指定的字符序列。

返回值

返回移除字符串头尾指定的字符序列生成的新字符串。

```
a = "  welcome to my world   "

print(a.strip())

# 去掉结尾处指定的!

b = "welcome to my world!"

print(b.strip("!"))
```

1.18 字符串去掉左边指定空格或字符

将字符串 a = "　welcome to my world ！"左边的空格去掉

lstrip() 方法用于截掉字符串左边的空格或指定字符。

lstrip()方法语法：str.lstrip([chars])

参数

chars --指定截取的字符。

返回值

返回截掉字符串左边的空格或指定字符后生成的新字符串。

```
# 只去除左边的空格
a = "  welcome to my world !  "
print(a.lstrip())


# 只去除左边的!
b = "!!welcome to my world !"
print(b.lstrip("!"))
```

1.19 字符串去掉右边指定空格或字符

将字符串 a = "　welcome to my world！　"右边的空格去掉

rstrip() 删除 string 字符串末尾的指定字符（默认为空格）.

rstrip()方法语法：str.rstrip([chars])

参数

chars -- 指定删除的字符（默认为空格）

返回值

返回删除 string 字符串末尾的指定字符后生成的新字符串。

```
# 只去除右边的空格
a = "  welcome to my world !  "
print(a.rstrip())

# 只去除右边的!
b = "!!welcome to my world !"
print(b.rstrip("!"))
```

1.20 去除字符串里面所有的空格

将字符串 a = " welcome to my world ! "里面的所有空格都去掉

replace() 方法把字符串中的 old（旧字符串） 替换成 new(新字符串)，如果指定第三个参数 max，则替换不超过 max 次。

replace()方法语法：str.replace(old, new[, max])

参数

old -- 将被替换的子字符串。

new -- 新字符串，用于替换 old 子字符串。

max -- 可选字符串，替换不超过 max 次

返回值

返回字符串中的 old（旧字符串） 替换成 new(新字符串)后生成的新字符串，如果指定第三个参数 max，则替换不超过 max 次。

```
# 替换空格
a = "  welcome to my world !  "
print(a.replace(" ", ""))

# 也可以用 split()切割后合并
print("".join(a.split(" ")))
```

1.21 字符串去重后排序

s = "ajldjlajfdljfddd"，去重并从小到大排序输出"adfjl"

```
"""去重-普通解决思路"""
s = "ajldjlajfdljfddd"

s1 = []
for i in s:
    if i not in s1:
```

```
        s1.append(i)
print(s1)


# 排序
print("".join(sorted(s1)))
```

集合（set）是一个无序的**不重复元素**序列。

可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。

sorted() 是 python 里面的一个内建函数,用于排序

参数说明

iterable 可迭代对象，如：str、list、tuple、dict 都是可迭代对象（这里就不局限于 list 了）

key 用列表元素的某个属性或函数进行作为关键字（此函数只能有一个参数）

reverse 排序规则. reverse = True 降序或者 reverse = False 升序，默认升序

return 有返回值，返回新的队列

```
s = "ajldjlajfdljfddd"
# set 集合去重
print(set(s))
# 拼接字符
print("".join(set(s)))
```

```
# 排序
b = "".join(set(s))
print("".join(sorted(b)))
```

1.22 字符串去重保留顺序

s = "ajldjlajfdljfddd"，去重保留原来的顺序，输出"adfjl"

集合（set）是一个**无序的**不重复元素序列。

sorted() 是 python 里面的一个内建函数，用于排序

```
s = "ajldjlajfdljfddd"
# set 集合去重  无序的
print(set(s))
# 拼接字符
print("".join(set(s)))
# 按原来的顺序 s.index() 索引排序
b = "".join(set(s))
print("".join(sorted(b, key=lambda x: s.index(x))))
```

1.23 打印菱形图案

题目 打印出如下图案（菱形）：

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

```
line = 7


# 菱形的一半
lines = (line+1)//2


# 每行的 star 数量
stars = [1+2*i for i in range(0, lines)] + [2*j-1 for j
in range(lines-1, 0,-1)]
# print(stars)


# 根据每行 star 数量和行数，算出空格个数
for star in stars:
    kong = (line-star)//2
    print(" "*kong + "*"*star)
```

1.24 输入一个正整数，判断是几位数

题目 给一个不多于 5 位的正整数，要求：

一、求它是几位数，

二、逆序打印出各位数字。

len() 方法返回对象（字符、列表、元组等）长度或项目个数。

len()方法语法：len(s)

参数

s -- 对象。

返回值

返回对象长度。

```
a = 12345


print("a 的位数是：", len(str(a)))
print("逆序打印：", str(a)[::-1])
```

判断字符相关

1. **isupper()** 方法检测字符串中所有的字母是否都为大写。只判断 a-z 的英文字符是不是全部大写 A-Z

```
a = "hello world"
print(a.isupper())   # False
b = "HELLO WORLD"
print(b.isupper())   # True
c = "Hello"
```

```
print(c.isupper())   # False
d = "HELL 123!张三"
print(d.isupper())   # True
```

2. **islower()** 方法检测字符串是否由小写字母组成。

```
a = "hello"
print(a.islower())  # True
b = "Hello"
print(b.islower())  # False
```

3. **istitle()** 方法检测字符串中所有的单词拼写首字母是否为大写，且其他字母为小写。

```
a = "hello"
print(a.istitle())  # False
b = "Hello"
print(b.istitle())  # True
```

4. **isalnum()** 方法检测字符串是否由字母和数字组成。判断 string 至少有一个字符并且所有字符都是字母或数字则返回 True,否则返回 False

```
a = "hello"
print(a.isalnum())  # True
b = "Hello123"
print(b.isalnum())  # True
c = "123"
```

```
print(b.isalnum())  # True
b = "hello 123"
print(b.isalnum())  # False
```

5. **isalpha()** 方法检测字符串是否只由字母组成。

6. **isdigit()** 方法检测字符串是否只由数字组成。

第 2 章 小学数学题

2.1.水仙花数

如果一个 3 位数等于其各位数字的立方和，则称这个数为水仙花数。

例如：153 = 1^3 + 5^3 + 3^3，因此 153 就是一个水仙花数

那么问题来了，求 1000 以内的水仙花数（3 位数）

```
for a in range(100, 1000):
    s = sum([int(i)**3 for i in str(a)])
    if s == a:
        print(a)
```

2.2 完全数

如果一个正整数等于除它本身之外其他所有除数之和，就称之为完全数。

例如：6 是完全数，因为 6 = 1+2+3；

下一个完全数是 28 = 14+7+4+2+1。

求 1000 以下的完全数

```
for i in range(1, 1000):
    s = []
    for j in range(1, i):
        if i % j == 0:
            s.append(j)
    if sum(s) == i:
        print(i)
```

2.3 数字 1+2+3...+100 求和

求 1+2+3...+100 和

```
# 1.累加
s = 0
a = 1
while a < 101:
    s += a    # 等同于 s = s+a
    a += 1    # 自增+1
print(s)
```

使用 range()函数生成 1-100 的数字

```
# 2.range(101)
s1 = 0
for i in range(101):
    s1 += i
print(s1)
```

用一行代码解决

```
# 3.用一行代码
print(sum(range(101)))
```

2.4 计算求 1-2+3-4+5-...-100 的值

奇数相加，偶数相减

```
a = 1
s = 0
while a <= 100:
    # 判断奇偶
    if a % 2 == 1:
        s += a   # 等价 s = s + a
    else:
        s -= a
```

```
    a += 1    # 自增+1(类似于 i++)  等价于 a = a+1
print(s)
```

也可以用列表推导式

```
print(sum([1*j if j % 2 else -1*j for j in range(1, 101)]))
```

2.5 计算求 1+2-3+4-5... ...100 的值

这一题有点需要注意的是第一个数字 1 是正数，后面的数都是奇数相减，偶数相加

```
a = 1
s = 0
while a <= 100:
    # 判断奇偶
    if a % 2 == 0 or a == 1:
        s += a   # 等价 s = s + a
    else:
        s -= a
    a += 1    # 自增+1(类似于 i++)  等价于 a = a+1
print(s)
```

2.6 计算 1-n 之间的所有 5 的倍数之和

定义一个函数：计算 1-n 之间的所有 5 的倍数之和，默认计算 1-100 （ n 是 一个整

数)

```
a = 1
n = 100
s = 0
while a <= n:
    # 判断是不是 5 的倍数
    if a % 5 == 0:
        s += a
    a += 1   # 自增+1(类似于 i++)  等价于 a = a+1
print(s)
```

列表推导式一行代码解决

```
n = 100
print(sum([j for j in range(1, n+1) if j % 5 == 0]))
```

2.7 n 个自然数的立方和

计算公式 $1^3 + 2^3 + 3^3 + 4^3 + ...+ n^3$

实现要求:

输入 : n = 5

输出 : 225

对应的公式 : $1^3 + 2^3 + 3^3 + 4^3 + 5^3 = 225$

Python 里面计算一个数的立方有 2 种方式：

使用运算符号**，如：a**3

使用 pow()函数，如：pow(a,3)

```
# 1.累加
n = int(input("输入一个数字:"))
s = 0
a = 1
while a <= n:
    s += a**3    # 等同于 s = s+a
    a += 1    # 自增+1
print(s)
```

使用 range()

```
# 2.range()
n = int(input("输入一个数字:"))
s1 = 0
for i in range(n+1):
    s1 += i**3
print(s1)
```

使用一行代码求和

```
# 3.用一行代码
n = int(input("输入一个数字:"))
print(sum([pow(i, 3) for i in range(n+1)]))
```

2.8 阶乘 10!

阶乘的意思: *10!*=10x9x8x7x6x5x4x3x2x1

求 10!

```
n = 10
s = 1
while n >= 1:
    s *= n
    n -= 1   # 自减，每次-1
print(s)
```

使用 range()函数

```
n = 10
s = 1
for i in range(1, n+1):
    s *= i
print(s)
```

2.9 求 1+2!+3!+...+10!的和

跟上面一题思路差不多，多一个累加求和

```
"""
计算 1+2! +3! ...+10!
"""
s = 0
a = 1
for i in range(1, 11):
    a *= i       # 得到 i 的阶乘值
    print("%s 的阶乘值: %s" % (i, a))
    s += a       # 累加
print(s)
```

2.10 求 s=a+aa+aaa+aaaa+aa...a 的值

```
"""
求 s=a+aa+aaa+aaaa+aa...a 的值
如: n = 5  a = 3
33333 = 3*10**4+ 3*10**3+ 3*10**2 + 3*10**1 +3*10**0
"""
a = 3           # a 是数字
n = 5           # n 是位数，5 位数
```

```
next_a = 0
s = 0      # s 是求和的累加值
for i in range(1, n+1):
    next_a += a*10**(i-1)
    print(next_a)
    s += next_a
print(s)
```

另外一种解决思路，把数字当字符串处理，再转 int 类型，更简单一些

```
a = sum([int('3' * i) for i in range(1, 6)])
print(a)
```

2.11 斐波那契数列

已知一个数列：1、1、2、3、5、8、13、。。。。的规律为从 3 开始的每一项都等于其前两项的和，这是斐波那契数列。

求满足规律的 100 以内的所有数据

```
num = 100
a = 0
b = 1
n = 1
print(n, end=" ")
```

```
while n <= num:
    n = a+b
    if n <= num:
        print(n, end=" ")
        a, b = b, n
    else:
        break
```

第 3 章 列表练习题

3.1 反转（判断对称）

如何判断一个数组是对称数组：

要求：判断数组元素是否对称。例如[1，2，0，2，1]，[1，2，3，3，2，1]这样的都是对称数组

用 Python 代码判断，是对称数组打印 True，不是打印 False,如：

x = [1, "a", 0, "2", 0, "a", 1]

```
"""
用 Python 代码判断，是对称数组打印 True，不是打印 False,如：
x = [1, "a",  0, "2", 0, "a", 1]
"""
x = [1, "a",  0, "2", 0, "a", 1]
```

```
# 方法 1：切片
print(x == x[::-1])

# 方法 2：反转函数 reversed()
print(reversed(x))  # 返回可迭代对象
print(x == list(reversed(x)))

# 方法 3：深拷贝
import copy
b = copy.deepcopy(x)  # 深拷贝
x.reverse()           # 对 x 反转
print(x == b)
```

3.2 列表切片

如果有一个列表 a=[1,3,5,7,11]

问题：1 如何让它反转成[11,7,5,3,1]

2.取到奇数位值的数字，如[1,5,11]

```
a = [1, 3, 5, 7, 11]
# 1.切片反转
```

```
print(a[::-1])
print(a[::2])
```

反转的时候 a.reverse() 不会生成新的列表，reversed(a) 会生成新的列表

```
# 2.反转列表
a.reverse()    # 不会生成新列表
print(a)


# 3.reversed()函数
b = reversed(a)      # 生成新的列表 b
print(list(b))
```

3.3 列表大小排序

问题：对列表 a 中的数字从小到大排序

a = [1, 6, 8, 11, 9, 1, 8, 6, 8, 7, 8]

python 的排序有两个方法，一个是 list 对象的 sort 方法，另外一个是 builtin 函数里面 sorted，主要区别：

sort 仅针对于 list 对象排序，无返回值，会改变原来队列顺序

sorted 是一个单独函数，可以对可迭代（iteration）对象排序，不局限于 list，它不改变原生数据，重新生成一个新的队列

sorted()参数说明

iterable 可迭代对象，如：str、list、tuple、dict 都是可迭代对象（这里就不局限于 list 了）

key 用列表元素的某个属性或函数进行作为关键字（此函数只能有一个参数）

reverse 排序规则. reverse = True 降序或者 reverse = False 升序，默认升序

return 有返回值值，返回新的队列

```
a = [1, 6, 8, 11, 9, 1, 8, 6, 8, 7, 8]


# 1.排序
a.sort()
print(a)


# 2.sorted()排序
print(sorted(a))
```

3.4 取出最大值最小值

L1 = [1, 2, 3, 11, 2, 5, 3, 2, 5, 33, 88]

找出列表中最大值和最小值

min()和 max()函数是去除最小值和最大值

```
L1 = [1, 2, 3, 11, 2, 5, 3, 2, 5, 33, 88]
print(min(L1))    # 最小值
print(max(L1))    # 最大值
```

3.5 找出列表中单词最长的一个

a = ["hello", "world", "yoyo", "congratulations"]

找出列表中单词最长的一个

```
a = ["hello", "world", "yoyo", "congratulations"]
# 找出字符串最短和最长的值
print(min(a, key=lambda x:len(x)))
print(max(a, key=lambda x:len(x)))
```

3.6 切片取出列表中最大的三个数

取出列表中最大的三个值

L1 = [1, 2, 3, 11, 2, 5, 3, 2, 5, 33, 88]

```
L1 = [1, 2, 3, 11, 2, 5, 3, 2, 5, 33, 88]
# 排序后取倒数 3 个
print(sorted(L1)[-3:])
```

```
# 倒叙后取前 3 个
print(sorted(L1, reverse=True)[:3])
```

3.7 列表按绝对值排序

a = [1, -6, 2, -5, 9, 4, 20, -3] 按列表中的数字绝对值从小到大排序

sorted()参数说明

iterable 可迭代对象，如：str、list、tuple、dict 都是可迭代对象（这里就不局限于 list 了）

key 用列表元素的某个属性或函数进行作为关键字（此函数只能有一个参数）

reverse 排序规则. reverse = True 降序或者 reverse = False 升序，默认升序

return 有返回值值，返回新的队列

```
a = [1, -6, 2, -5, 9, 4, 20, -3]
a.sort(key=lambda x:abs(x))
print(a)


print(sorted(a, key=lambda x:abs(x)))
```

3.8 按字符串长度排序

b = ["hello", "helloworld", "he", "hao", "good"]

按 list 里面单词长度倒叙

```
# 按 list 里面单词长度倒叙
a = ["hello", "helloworld", "he", "hao", "good"]


# 1 reverse=True 是倒叙
a.sort(key=lambda x: len(x), reverse=True)
print(a)


# 2 sorted()
print(sorted(a, key=lambda x: len(x), reverse=True))
```

3.9 去重与排序

L1 = [1, 2, 3, 11, 2, 5, 3, 2, 5, 33, 88]

如何用一行代码得出[1, 2, 3, 5, 11, 33, 88]

L2 = [1, 2, 3, 4, 5] ,L[10:]结果是多少（报错？还是 None，还是[]）

```
L1 = [1, 2, 3, 11, 2, 5, 3, 2, 5, 33, 88]
# set 去重，sorted 排序
print(sorted(set(L1)))
```

```
L2 = [1, 2, 3, 4, 5]
print(L2[10:])  #  切片为空[], 不会报错
```

3.10 去重保留顺序

将列表中的重复值取出(仅保留第一个)，要求保留原始列表顺序

如 a=[3, 2, 1, 4, 2, 6, 1]　　输出[3, 2, 1, 4, 6]

```
a = [3, 2, 1, 4, 2, 6, 1]


# 方法 1
b = []
for i in a:
    if i not in b:
        b.append(i)
print(b)


# 方法 2 一行代码
print(sorted(set(a), key=lambda x:a.index(x)))
```

3.11 列表合并

a = [1, 3, 5, 7]

b = ['a', 'b', 'c', 'd']

如何得到[1, 3, 5, 7, 'a', 'b', 'c', 'd']

```
a = [1, 3, 5, 7]
b = ['a', 'b', 'c', 'd']

# 1.相加
print(a+b)

# 2.在列表 a 基础上追加
a.extend(b)
print(a)
```

3.12 生成列表(列表推导式)

用一行代码生成一个包含 1-10 之间所有偶数的列表

```
a = [i for i in range(1, 11) if i % 2 == 0]
print(a)
```

3.13 列表成员的平方

列表 a = [1,2,3,4,5]，计算列表成员的平方数，得到[1,4,9,16,25]

```
a = [1, 2, 3, 4, 5]

b = [i**2 for i in a]

print(b)
```

3.14 找出列表大于 0 的数

使用列表推导式，将列表中 a = [1, 3, -3, 4, -2, 8, -7, 6]

找出大于 0 的数，重新生成一个新的列表

```
a = [1, 3, -3, 4, -2, 8, -7, 6]


b = [i for i in a if i > 0]

print(b)
```

3.15 统计列表有多少大于 0

统计在一个队列中的数字，有多少个正数，多少个负数，如[1, 3, 5, 7, 0, -1, -9, -4, -5, 8]

```
a = [1, 3, 5, 7, 0, -1, -9, -4, -5, 8]

print(len([i for i in a if i > 0]))

print(len([i for i in a if i < 0]))
```

3.16 列表排除筛选

a = ["张三","张四","张五","王二"] 如何删除姓张的

```
a = ["张三", "张四", "张五", "王二"]
print([i for i in a if not i.startswith("张")])
```

3.17 列表过滤(filter)

题 1：有个列表 a = [1, 3, 5, 7, 0, -1, -9, -4, -5, 8] 使用 filter 函数过滤出大于 0 的数

filter() 函数用于过滤序列，过滤掉不符合条件的元素，返回由符合条件元素组成的新列表。

filter() 方法的语法: filter(function, iterable)

参数

function -- 判断函数。

iterable -- 可迭代对象。

返回 filter object 迭代器对象

```
a = [1, 3, 5, 7, 0, -1, -9, -4, -5, 8]
def great_then_0(x):
    """判断大于 0"""
    return x > 0
print(filter(great_then_0, a))
print(list(filter(great_then_0, a)))
```

题 2：列表 b = ["张三", "张四", "张五", "王二"] 过滤掉姓张的姓名

```
b = ["张三", "张四", "张五", "王二"]
def remove_zhang(x):
    return not str(x).startswith("张")
print(list(filter(remove_zhang, b)))
# 也可以用 lambda
print(list(filter(lambda x: not str(x).startswith("张"),
b)))
```

3.18 过滤列表中不及格学生(filter)

过滤掉列表中不及格的学生

a = [

{"name": "张三", "score": 66},

{"name": "李四", "score": 88},

{"name": "王五", "score": 90},

{"name": "陈六", "score": 56},

]

filter() 函数用于过滤序列，过滤掉不符合条件的元素，返回由符合条件元素组成的新列表。

filter() 方法的语法: filter(function, iterable)

参数

function -- 判断函数。

iterable -- 可迭代对象。

返回 filter object 迭代器对象

```
# 过滤掉列表中不及格的学生
a = [
        {"name": "张三", "score": 66},
        {"name": "李四", "score": 88},
        {"name": "王五", "score": 90},
        {"name": "陈六", "score": 56},
     ]

# 判断函数
def score_60(x):
    """判断 score>=60"""
    return x.get("score") >= 60

# 返回
print(filter(score_60, a))  # filter object
print(list(filter(score_60, a)))
```

判断函数 可以用 lambda 函数代替

```
# 过滤掉列表中不及格的学生
a = [
        {"name": "张三", "score": 66},
        {"name": "李四", "score": 88},
        {"name": "王五", "score": 90},
        {"name": "陈六", "score": 56},
    ]

# 返回
print(list(filter(lambda x: x.get("score")>=60, a)))
```

3.19 找出列表中最大数出现的位置

有个列表 a = [1, 2, 3, 11, 2, 5, 88, 3, 2, 5, 33]

找出列表中最大的数，出现的位置，下标从 0 开始

```
a = [1, 2, 3, 11, 2, 5, 88, 3, 2, 5, 33]


# 先找到最大的数
print(max(a))
# 出现的位置
print(a.index(max(a)))
```

3.20 找出列表中出现次数最多的元素

a = [

'my', 'skills', 'are', 'poor', 'I', 'am', 'poor', 'I',

'need', 'skills', 'more', 'my', 'ability', 'are',

'so', 'poor'

]

找出列表中出现次数最多的元素

先看下普通解决思路，先分别统计每个成员出现的次数，然后生成键值对的字典

对字典的 value 排序，取出最大值

```
a = [
    'my', 'skills', 'are', 'poor', 'I', 'am', 'poor', 'I',
    'need', 'skills', 'more', 'my', 'ability', 'are',
    'so', 'poor'
]
dict1 = {}
for i in a:
    if i not in dict1.keys():
        dict1[i] = a.count(i)
print(dict1)  # 分别统计每个字符出现的次数
```

```
# 取出最大的值
print(list(dict1.items()))
# 排序后取出最大值
print(sorted(list(dict1.items()), key=lambda
x:x[1])[-1][0])
```

如果能数量掌握 max 函数的使用，可以一行代码解决

```
a = [
    'my', 'skills', 'are', 'poor', 'I', 'am', 'poor', 'I',
    'need', 'skills', 'more', 'my', 'ability', 'are',
    'so', 'poor'
]
print(max(a, key=lambda x:a.count(x)))
```

3.21 分别统计列表中每个成员出现的次数

a = [

'my', 'skills', 'are', 'poor', 'I', 'am', 'poor', 'I',

'need', 'skills', 'more', 'my', 'ability', 'are',

'so', 'poor'

]

用前面讲到的原始方法

```
a = [
    'my', 'skills', 'are', 'poor', 'I', 'am', 'poor', 'I',
    'need', 'skills', 'more', 'my', 'ability', 'are',
    'so', 'poor'
]
print(max(a, key=lambda x:a.count(x)))
dict1 = {}
for i in a:
    if i not in dict1.keys():
        dict1[i] = a.count(i)
print(dict1)  # 分别统计每个字符出现的次数
```

使用内置方法 Counter

```
from collections import Counter
a = [
    'my', 'skills', 'are', 'poor', 'I', 'am', 'poor', 'I',
    'need', 'skills', 'more', 'my', 'ability', 'are',
    'so', 'poor'
]
b = Counter(a)
print(dict(b))
```

3.22 列表查找元素位置

给定一个整数数组 A 及它的大小 n，同时给定要查找的元素 val，

请返回它在数组中的位置(从 0 开始)，若不存在该元素，返回-1。

若该元素出现多次请返回第一个找到的位置

如 A1=[1, "aa", 2, "bb", "val", 33]

或 A2 = [1, "aa", 2, "bb"]

列表查找元素出现的位置可以用 list.index()方法，元素不在列表中的时候会抛异常

（注意的是 list 查找没有字符串中的 str.find() 和 str.rfind()方法）

```
A1 = [1, "aa", 2, "bb", "val", 33]
A2 = [1, "aa", 2, "bb"]

# index 找不到会抛异常
if "val" not in A1:
    print(-1)
else:
    print(A1.index("val"))

# 三元表达式
a = A1.index("val") if "val" in A1 else -1
print(a)
```

```
b = A2.index("val") if "val" in A2 else -1
print(b)
```

3.23 列表查找两数之和

给定一个整数数组 nums 和一个目标值 target ，请你在该数组中找出和为目标值的那两个整数，并返回他

们的数组下标。

你可以假设每种输入只会对应一个答案。但是，数组中同一个元素不能使用两遍。

示例:

给定 nums=[2，7，11，15]，target=9

因为 nums[0] + nums[1] =2+7 = 9

所以返回[0， 1]

```
nums = [2, 7, 11, 15]
target = 9
for i in range(0, len(nums)):
    for j in range(i+1, len(nums)):
        if nums[i]+nums[j] == target:
            print([i, j])
```

3.24 二维数组取值(矩阵)

有 a = [["A", 1], ["B", 2]] ，如何取出 2

List 通过下标取值，嵌套一层继续下标取值

```
a = [["A", 1], ["B", 2]]
print(a[1][1])
```

3.25 二维数组拼接

a = [[1,2],[3,4],[5,6]] 如何一句代码得到 [1, 2, 3, 4, 5, 6]

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

3.26 列表转字符串

L = [1, 2, 3, 5, 6]，如何得出 '12356'?

需注意的是列表中元素都是数字类型，不能直接用 join(),必须是字符串才能 join()拼接

```
L = [1, 2, 3, 5, 6]
b = [str(i) for i in L]
print("".join(b))
```

3.27 两个列表如何得到字典

a = ["a", "b", "c"]

b = [1, 2, 3]

如何得到 {'a': 1, 'b': 2, 'c': 3}

普通解决思路

```
a = ["a", "b", "c"]
b = [1, 2, 3]


d = {}
for i, j in zip(a, b):
    d[i] = j
print(d)
```

zip() 函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的对象，这样做的好处是节约了不少的内存。

```
a = ["a", "b", "c"]
b = [1, 2, 3]
print(dict(zip(a, b)))
```

3.28 列表按 age 从小到大排序

如下列表

people = [

{"name":"yoyo", "age": 20},

{"name":"admin", "age": 28},

{"name":"zhangsan", "age": 25},

]

按年龄 age 从小到大排序

```
people = [
    {"name":"yoyo", "age": 20},
    {"name":"admin", "age": 28},
    {"name":"zhangsan", "age": 25},
]
# 按 age 从小到大排序
print(sorted(people, key=lambda x: x["age"]))
```

3.29 列表插入元素

现有 nums=[2, 5, 7] ，如何在该数据最后插入一个数字 9 ，如何在 2 后面插入数字 0

```
nums = [2, 5, 7]
# 最后面插入 9
nums.append(9)
print(nums)    # [2, 5, 7, 9]
```

```
# insert 插入
nums.insert(nums.index(2)+1, 0)
print(nums)
```

3.30 打乱列表顺序随机输出

有个列表 a = [1, 2, 3, 4, 5, 6, 7, 8, 9]

如何打乱列表 a 的顺序,每次得到一个无序列表

random.shuffle()是打乱原来列表的顺序，所以返回值是 None

```
import random


a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
random.shuffle(a)  # 打乱原来的列表，不会生成新的列表
print(a)
```

第 4 章 元祖字典集合

4.1 输出 1-100 除 3 余 1 的数，结果为 tuple

输出 1-100 除 3 余 1 的数，结果为 tuple

```
a = [i for i in range(1, 101) if i%3 == 1]
print(a)
```

```
# 转元祖
print(tuple(a))
```

需注意的是元祖没有推导式，圆括号的推导式是生成器

```
# 注意：元祖没有推导式，圆括号的是生成器
b = (i for i in range(1, 101) if i%3 == 1)
print(b)  # generator object
```

4.2 把 2 个元祖转字典

将('a', 'b', 'c', 'd', 'e') 和 (1,2, 3, 4, 5)两个 tuple 转成

（1， 2， 3， 4， 5）为 key, ('a', 'b', 'c', 'd', 'e') 为 value 的字典

```
a = ('a', 'b', 'c', 'd', 'e')
b = (1, 2, 3, 4, 5)


print(dict(zip(b, a)))
```

4.3 把字典的 value 值转成 str

0061CF NM,

```
"""
字典遍历
"""
```

```
a = {'aa': 11, 'bb': 222}


for key, value in a.items():
    a[key] = str(value)
print(a)
```

也可以用字典推导式

```
a = {'aa': 11, 'bb': 222}
d = {i: str(j) for i, j in a.items()}
print(d)
```

4.4 (1)和(1,)区别，[1]和[1,]

a = [1,2,3] 和 b = [(1),(2),(3)] 以及 c = [(1,),(2,),(3,)] 的区别?

当我们用中括号的时候，肯定是列表，所以[1]和[1,]没区别

```
a = [1]
b = [1, ]
print(a)    # [1]
print(b)    # [1]
```

圆括号不一样，圆括号也可以是运算符，如(1+1)*2

```
c = (1+1)*2
print(c)   # 4
```

```
d = (2+3)
print(d)   # 5
```

圆括号又可以表示元组，所以这里有个规定：

当圆括号里面只有一个成员的时候，那么圆括号没啥意义，是可以去掉的

如果元组只有一个成员，需在后面加一个逗号

```
f = (1)
g = (1, )
print(f)   # 1
print(g)   # (1,)
```

于是上面的题得到的结果就是

```
a = [1, 2, 3]
b = [(1), (2), (3)]
c = [(1,), (2,), (3,)]
print(a)   # [1, 2, 3]
print(b)   # [1, 2, 3]
print(c)   # [(1,), (2,), (3,)]
```

4.5 map 函数将[1，2，3，4]处理成[1,0,1,0]

有个列表 a = [1, 2, 3, 4] 计算列表中每个数除以 2 取出余数 得到 [1,0,1,0]

map() 会根据提供的函数对指定序列做映射。

第一个参数 function 以参数序列中的每一个元素调用 function 函数，返回包含每次 function 函数返回值的新列表。

map() 函数语法：map(function, iterable, ...)

- function -- 函数
- iterable -- 一个或多个序列

Python 3.x 返回迭代器

```
a = [1, 2, 3, 4]
# map 使用
def get_yushu(x):
    return x % 2
print(map(get_yushu, a))  # map object
print(list(map(get_yushu, a)))  # [1, 0, 1, 0]

# 使用匿名函数
print(list(map(lambda x: x%2, a)))
```

4.6 map 函数将列表[1,2,3,4,5]转变成[1,4,9,16,25]

请将列表 [1,2,3,4,5] 使用 python 方法转变成 [1,4,9,16,25]

map 函数的功能可以理解成，对可迭代对象中的成员分别做一个功能计算

得到一个新的可迭代对象

```python
a = [1, 2, 3, 4, 5]
# 计算平方的函数
def seq(x):
    return x**2
print(list(map(seq, a)))

# 匿名函数
print(list(map(lambda x: x**2, a)))
```

4.7 map 函数 a=[1,3,5],b=[2,4,6]相乘得到[2,12,30]

map 函数对列表 a=[1,3,5],b=[2,4,6]相乘得到[2,12,30]

map 函数是可以传多个可迭代对象的

```python
a = [1, 3, 5]
b = [2, 4, 6]

def chengji(x, y):
    return x*y
print(list(map(chengji, a, b)))
```

```
# 匿名函数
print(list(map(lambda x, y: x*y, a, b)))
```

4.8 reduce 函数计算 1-100 的和

在 Python3 中，reduce() 函数已经被从全局名字空间里移除了，它现在被放置在 functools 模块里，如果想要使用它，则需要通过引入 functools 模块来调用 reduce() 函数

使用语法：reduce(function, sequence, initial=None)

参数:

function 调用的 function 必须有 2 个参数

sequence 传一个序列

initial 是初始值，默认为 None

例如：

reduce（lambda x，y:x+y，[1，2，3，4，5]）计算((((1+2)+3)+4)+5)

```
# reduce 计算 1-100 的和
from functools import reduce



def add(x, y):
    return x + y
print(reduce(add, range(1, 101)))
```

```
# 也可以用匿名函数
print(reduce(lambda x,y: x+y, range(1, 101)))
```

4.9 reduce 函数计算 10!

使用语法：reduce(function, sequence, initial=None)

计算 10!

```
# reduce 计算 10!
from functools import reduce

print(reduce(lambda x, y: x*y, range(10, 0, -1)))
```

计算 1! +2! +3! +。。。+10!

```
# reduce 计算 1!+2!+...+10!
from functools import reduce

# # i 的阶乘
# i = 10
# a = reduce(lambda x, y: x*y, range(1, i+1))

b = reduce(lambda a,b: a+b, [reduce(lambda x, y: x*y,
```

```
range(1, i+1)) for i in range(1, 11)])
print(b)
```

4.10 两个字典合并 a={"A":1,"B":2},b={"C":3,"D":4}

两个字典合并 a={"A":1,"B":2},b={"C":3,"D":4}

```
a={"A": 1,"B": 2}
b={"C": 3,"D": 4}

a.update(b)  # b 合并到 a
print(a)
```

4.11 {'a':1,'b':2,'c':1} 得到 {1:['a','c'],2:['b']}

m1={'a':1,'b':2,'c':1} # 将同样的 value 的 key 集合在 list 里，输出{1:['a','c'],2:['b']}

```
m1 = {'a':1,'b':2,'c':1}

d = [(i[1],i[0]) for i in m1.items()]
print(d)

g = {}
```

```
for j in d:
    if j[0] not in g.keys():
        g[j[0]] = [j[1]]
    else:
        g[j[0]].append(j[1])
print(g)
```

4.12 字典按 key 排序 d={"name":"zs","age":18,"}

d={"name":"zs","age":18,"city":"深圳","tel":"1362626627"}

字典根据键从小到大排序

```
d={"name":"zs","age":18,"city":"深圳
","tel":"1362626627"}


# dict 转 list of tuple 按 key 排序
f = sorted(d.items(), key=lambda x:x[0])
print(f)  # [('age', 18), ('city', '深圳')..]


print(dict(f))
```

4.13 集合（交集、差集、并集）

a = [2, 3, 8, 4, 9, 5, 6]

b = [2, 5, 6, 10, 17, 11]

1.找出 a 和 b 中都包含了的元素

2.a 或 b 中包含的所有元素

3.a 中包含而集合 b 中不包含的元素

关于集合的三个概念：**并集、交集、差集**

1、并集

对于两个给定集合 A、B，由两个集合所有元素构成的集合，叫做 A 和 B 的并集。

记作：AUB　读作“A 并 B”

例：　{3,5}U{2,3,4,6}= {2,3,4,5,6}

2、交集

对于两个给定集合 A、B，由属于 A 又属于 B 的所有元素构成的集合，叫做 A 和 B 的交集。

记作：　A∩B　　读作“A 交 B”

例：　A={1,2,3,4,5}，B={3,4,5,6,8}，A∩B={3,4,5}

3、差集

记 A，B 是两个集合，则所有属于 A 且不属于 B 的元素构成的集合，叫做集合 A 减集合 B

记作： A-B

例： A={1,2,3,4,5}, B={3,4,5,6,8}， A-B={1, 2}

```
a = [2, 3, 8, 4, 9, 5, 6]
b = [2, 5, 6, 10, 17, 11]

# 集合 a 和 b 中都包含了的元素
print(set(a) & set(b))  # {2, 5, 6}

# a 或 b 中包含的所有元素
print(set(a) | set(b)) # {2, 3, 4, 5, 6, 8, 9, 10, 11,
17}

# a 中包含而集合 b 中不包含的元素
print(set(a) - set(b))   # {8, 9, 3, 4}

# 不同时包含于 a 和 b 的元素
print(set(a) ^ set(b))   # {3, 4, 8, 9, 10, 11, 17}
```

第 5 章 综合练习题（上机考试）

5.1 有 1、2、3、4 组成无重复数的三位数（排列组合）

有 1、2、3、4 数字能组成多少互不相同无重复数的三位数?

分别打印这些三位数的组合

这一题很多小伙伴能想到的最直接的方法是嵌套三个 for 循环，然后判断 3 个数字不相等，得到组合的情况

```
s = 0
for i in range(1, 5):
   for j in range(1, 5):
       for k in range(1, 5):
           if i != j and i != k and j != k:
               print(i, j, k)
               s += 1
print("总过能生成的排列组合: ", s)
```

这种方式简单粗暴，毫无美感可言，只能算勉强及格。如果数据量非常大的时候，是很消耗内存的，于是会想到生成器。

```
def permutation():
    """
```

```
    定义一个生成器,函数带 yield 就是生成器
    """
    for i in range(1, 5):
        for j in range(1, 5):
            for k in range(1, 5):
                if i != j and i != k and j != k:
                    yield "%s%s%s"%(i,j,k)

for i in permutation():  # 遍历
    print(i)
# 也可以通过 len 来求长度
print(len(list(permutation())))
```

itertools 是 python 内置的模块，使用简单且功能强大，里面有个 permutations 方法就是专门生成排列组合的生成器。

```
from itertools import combinations,permutations


# 分别打印每种排列情况
for i, j, k in permutations(['1', '2', '3', '4'], 3):
    print(i, j, k)
# 计算总数
print(len(list(permutations(['1', '2', '3', '4'], 3))))
```

5.2 冒泡排序

a = [11, 2, 33, 1, 5, 88, 3]

冒泡排序：

依次比较两个相邻的元素，如果顺序（如从小到大、首字母从 A 到 Z）

错误就把他们交换过来

解决思路：

n 个元素，依次比较相邻 2 个元素，第一轮保证最大的数在最后一个位置

需比较 n-1 次

第二轮（最后一个数不用管），保证第二大的数在倒数第二个位置，需比较 n-2 次

依此类推。。。直到最后前面 2 个数比较，需比较 1 次

```
a = [11, 2, 33, 1,  5, 88, 3]


for i in range(len(a)-1, 0, -1):
    print(i)  # 需比较的次数
    for j in range(i):
        if a[j] > a[j+1]:      # 比较相邻 2 个数，交换位置
            a[j], a[j+1] = a[j+1], a[j]
```

```
        # print(a)
print(a)
```

5.3 文本中每行中长度超过 3 的单词

在以下文本中找出 每行中长度超过 3 的单词:

Call me Ishmael. Some years ago - never mind how long precisely - having

little or no money in my purse, and nothing particular to interest me

on shore, I thought I would sail about a little and see the watery part

of the world. It is a way I have of driving off the spleen, and regulating

the circulation. - Moby Dick

python 的预期结果(**尽量不超过 3 行搞定**):

[['Call', 'Ishmael.', 'Some', 'years', 'never', 'mind', 'long', 'precisely', 'having'],

['little', 'money', 'purse,', 'nothing', 'particular', 'interest'],

['shore,', 'thought', 'would', 'sail', 'about', 'little', 'watery', 'part'],

['world.', 'have', 'driving', 'spleen,', 'regulating'],

['circulation.', 'Moby', 'Dick']]

```
# 在以下文本中找出 每行中长度超过 3 的单词
aa = '''Call me Ishmael. Some years ago - never mind how
long precisely - having
```

```
little or no money in my purse, and nothing particular
to interest me
on shore, I thought I would sail about a little and see
the watery part
of the world. It is a way I have of driving off the spleen,
and regulating
the circulation. - Moby Dick'''

# 按每行切割
print(aa.split("\n"))
# 每行再按空格区分单词
b = [i.split(" ") for i in aa.split("\n")]
print(b)

# 判断子列表里面单词大于 3
c = [[j for j in i.split(" ") if len(j) > 3] for i in
aa.split("\n")]
print(c)
```

5.4 列表数据写入 txt（open 读写）

有一个数据 list of dict 如下

a = [

{"yoyo1": "123456"},

{"yoyo2": "123456"},

{"yoyo3": "123456"},

]

写入到本地一个 txt 文件，内容格式如下：

yoyo1,123456

yoyo2,123456

yoyo3,123456

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。
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。一般用于非文本文件如图片等。
r+	打开一个文件用于读写。文件指针将会放在文件的开头。
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。一般用于非文本文件如图片等。
w	打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
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打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。一般用于非文本文件如图片等。
w+	打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
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打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。一般用于非文本文件如图片等。
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。
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。
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。

```
a = [
    {"yoyo1": "123456"},
```

```
    {"yoyo2": "123456"},
    {"yoyo3": "123456"},
]

with open("xx.text", "a", encoding="utf-8") as fp:
    for i in a:
        for key,value in i.items():
            fp.write("%s,%s\n"%(key, value))
```

5.5 判断邮箱程序（正则）

写一个小程序：控制台输入邮箱地址（格式为 username@companyname.com）， 程序识别用户名和公司名后，将用户名和公司名输出到控制台。

要求：

1. 校验输入内容是否符合规范（xx@yy.com），如是进入下一步，如否则抛出提 示"incorrect email format"。注意必须以.com 结尾

2. 可以循环“输入--输出判断结果”这整个过程

3. 按字母 Q（不区分大小写）退出循环，结束程序

```
import re
```

```
while True:
    text = input("Please input your Email address: ")
    if text.upper() == "Q":  # lower()小写 s.upper()大写
        quit()
    if
re.match(r'^[0-9a-zA-Z_]{0,19}@[0-9a-zA-Z]{1,13}\.com
$',text):
        # print('Email address is Right!')
        username = re.findall("^(.+?)@", text)
        print(username[0])
        companyname = re.findall("@(.+?).com", text)
        print(companyname[0])
    else:
        print('Please reset your right Email address!')
```

5.6 判断一个字符串的括号自否闭合（栈）

判断一个字符串的括号自否闭合（包括大小中括号）

左括号和右括号必须是一一对应

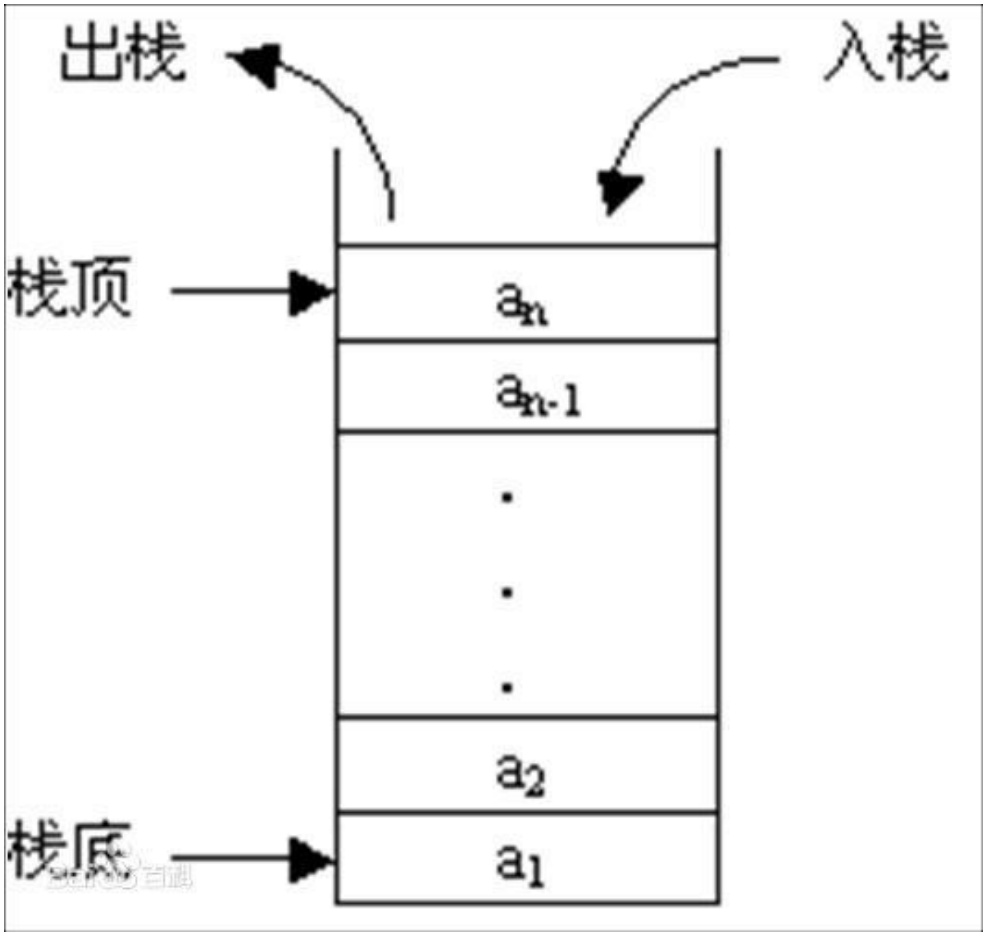
比如：{[{()}]()}　就是一个闭合的字符串

{[{()}]([)]} 这个里面（[)] 这种就是不闭合

栈：限定仅在表尾进行插入和删除操作的线性表。这一端被称为栈顶，相对地，把另一端称

为栈底。向一个栈插入新元素又称作进栈、入栈或压栈，它是把新元素放到栈顶元素的上面，

使之成为新的栈顶元素




```
def is_str_close(a):
    '''判断括号是否闭合 作者：上海-悠悠
    '''
    b = []
    flag = True
    for i in a:
        if i == "{" or i == "[" or i == "(": # 左边的括
号加进去
```

```
            b.append(i)
        elif i == "}":  # 遇到右边括号}弹出最后面的一个{
            if len(b) == 0 or b.pop() != "{":
                return False
        elif i == "]":  # 遇到右边括号]弹出最后面的一个[
            if len(b) == 0 or b.pop() != "[":
                return False
        elif i == ")":  # 遇到右边括号)弹出最后面的一个(
            if len(b) == 0 or b.pop() != "(":
                return False
    # 判断最后列表 b 里面的左边括号是否全部被弹出
    if len(b) != 0:
        flag = False
    return flag
if __name__ == '__main__':
    a = "{[{()}]()}"
    print(is_str_close(a))
    b = "({[{()}]()}"
    print(is_str_close(b))
    c = "{[{()}]()}]"
    print(is_str_close(c))
```

5.7 计算纯数字子串组成的单一数字（子串）

有一个纯数字组成的字符串，返回连续单一数字子串的个数

输入字符串：　“22252”

只含单一数字的子串是

1 个字符：2 出现 4 次，5 出现 1 次

2 个字符　22 出现 2 次

3 个字符　222 出现 1 次

4 个子串 0 次

5 个字符 0 次

总共 4+1+2+1 =8

输出结果：8

示例:

输入：22252

输出：　8

```
# a = "22252"
a = str(input("输入数字串"))
s = 0
for i in range(1, len(a)+1):
    b = []
```

```
    for j in range(len(a)-i+1):
        # 生成的组合
        new_s = a[j:j+i]
        # 判断是不是单一纯数字
        if i == new_s.count(new_s[0]):
            b.append(a[j:j+i])
    # print(b)
    s += len(b)
print("总共组合: ", s)
```

5.8 移除字符串里面的'ab'

有一个字符串列表['aababbc', 'badabcab'] 将字符串中的'ab' 移除

比如'aababbc' 移除里面的 ab 后得到 abc 需继续移除 ab，得到 c，直到字符串中不会出现连续的 ab

```
a = ['aababbc', 'badabcab']
b = []
for str_a in a:
    while 'ab' in str_a:
        str_a = str_a.replace('ab', '')
    b.append(str_a)
print(b)
```

5.9 看代码得结果（join 用法）

x="abc",y="def",z=["d","e","f"],

分别求出 x.join(y) 和 x.join(z)返回的结果

这一题主要考 join 的语法: str. join(self, iterable)

字符串调用 join 方法，参数传一个可迭代对象，y 是字符串，z 是一个列表，都是可迭代对象，所以上面的 x.join(y) 和 x.join(z) 返回结果是一样的。

先回顾下"hello_world_yoyo" 转成 ["hello", "world", "yoyo"]

然后["hello", "world", "yoyo"] 转成 "hello_world_yoyo"

```
a = "hello_world_yoyo"
print(a.split("_"))
# 运行结果 ['hello', 'world', 'yoyo']
b = ["hello", "world", "yoyo"]
print("_".join(b))
# 运行结果 hello_world_yoyo
```

上面的能懂的话，那么这题就是把下划线_字符换成了字符 abc

于是结果是:

dabceabcf

dabceabcf

```
x = "abc"
y = "def"
z = ["d", "e", "f"]
print(x.join(y))  # dabceabcf
print(x.join(z))  # dabceabcf
```

5.10 看代码得结果（类和继承）

阅读以下代码，打印结果是什么?

```
class A(object):

    def __init__(self):
        self.__Gender()
        self.Name()

    def __Gender(self):
        print("A.__Gender()")

    def Name(self):
        print("A.Name()")

class B(A):
```

```
    def __Gender(self):
        print("B.__Gender()")

    def Name(self):
        print("B.Name()")
b = B()
```

相关知识点：

1.B 类继承了 A 类

2.B()实例化的时候会自动运行__init__()方法

3.A 类里的__Gender() 是私有方法，所以只会执行 A 类里面的

4. Name 是实例方法，继承后 B 类的 Name 会覆盖掉 A 类的 Name 方法

运行结果：

A.__Gender()

B.Name()

5.11 看代码得结果（闭包）

阅读以下代码，得到的结果是什么

```
def fun():
    temp = [lambda x: i*x for i in range(4)]
    return temp

for everyLambda in fun():
    print(everyLambda(2))
```

关于闭包的相关讲解可以查看

https://www.cnblogs.com/yoyoketang/p/14479432.html

把匿名函数和列表推倒式转成普通代码，可以得到

```
def fun():
    temp = []
    for i in range(4):
        def inner(x):
            return i*x
        temp.append(inner)
    return temp

for everyLambda in fun():
    print(everyLambda(2))
```

这里只定义了一个函数 inner()，有 2 个变量，i 是函数外部的变量，x 是函数内部的变量。

外部变量 i 是可变的 0， 1， 2，3.但最后会被 3 覆盖前面的

内部变量 x 是不可变的，传值是 2

于是得到的结果是：

6

6

6

6

5.12 看代码得结果(列表推导式)

```
A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))


A1 = range(10)

A2 = [i for i in A1 if i in A0]

A3 = [A0[s] for s in A0]

A4 = [i for i in A1 if i in A3]

A5 = {i:i*i for i in A1}

A6 = [[i,i*i] for i in A1]
```

Python3 里面 range(10) 返回的不再是列表了，返回一个对象

range(0, 10)

[]

[1, 2, 3, 4, 5]

[1, 2, 3, 4, 5]

{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

[[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

5.13 看代码得结果(函数)

阅读以下代码，得到什么结果

```
def f(x,l=[]):
    for i in range(x):
        l.append(i*i)
    print l


f(2)

f(3,[3,2,1])

f(3)
```

运行的结果

[0, 1]

[3, 2, 1, 0, 1, 4]

[0, 1, 0, 1, 4]

5.14 看代码得结果(深拷贝和浅拷贝)

写出以下程序的输出结果

```
from copy import deepcopy, copy


li = [1, 2, [3, 4]]

li_sliced = li[:]

li_copied = copy(li)

li_deep_copied = deepcopy(li)


li[0] = 888

li[2][0] = 666

print(li_sliced, li_copied, li_deep_copied)
```

深拷贝和浅拷贝相关知识，

查看这篇 https://www.cnblogs.com/yoyoketang/p/14449962.html

深拷贝不管对原来列表如何改变，都不会变，所以 deepcopy 得到的结果是[1, 2, [3, 4]]

在 python 中有 6 个标准数据类型，他们分为可变和不可变两类。

- 不可变类型：Number（数字）String（字符串）Tuple（元组）
- 可变类型：List（列表）Dictionary（字典）Set（集合）

切片和 copy 都是浅拷贝，浅拷贝的时候数字是不可变类型，不会变

List 列表是可变类型，会随着原来列表的值改变

运行结果：[1, 2, [666, 4]] [1, 2, [666, 4]] [1, 2, [3, 4]]

5.15 map reduce filter 的使用

分别使用 map reduce filter 函数完成下面的任务

1.计算 1- 100 的和

2.1-10,对每个数字的平方

3. ["a", "ab", "abc", "bc", "cd"] 输出含有 c 字符的元素，返回值是 list

```
"""
1.计算 1- 100 的和
2.1-10,对每个数字的平方
3. ["a", "ab", "abc", "bc", "cd"] 输出含有 c 字符的元素，返回值是 list
"""
from functools import reduce
# 1.1- 100 的和
print(reduce(lambda x,y: x+y, range(1, 101)))


# 2.1-10 数字的平方
print(list(map(lambda x:x*x, range(1,11))))
```

```
# 输出含有 c 字符的元素
a = ["a", "ab", "abc", "bc", "cd"]
print(list(filter(lambda x:"c" in x, a)))
```

5.16 通过切片操作完成以下任务(切片)

有个字符串为"abcdefg.exe" 通过切片完成下面的任务

1.输出前 3 个字符

2.输出后 2 个字符

3.倒叙输出

4.间隔 1 个字符串输出

```
a = "abcdefg.exe"
print(a[:3])
print(a[-2:])
print(a[::-1])
print(a[::2])
```

5.17 根据列表数字出现次数排序去重(排序)

a=[1,2,1,2,2,2,3,4,5,6,56,7,1,3,4]

按列表中数字出现的次数，从高到低排序，并去除重复的

比如 2 出现了 4 次，排第一个位置。1 出现了 3 次，于是可以得到：[2, 1, 3, 4, 5, 6, 7, 56]

```
a = [1,2,1,2,2,2,3,4,5,6,56,7,1,3,4]


# 先 set 去重得到 {1, 2, 3, 4, 5, 6, 7, 56} 无序
print(set(a))
# 根据 set(a)里面成员在 a 列表中的次数排序
print(sorted(set(a), key=lambda x:a.count(x),
reverse=True))
```

也可以先统计每个元素出现的次数，再排序

```
# 先统计元素和对应的次数(元素，次数)
b = [(i, a.count(i)) for i in set(a)]
print(b) # [(1, 3), (2, 4), (3, 2), (4, 2), (5, 1), (6,
1), (7, 1), (56, 1)]


# 对 b 根据次数排序
b.sort(key=lambda x:x[1], reverse=True)
print(b)
# map 对列表成员计算得到 i[0]
print(list(map(lambda x:x[0], b)))
```

统计每个成员出现的次数可以用 Counter

```
a = [1,2,1,2,2,2,3,4,5,6,56,7,1,3,4]
```

```python
from collections import Counter
# 1.分别统计每个元素出现的次数
b = dict(Counter(a))
print(b)
# 2.按字典的 value 排序
c = sorted(b.items(), key=lambda x:x[1], reverse=True)
print(c)
# 3.map 对列表重新取值
print(list(map(lambda x:x[0], c)))
```

5.18 补缺失的代码-给个路径查找文件（递归）

def print_directory_contents(sPath):

"""

这个函数接受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。

"""

使用 os 模块查看文件路径，相关知识点

os.listdir(path) 返回 path 路径下所有的文件夹名称和文件名称，返回 list

类似 cmd 的 dir 命令

os.path.join(path, filename) 拼接路径和文件名称，返回一个新的路径

几个判断：

os.path.isfile(filepath) 判断是否是一个文件，返回 True 或 False

os.path.isdir(filepath) 判断是否是一个文件夹 ，返回 True 或 False

os.path.isabs(path) 判断是绝对路径 ，返回 True 或 False

os.path.exists 判断文件或文件夹是否存在

```
import os
sPath = r"D:\MyDjango"
# 类似于 cmd 目录 dir 查看当前路径下所有的文件和文件夹
print(os.listdir(sPath))

for child_file in os.listdir(sPath):
    # 拼接目录和文件
    file_path = os.path.join(sPath, child_file)
    # 判断子是不是一个文件夹
    if os.path.isdir(file_path):
        print("文件夹:", file_path)   # 文件夹
    else:
        print("文件:", file_path)
```

上面是针对一个给定的文件路径，可以获取当当前文件夹下的文件，判断如果是一个文件夹，

继续往里层获取文件，用到递归

```
def print_directory_contents(sPath):
    """
    这个函数接受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 sChild in os.listdir(sPath):
        sChildPath = os.path.join(sPath, sChild)
        if os.path.isdir(sChildPath):
            print_directory_contents(sChildPath)
        else:
            print(sChildPath)
```

5.19 如何判断一个字符串有没有重复字符

判断一个字符串是否包含重复字符。例如： “hello”就包含重复字符 ‘l’ ，而 “world” 就不包含重复字符，有重复打印 True，没重复打印 False

如果不考虑实现过程，可以直接判断字符串长度和去重后的长度是否一样

```
a = "hello"
print(False if len(a) == len(set(a)) else True)
```

如果想自己写实现过程，可以遍历字符串，挨个统计每个字符出现的次数，当判断到统计次数大于 1 就结束

```
def isdup(strs):
    '''判断字符串是否有重复字符'''
    for ch in strs:
        counts = strs.count(ch)
        if counts > 1:
            return True
    return False
if __name__ == '__main__':
    print(isdup("hello"))
    print(isdup("world"))
```

5.20 找出一个字符串中子串不含有重复字符的最长子串（子串）

给定一个字符串，请你找出其中不含有重复字符的最长子串的长度。

示例 1:

输入:" abcabcbb" 输出: 3

解释:因为无重复字符的最长子串是"abc"， 所以其长度为 3。

示例 2:

输入: "bbbbb"" 输出: 1

解释:因为无重复字符的最长子串是"b"， 所以其长度为 1。

示例 3:

输入: " pwwkew" 输出: 3

解释:因为无重复字符的最长子串是"wke"', 所以其长度为 3。

请注意，你的答案必须是子串的长度，"pwke"是一个子序列，不是子串。

```
a = "pwwkew"
b = []
for i in range(1, len(a)+1):
    for j in range(len(a)-i+1):
        # 生成的组合
        new_s = a[j:j+i]
        # 判断是不是重复字串
        if len(new_s) == len(set(new_s)):
            b.append(new_s)
print(b)
# b 中最长的字符
```

```
print(max(b, key=lambda x: len(x)))
print("最长字串长度:", len(max(b, key=lambda x: len(x))))
```

5.21 一个字符串中所有子串是回文的次数（子串）

回文是指正序（从左向右）和倒序（从右向左）读都是一样的。

例如：121 ，abcdedcba，123321 等都是回文

这种的字符串“ABCABADCSABBAUYIIYU”找出回文出现的次数

子串回文是：'BB', 'II','ABA','ABBA', 'YIIY','UYIIYU' 总共有 6 个

```
a = "ABCABADCSABBAUYIIYU"
s = 0
# 回文数大于 2 个字符
for i in range(2, len(a)+1):
    b = []
    for j in range(len(a)-i+1):
        # 生成的组合
        new_s = a[j:j+i]
        if new_s == new_s[::-1]:
            b.append(new_s)
    # print(b)
```

```
    s += len(b)
print("总共子串回文: ", s)
```

其它练习题

1 如何用 python 实现栈（Stack）的操作?

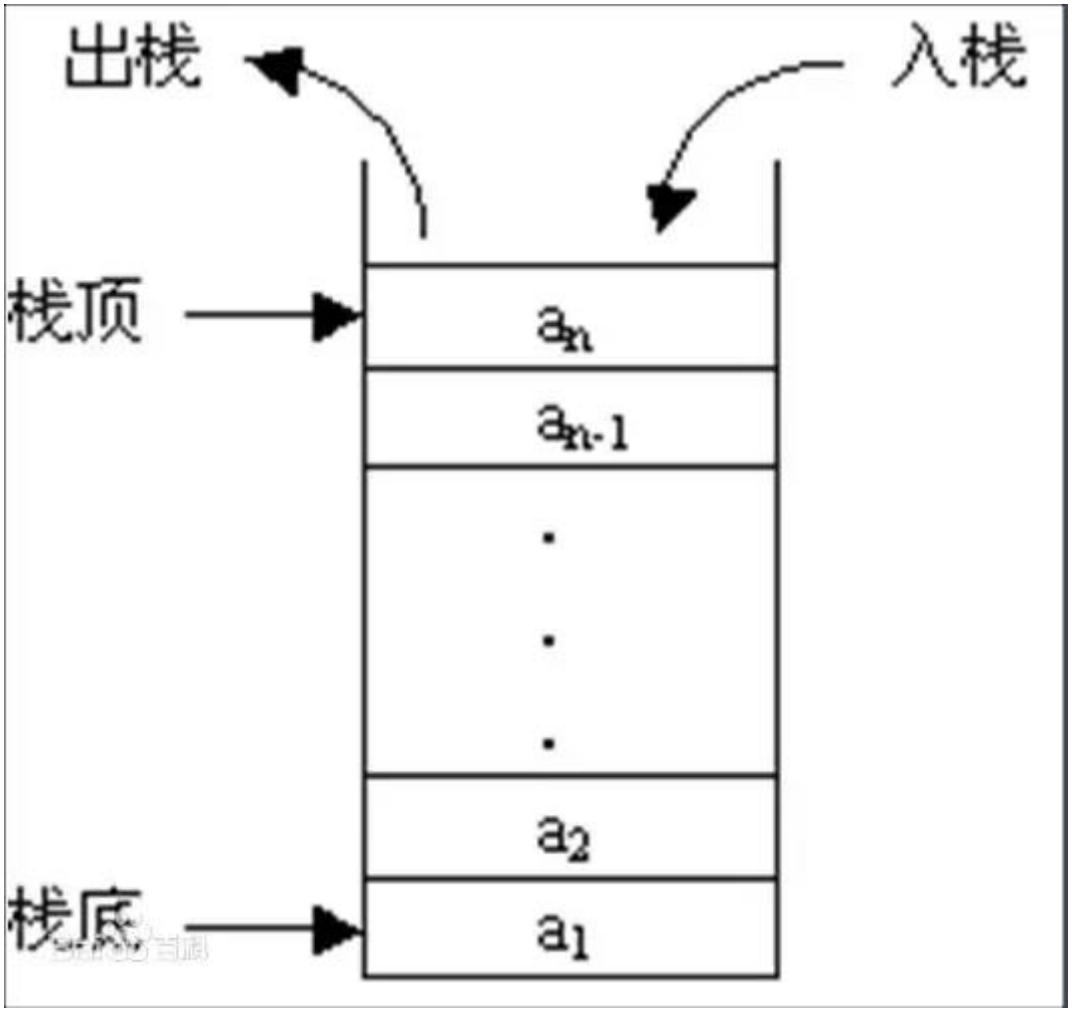
什么是栈（Stack）

栈是一个很基本的数据结构，也是非常重要的数据结构，栈的特点：“先进后出，后进先出”，

举个生活中最常见的例子

弹夹大家并不陌生，先按进去的最后才打出来，最后按进去的最先打出来，这就是栈的数据结构。

下图是详细的栈结构



所有的操作只在一端进行（如：弹夹），有 2 个最基本的操作，入栈（子弹按进去） 和 出栈 （子弹弹出来）。

栈有两端，最底端叫栈底，最上端叫栈顶。

弄清楚了栈的基本结构和操作后，就可以用 python 写一个栈（Stack）

定义抽象数据类型栈的各种操作：

- Stack（）:创建一个空栈，不包含任何数据项

- push（item）：将 item 加入栈顶，无返回值
- pop（）：将栈顶数据项移除，并返回所移除的值，栈被修改
- peek（）：“窥视”栈顶数据项，返回栈顶的数据项但不移除，栈不被修改
- isEmpty（）：返回栈是否为空栈
- size（）：返回栈中有多少个数据项

```
class Stack(object):
    def __init__(self):
        self.items = []

    def is_empty(self):
        """判断是否为空集"""
        return self.items == []

    def push(self, item):
        """添加新元素到栈顶"""
        self.items.append(item)

    def pop(self):
        """删除栈顶元素"""
        return self.items.pop()
```

```
    def peek(self):
        """窥探栈顶元素"""
        return self.items[-1]

    def size(self):
        """查看栈的大小"""
        return len(self.items)

if __name__ == '__main__':
    stack = Stack()
    print(stack.is_empty())
    stack.push(2)  # 入栈
    stack.push(3)  # 入栈
    print(stack.size())
    print(stack.peek())
    stack.pop()  # 出栈
    print(stack.size())
```

运行结果

```
True
2
3
```

1

2 找出列表中连续的数字，只取首尾（栈）

找出列表中连续的数字，然后只取首尾

有一个列表[1,2,3,4,8,6,7,11,15]

输出[(1,4),(6,8)]

```
"""
有一个列表[1,2,3,4,8,6,7,11,15]
输出[(1,4),(6,8)]
"""
a = [1,2,3,4,8,6,7,11,15]
b = []
s = []  # 空栈
for i in sorted(set(a)):
    if len(s) == 0 or s[-1] + 1 == i:
        s.append(i)  # 入栈
    else:
        if len(s) >= 2:
            b.append((s[0], s[-1]))
        s.clear()    # 清空
```

```
        s.append(i)  # 入栈
print(b)
```

另外一个解决思路

```
aa =[1,2,3,4,8,6,7,11,15, 99, 100]
l1 =[]
l2 = []
for x in sorted(set(aa)):
    l1.append(x)
    if x+1 not in aa:
        if len(l1) !=1:
            l2.append((l1[0], l1[-1]))
        l1 = []
print(l2)
```

3 找出两个字符串中最大公共子字符串（子串）

算法题(语言不限):

找出两个字符串中最大公共子字符串，如"abjeccarde"，"sjdgcargde"的最大子串为"car"

解决思路:

1.先遍历 a 的子字符串

2.判断 a 的子字符串同时也在字符串 b 里，添加到 f 列表

3.最后 f 列表里面取出最后一个，就是最长的子串了

```python
# 作者-上海悠悠 QQ交流群:717225969
# blog地址 https://www.cnblogs.com/yoyoketang/


a = "abjeccarde"
b = "sjdgcargde"
f = []
# i是字串长度，从1到len(a)
for i in range(1, len(a)+1):
    # j是下标位置
    for j in range(len(a)+1-i):
        e = a[j:j+i]
        # 判断字串同时也在b，添加过去
        if e in b:
            f.append(e)
print(f)
# -1是取出最长的
print(f[-1])
```

运行结果:

['a', 'j', 'e', 'c', 'c', 'a', 'r', 'd', 'e', 'ca', 'ar', 'de', 'car']

car

上面的解决思路，虽然没太大问题，得到的结果是一样的，但是这题考的是算法。

要找出最长的子串，可以先从长的子串遍历，判断符合条件就应该立即结束，没必要继续往下找了。找出公共的子串

```
a = "abjeccarde"
b = "absjdgcargde"
f = []
# 遍历查找字串长度从最长到 1
for i in range(len(a), 0, -1):
    for j in range(len(a)+1-i):
        e = a[j:j+i]
        if e in b:
            f.append(e)
            # 符合条件就结束跳出循环
            break
    if f:
        break  # 跳出外面的循环
print(f[0] if f else None)
```

运行结果：car

如果考虑到有多个值符合条件，可以去掉里层的 break

```
a = "abjeccardek"
b = "absjdgcargdek"
f = []
# 遍历查找字串长度从最长到 1
for i in range(len(a), 0, -1):
    for j in range(len(a)+1-i):
        e = a[j:j+i]
        if e in b:
            f.append(e)
            # # 符合条件就结束跳出循环，考虑到多个值，去掉里层 break
            # break
    if f:
        break  # 跳出外面的循环
print(f)
```

运行结果：['car', 'dek']

4. 请按长度为 8 拆分每个字符串后输出到新的字符串数组

题目

连续输入字符串，请按长度为 8 拆分每个字符串后输出到新的字符串数组；

长度不是 8 整数倍的字符串请在后面补数字 0，空字符串不处理。

输入描述：

连续输入字符串(输入 2 次,每个字符串长度小于 100)

举例：

输入： abc

123456789

输出：

abc00000

12345678

90000000

实现代码

这题首先考察字符串的个数，分为小于 8，等于 8，大于 8 的情况，其中大于 8 的字符按每 8 个字符切割，最后的余数不足 8 个继续补齐。

输入要求：输入 2 次,每个字符串长度小于 100。当大于 100 的时候，可以让用户重新输入，直到小于 100

```
b = ""
n = 2
while n:
    a = input("")
    if len(a) > 100:
        print("请输入字符小于100")
        continue
    else:
        if len(a) < 8:
            b += a+(8-len(a))*'0'+"\n"
        elif len(a) == 8:
            b += a+"\n"
        else:
            for i in range(len(a)//8):
                b += a[8*i:8*(i+1)]+"\n"
            # 判断是不是8的整数倍，如果不是，取出后面的
            if len(a)%8 != 0:
                yu = a[8*(len(a)//8):]
                b += yu+(8-len(yu))*'0'+"\n"
        n -= 1
print(b)
```

运行效果

abc

123456789

abc00000

12345678

90000000

5.实现删除字符串中出现次数最少的字符

题目

实现删除字符串中出现次数最少的字符，若多个字符出现次数一样，则都删除。

输出删除这些单词后的字符串，字符串中其它字符保持原来的顺序。

输入描述:

字符串只包含小写英文字母，不考虑非法输入，输入的字符串长度小于等于 20 个字节。

输出描述:

删除字符串中出现次数最少的字符后的字符串。

输入例子:

abcdd

输出例子:

dd

解决代码

普通点的解决思路，先分别统计每个字符出现的次数，再得到最小的次数，下一步遍历删除字符里面次数最少的，可以用 replace()方法替换。

```
a = "abcdd"


# 先分别统计每个元素出现的次数
d = {}
for i in a:
    if i not in d.keys():
        d[i] = 1
    else:
        d[i] += 1
print(d)


# 再次遍历去掉次数最少的
for j in d.keys():
    # 判断等于最小的次数  min(d.values())
```

```
    if d[j] == min(d.values()):
        a = a.replace(j, '')
print(a)
```

如果全部用内置函数，可以先用 min 函数得到最少次数的字符，根据此字符就能得到最小的次数，字符串过滤用 filter 函数实现

```
# 首先找到出现最少次数的字符
min_str = min(a, key=lambda x: a.count(x))
# 根据这个字符得到次数
n = a.count(min_str)
# 再根据 filter 函数筛选
print("".join(filter(lambda x:a.count(x)>n, a)))
```

多组输入

题目描述：

实现删除字符串中出现次数最少的字符，若多个字符出现次数一样，则都删除。输出删除这些单词后的字符串，字符串中其它字符保持原来的顺序。

注意每个输入文件有多组输入，即多个字符串用回车隔开

输入描述：

字符串只包含小写英文字母，不考虑非法输入，输入的字符串长度小于等于 20 个字节。

输出描述：

删除字符串中出现次数最少的字符后的字符串。

```
aa = []
n = 3
while n > 0:
    a = input("")
    aa.append(a)
    n -= 1
for j in aa:
    min_str = min(j, key=lambda x: j.count(x))
    # 根据这个字符得到次数
    n = j.count(min_str)
    # 再根据 filter 函数筛选
    print("".join(filter(lambda x:j.count(x)>n, j)))
```

运行结果

aabcd

abcabcf

fbasdfsa

aa

abcabc

fasfsa

6 用户输入序号，显示用户选中的商品

输出商品列表，用户输入序号，显示用户选中的商品

商品 li = ["手机", "电脑", '鼠标垫', '游艇']

要求：1：页面显示 序号 + 商品名称，如：

1 手机

2 电脑

2： 用户输入选择的商品序号，然后打印商品名称

3：如果用户输入的商品序号有误，则提示输入有误，并重新输入。

4：用户输入 Q 或者 q，退出程序。

```
while True:
    li = ["手机", "电脑", "鼠标垫", "游艇"]
    for i in li:
        print(li.index(i)+1, i)
    keys = input('请输入选择的商品序号/输入 Q 或者 q 退出程
序: ')
    if keys.isdigit():
```

```
        num = int(keys)
        if num > 0 and num <= len(li):
            print(li[num-1])
        else:
            print('请输入有效数字')
    elif keys.upper() == 'Q':
        break
    else:
        print('请输入数字')
```

递归

①[{"a1":[{"a2":[{"a3":[{"a4":[{"a5":[]}]}]}]}]}] #返回 a4 数据

②[{"a1":[{"a2":[{"a3":[]}]}]}] #返回 a2 数据

多层嵌套，需要判断最后一层列表是否为空

如果最后一层列表为空返回 上一层数据

递归咋写

```
def get_next_data(x):
    if len(x) == 0:
        return
    else:
```

```
        xx = list(x[0].values())[0]
        if len(list(xx[0].values())[0]) == 0:
            return xx
        else:
            return get_next_data(xx)


if __name__ == '__main__':
    aa = [{"a1":[{"a2":[{"a3":[{"a4":[{"a5":[]}]}]}]}]}]
    r1 = get_next_data(aa)
    print(r1)
    bb = [{"a1":[{"a2":[{"a3":[]}]}]}]
    r2 = get_next_data(bb)
    print(r2)
```

7.两个列表查找对应值

我有一个列表和一个字符串 list=[54, 48, 51, 55, 95, 113, 117, 97, 110, 102, 97, 110, 103, 116, 111, 1],str="qwenaskjdhfrewqfgdsbdadoeewrwijuyeruhnkjhskdfsdert"，需要通过列表里的值，在字符串中找到对应的字符，如 54-n 48-e...依次类推，直到列表每个值都找到对应的字母，我应该怎么用代码实现哦?

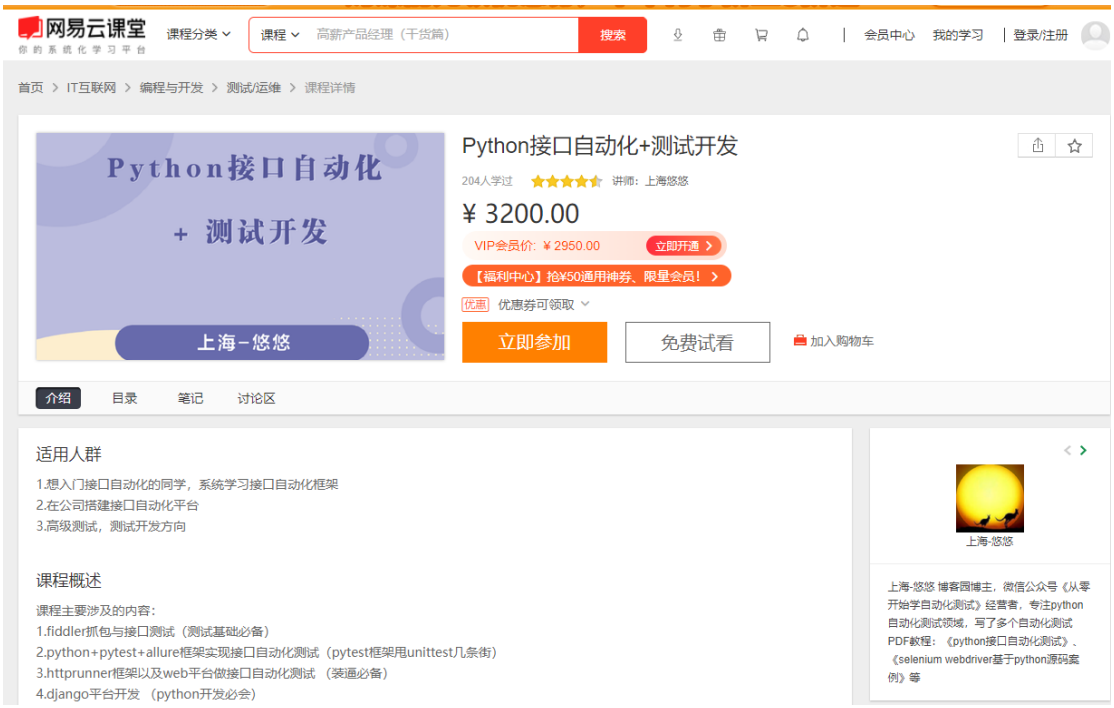
```
a=[54, 48, 51, 55, 95, 113, 117, 97, 110, 102, 97,
110, 103, 116, 111, 1]
b="qwenaskjdhfrewqfgdsbdadoeewrwijuyeruhnkjhskd
fsdert"

for i in a:
    if i <= len(b):
        print("{}-{}".format(i, b[i-1]))
    else:
        index = i % len(b)
        print("{}-{}".format(i, b[index-1]))
```

其它课程推荐

Python 接口自动化+测试开发

https://study.163.com/course/courseMain.htm?share=2&shareId=480000002230338&courseId=1210983809



适用人群

1.想入门接口自动化的同学，系统学习接口自动化框架

2.在公司搭建接口自动化平台

3.高级测试，测试开发方向

课程概述

课程主要涉及的内容：

1.fiddler 抓包与接口测试（测试基础必备）

2.python+pytest+allure 框架实现接口自动化测试（pytest 框架甩 unittest 几条街）

3.httprunner 框架以及 web 平台做接口自动化测试 （装逼必备）

4.django 平台开发 （python 开发必会）

5.pytest 框架结合 selenium 做 web 自动化测试（web 自动化项目实战）

6.docker 学习（加分项）

7.mockserver 环境搭建（加分项）

8.linux 搭建环境（加分项）

9.jmeter 压测（加分项）

10.jenkins+allure+邮件发送(加分项)

报名的同学，课后和工作中遇到的问题 可以协作远程解决

HttpRunner 3.x 接口自动化实战

https://study.163.com/course/courseMain.htm?courseId=1211857821&share=2&shareId=480000002230338

适用人群

1.本课程 2021 年 7 月录制

2.对 httprunner 3.x 框架，pytest 框架有兴趣

3.本课程不适合零基础的纯小白

课程概述

1.接口项目实例

课程接口项目是我自己部署的服务器，有详细的接口文档，每个接口都能任意访问

2.覆盖多种场景：

课程绝不是根据官方文档讲几个简单 demo, 会详细讲解每个关键字参数的使用，以及覆盖常见的场景实例，如：参数关联，文件上传，sign 签名，注册接口动态传参，连数据库操作，接口返回和数据库内容校验等。

提取和校验参数 jmespath 语法也会详细讲解。

3.课程会讲 2 种风格用例:

纯 Yaml 格式用例编写

以及 Pytest 风格用例编写

HttpRunner 2.x 接口自动化实战

https://study.163.com/course/courseMain.htm?courseId=1211561801&share=2&shareId=480000002230338

适用人群

1.本课程 2021 年 4 月录制

2.有一点 python 基础，和手工测过接口测试，知道 http 协议

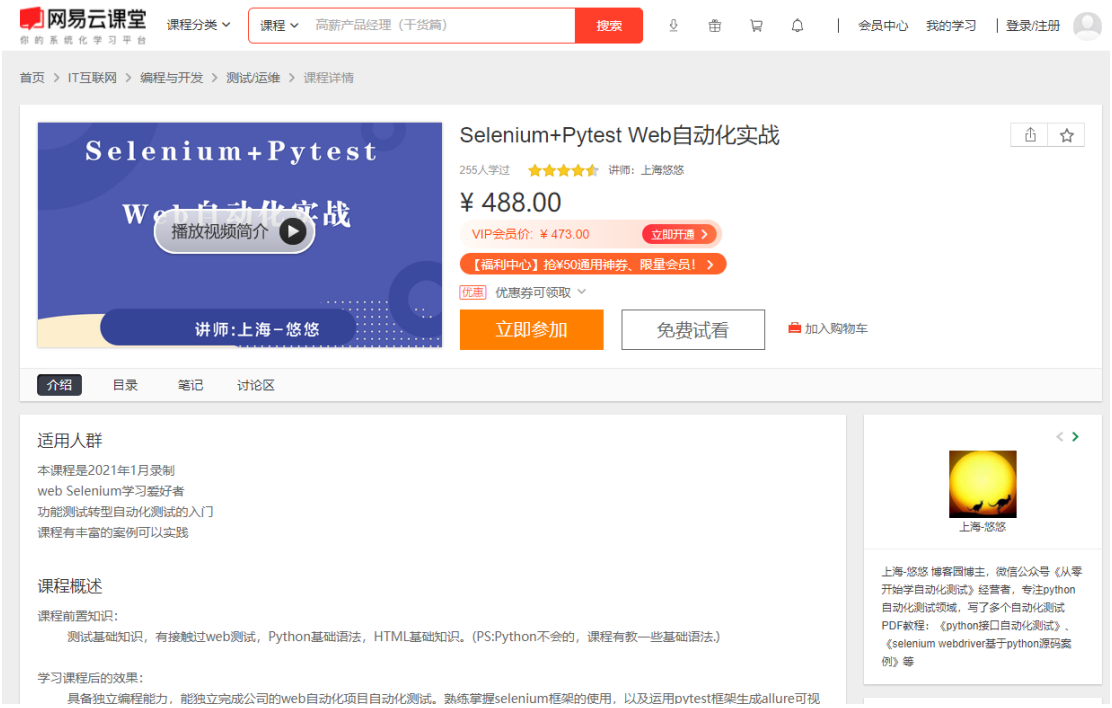
3.对 httprunner 框架有兴趣

课程概述

使用 HttpRunner 2.x 框架实现 http(s)协议接口自动化测试，使用 yaml 文件编写自动化用例，采用分层机制 API，TestCase, TestSuite 三层。完整项目接口案例，覆盖多种常见的场景

Selenium+Pytest Web 自动化实战

https://study.163.com/course/courseMain.htm?courseId=1210886815&share=2&shareId=480000002230338



适用人群

本课程是 2021 年 1 月录制

web Selenium 学习爱好者

功能测试转型自动化测试的入门

课程有丰富的案例可以实践

课程概述

课程前置知识：

测试基础知识，有接触过 web 测试，Python 基础语法，HTML 基础知识。(PS:Python 不会的，课程有教一些基础语法.)

学习课程后的效果：

具备独立编程能力，能独立完成公司的 web 自动化项目自动化测试。熟练掌握

selenium 框架的使用，以及运用 pytest 框架生成 allure 可视化报告。