**'''1、键盘录入一个数字，请用正则表达式 表现 该数值是否是1-100间的数字，是返回True 否则返回False  
'''""***# num=input('请输入一个数字：')  
# import re  
# f=re.match('^(100|[1-9]\d|[1-9])$',num)  
# if f:  
# print('True')  
# else:  
# print('False')***'''2、请用折半查找法，至少两种方法表现33和81，是否在 alist = [3,6,8,23,33,45,65,75,85,99]这个列表中，  
是返回True 否则返回False  
'''***# alist = [3,6,8,23,33,45,65,75,85,99]  
# def cha(a,item):  
# if len(a)==0:  
# return False  
# else:  
# x=len(a)//2  
# if a[x]==item:  
# return True  
# elif a[x]<item:  
# return cha(a[x+1:],item)  
# else:  
# return cha(a[:x],item)  
# print(cha(alist,33))  
# print(cha(alist,81))  
# def cha2(a,item):  
# first=0  
# last=len(a)-1  
# while first<=last:  
# x=(first+last)//2  
# if a[x]==item:  
# return True  
# elif a[x]<item:  
# first=x+1  
# else:  
# last=x-1  
# return False  
# print(cha2(alist,33))  
# print(cha2(alist,81))***'''3、请用3种方法打出斐波那契数列，个数不小于30 要求：迭代器、生成器、递归  
'''***# class Fib(object):  
# def \_\_init\_\_(self,n):  
# self.n=n  
# self.num1=0  
# self.num2=1  
# self.current=0  
# def \_\_iter\_\_(self):  
# return self  
# def \_\_next\_\_(self):  
# if self.current<self.n:  
# num=self.num1  
# self.num1,self.num2=self.num2,self.num1+self.num2  
# self.current+=1  
# return num  
# else:  
# raise StopIteration  
# f=Fib(30)  
# print(list(f))  
  
# def Fib(n):  
# num1,num2=0,1  
# for x in range(n):  
# num,num1,num2=num1,num2,num1+num2  
# yield num  
# f=Fib(30)  
# print(list(f))  
  
# def Fib(n):  
# if n==0:  
# return 0  
# elif n==1:  
# return 1  
# else:  
# return Fib(n-1)+Fib(n-2)  
#  
# for x in range(30):  
# print(Fib(x))***'''4、自定义一个返回函数的demo  
'''***# def f1():  
# def f2():  
# print(1)  
# return f2  
#  
# f1()()***'''5、请借用线程实现生产者消费者模式，要求执行100次数据还能保持每生产一个消费一个  
'''***# import threading,time,queue  
# def produce(q):  
# for x in range(200):  
# print('放入：',x)  
# q.put(x)  
# f2 = threading.Thread(target=consumer, args=(q,))  
# f2.start()  
# time.sleep(0.05)  
# def consumer(q):  
# print('取出：',q.get())  
#  
# if \_\_name\_\_ == '\_\_main\_\_':  
# q=queue.Queue()  
# f1=threading.Thread(target=produce,args=(q,))  
# f1.start()***'''6、请定义一个任务work1，使用进程池最大执行5个任务，至少批量执行3次，  
完成并行执行输出，注意需要设置主进程阻塞，等待子进程退出。  
'''***# import multiprocessing,time  
# def work1():  
# print('进程池')  
# time.sleep(1)  
# if \_\_name\_\_ == '\_\_main\_\_':  
# pool=multiprocessing.Pool(3)  
# for x in range(12):  
# pool.apply\_async(work1)  
# pool.close()  
# pool.join()***'''7、键盘录入一个邮箱，请用正则表达式匹配邮箱163、139、qq、sina 如：www.chenkai123@sina.com  
'''***# import re  
# f=re.match(r'^www\.\w{4,20}@(163|139|qq|sina)(\.com)$','www.chenkai123@sina.com')  
# if f:  
# print(f.group())  
# else:  
# print('匹配失败')***'''8、请用套接字实现tcp网络通讯 要求tcp分别实现服务端、客户端的通讯  
'''***# from socket import \*  
# tcp\_server\_socket=socket(AF\_INET,SOCK\_STREAM)  
# addr=('',8800)  
# tcp\_server\_socket.bind(addr)  
# tcp\_server\_socket.listen(100)  
# tcp\_client\_socket,ip=tcp\_server\_socket.accept()  
# tcp\_client\_socket.send('我是你爸爸'.encode('gbk'))  
# ret=tcp\_client\_socket.recv(1000)  
# print(ret.decode('gbk'))  
# tcp\_client\_socket.close()  
# tcp\_server\_socket.close()***'''9、使用线程模块的互斥锁解决多线程间恶意竞争资源的问题，模拟情景存在全局变量num = 0，  
分别创建执行函数work1和work2，并分别完成对全局变量num的修改，要求修改一次num值加1，  
循环修改1000000次，并分别输出最终的num的输出结果。  
'''***# import threading,time  
# def work1():  
# lock.acquire()  
# global num  
# for x in range(1000000):  
# num+=1  
# print(num)  
# lock.release()  
# def work2():  
# lock.acquire()  
# global num  
# for x in range(1000000):  
# num+=1  
# print(num)  
# lock.release()  
# num=0  
# if \_\_name\_\_ == '\_\_main\_\_':  
# f1=threading.Thread(target=work1)  
# f2=threading.Thread(target=work2)  
# lock=threading.Lock()  
# f1.start()  
# f2.start()***'''10、模拟场景当前有如下函数function(),一日老板突发奇想，想要在实现函数原有功能之前，实现附加验证功能A和B，  
要求分别实现并给函数添加具有模拟实现附加验证功能A和附加验证功能B的两个装饰器函数，并给function()添加并调用，  
显示输出效果。并用时间测验出执行时间  
def function():   
 print(“执行程序原有功能”)  
'''***# def w1(func):  
# def f():  
# print('验证A')  
# func()  
# return f  
# def w2(func):  
# def f():  
# print('验证B')  
# func()  
# return f  
#  
# @w1  
# @w2  
# def f():  
# print('被修饰函数')  
#  
# f()***'''11、请用进程实现进程间通信  
'''***# import multiprocessing,time  
# def write(q):  
# for x in range(10):  
# if q.full():  
# print('满啦')  
# break  
# else:  
# print('放入：',x)  
# q.put(x)  
#  
# def read(q):  
# while True:  
# if q.empty():  
# print('没啦')  
# break  
# else:  
# print("取出：",q.get())  
# if \_\_name\_\_ == '\_\_main\_\_':  
# q=multiprocessing.Queue(5)  
# f1=multiprocessing.Process(target=write,args=(q,))  
# f2=multiprocessing.Process(target=read,args=(q,))  
# f1.start()  
# f1.join()  
# f2.start()***'''12、请用sorted函数为列表排序 要求：表现分别实现升序和倒序  
'''***# list=[1,3,2,5,4]  
# print(sorted(list,key=lambda x:x,reverse=False))  
# print(sorted(list,key=lambda x:x,reverse=True))***'''13、请用递归函数实现表现阶乘的结果数列 如: 0 1 2 6 24 120 720 5040 40320 362880 .....  
'''***# def jiecheng(n):  
# if n==0:  
# return 0  
# elif n==1:  
# return 1  
# else:  
# return n\*jiecheng(n-1)  
# for x in range(10):  
# print(jiecheng(x))***'''14、自定义一个函数，实现一个闭包  
'''***# def f1():  
# a=10  
# def f2():  
# print(a)  
# return f2  
# f1()()***'''15、自定义一个迭代器，对列表[112,34,545,6578,6,645,43,3,4]进行遍历，重写iter方法和next方法  
'''***# class F(object):  
# def \_\_init\_\_(self,n):  
# self.n=n  
# self.curret=0  
# def \_\_iter\_\_(self):  
# return self  
# def \_\_next\_\_(self):  
# if self.curret<len(self.n):  
# n=self.n[self.curret]  
# self.curret+=1  
# return n  
# else:  
# raise StopIteration  
#  
#  
# if \_\_name\_\_ == '\_\_main\_\_':  
# list=[112,34,545,6578,6,645,43,3,4]  
# f=F(list)  
# for x in f:  
# print(x)***'''16、键盘录入一个网址，用正则表达式匹配网址，如：http://www.baidu.com  
'''***# x=input("请输入一个网址：")  
# import re  
# f=re.match(r'^http://www\.\w{4,25}(\.com)$',x)  
# if f:  
# print(f.group())  
# else:  
# print('匹配失败')***'''17、请用列表实现模拟一个队列，要求：添加3个元素，删除一个元素，展示队列变化  
Queue() 创建一个空的队列  
enqueue(item) 往队列中添加一个item元素  
dequeue() 从队列头部删除一个元素  
is\_empty() 判断一个队列是否为空  
size() 返回队列的大小  
'''***# class Queue(object):  
# def \_\_init\_\_(self):  
# self.queue=[]  
# def enqueue(self,item):  
# self.queue.append(item)  
# def dequeue(self):  
# return self.queue.pop(0)  
# def is\_empty(self):  
# if self.queue==[]:  
# return True  
# else:  
# return False  
# def size(self):  
# return len(self.queue)  
# q=Queue()  
# q.enqueue(1)  
# q.enqueue(2)  
# print(q.size())  
# print(q.dequeue())  
# print(q.size())***'''18、定义函数，使用isinstance方法判断列表、元组、数字、集合、字符串、字典中的可迭代对象  
要求（按完成的功能量给与相应条件的分数，未实现该条件功能效果不给分）：  
1) 正确引入isinstance需要导入的python库  
2) 正确用代码判断并且打印方法返回的对象  
3) 正确判断所有对象的返回值，并无遗漏  
'''***# from collections import Iterable  
# def pan(x):  
# return isinstance(x,Iterable)  
# print(pan([1]))  
# print(pan((1)))  
# print(pan(1))  
# print(pan({1,2}))  
# print(pan('12'))  
# print(pan({'name':19}))***'''19、使用filter过滤器和lambda表达式得到1-20之间能被2整除的数  
'''***# print(list(filter(lambda x:x%2==0,[x for x in range(1,21)])))***'''20、请用列表模拟一个栈，要求：添加至少3个元素，弹出一个元素，并分别查看栈顶，展示栈的变化  
Stack() 创建一个新的空栈  
push(item) 添加一个新的元素item到栈顶  
pop() 弹出栈顶元素  
peek() 返回栈顶元素  
is\_empty() 判断栈是否为空  
size() 返回栈的元素个数  
'''***# class Stack(object):  
# def \_\_init\_\_(self):  
# self.stack=[]  
# def push(self,item):  
# self.stack.append(item)  
# def pop(self):  
# return self.stack.pop()  
# def peek(self):  
# return self.stack[-1]  
# def is\_empty(self):  
# if self.stack==[]:  
# return True  
# else:  
# return False  
# def size(self):  
# return len(self.stack)  
  
# s=Stack()  
# s.push(1)  
# s.push(2)  
# print(s.pop())  
# print(s.peek())  
# print(s.size())***'''21、分别用冒泡排序选择排序对列表进行升序排序， 列表：alist = [ 4,3,6,2,7,9,1]  
'''***# alist = [ 4,3,6,2,7,9,1]  
# for x in range(len(alist)-1):  
# for i in range(len(alist)-1-x):  
# if alist[i]>alist[i+1]:  
# alist[i],alist[i+1]=alist[i+1],alist[i]  
# print(alist)***'''22、使用递归实现0--100和，偶数和，奇数和  
'''***# def sum1(n):  
# if n==1:  
# return 1  
# else:  
# return n+sum1(n-1)  
# def sum2(n):  
# if n==0:  
# return 0  
# else:  
# if n%2==0:  
# return sum2(n-2)+n  
# else:  
# return sum2(n-1)  
# def sum3(n):  
# if n==0:  
# return 0  
# elif n==1:  
# return 1  
# else:  
# if n%2==1:  
# return n+sum3(n-2)  
# else:  
# return sum3(n-1)  
# print(sum1(100),sum2(100),sum3(100))***'''23、请用线程实现 守护主线程  
'''***# import threading,time  
# def work():  
# print('子线程')  
# time.sleep(2)  
# print('231')  
# if \_\_name\_\_ == '\_\_main\_\_':  
# f=threading.Thread(target=work)  
# f.setDaemon(True)  
# f.start()  
# print('主进程')***'''24、自定义协程 要求使用yield 和next关键字实现  
'''***# def produce(c):  
# c.send(None)  
# for x in range(10):  
# print('生产：',x)  
# c.send(x)  
#  
# def consumer():  
# r=''  
# while True:  
# n = yield r  
# print('消费:', n)  
# if \_\_name\_\_ == '\_\_main\_\_':  
# produce(consumer())***'''25、用demo体现线程间全局变量共享  
'''***# import threading,time  
# num=0  
# def work1():  
# global num  
# for x in range(100):  
# num+=1  
# print(num)  
# def work2():  
# global num  
# for x in range(100):  
# num+=1  
# print(num)  
# if \_\_name\_\_ == '\_\_main\_\_':  
# f1=threading.Thread(target=work1)  
# f2=threading.Thread(target=work2)  
# f1.start()  
# f1.join()  
# f2.start()***'''26、键盘录入一个手机号，请用正则表达式匹配手机号，要求开头为1号段3-8 共11位 结尾不能以147结尾  
'''***# import re  
# num=input('请输入一个手机号：')  
# f=re.match(r'^1[3-8]\d{8}[^147]$',num)  
# if f:  
# print(f.group())  
# else:  
# print('匹配失败')***'''27、请使用reduce()函数，让一个参数x，对一个列表[1,2,3,4,5,6]的数值依次相乘，将结果返回  
'''  
from** functools **import** reduce  
*# print(reduce(lambda x,y:x\*y,[x for x in range(1,7)]))***'''28、使用map（）函数将第一个参数a依次作用在参数[1,2,3,4,5,6]的每一个元素上，  
返回包含每次函数返回值的新迭代器   
'''***# print(list(map(lambda x:x,[x for x in range(1,7)])))***'''29、暴力破解：  
条件：密码：  
1.a+b+c=1000  
2.a\*\*2+b\*\*2=c\*\*2  
要求：算出执行时间  
'''***# import timeit  
# def f():  
# for x in range(1000):  
# for y in range(1000):  
# if x \*\* 2 + y \*\* 2 == (1000 - x - y) \*\* 2:  
# print(x, y, 1000 - x - y)  
#  
# t=timeit.Timer('f()','from \_\_main\_\_ import f')  
# print(t.timeit(1))***'''30、请用套接字实现udp网络通讯 要求ucp分别实现服务端、客户端的通讯  
'''***# from socket import \*  
# udp=socket(AF\_INET,SOCK\_DGRAM)  
# addr=('192.168.1.6',8080)  
# udp.sendto('我是你爸爸'.encode('gbk'),addr)  
# ret=udp.recvfrom(1000)  
# print(ret[0].decode('gbk'),ret[1])  
# udp.close()  
  
# from socket import \*  
# udp=socket(AF\_INET,SOCK\_DGRAM)  
# addr=('',8080)  
# udp.bind(addr)  
# udp.sendto('我是你爸爸'.encode('gbk'),('192.168.1.6',8800))  
# ret=udp.recvfrom(10000)  
# print(ret[0].decode('gbk'))  
# udp.close()***'''31、定义一个类，实现线程的执行功能，定义三个任务，work1、work2、work3，  
打印出每个任务当前执行的是那条线程，正确继承模块，主动终止work2，并判断work2是否存活  
'''***# import threading,time  
# def work1():  
# print(1)  
# time.sleep(3)  
# def work2():  
# print(2)  
# time.sleep(2)  
# def work3():  
# print(3)  
# time.sleep(1)  
# if \_\_name\_\_ == '\_\_main\_\_':  
# f1=threading.Thread(target=work1)  
# f2=threading.Thread(target=work2)  
# f3=threading.Thread(target=work3)  
# f1.start()  
# print(threading.current\_thread())  
# f2.start()  
# print(threading.current\_thread())  
# f3.start()  
# print(threading.current\_thread())  
# print(threading.enumerate())***'''32、使用genvent实现协程  
'''***# import gevent  
# def work1():  
# print(1)  
# def work2():  
# print(2)  
# if \_\_name\_\_ == '\_\_main\_\_':  
# gevent.joinall(  
# [  
# gevent.spawn(work1),  
# gevent.spawn(work2)  
# ]  
# )***'''33、greenlet实现协程  
'''***# from greenlet import greenlet  
# def produce():  
# for x in range(10):  
# print('生产：',x)  
# f2.switch(x)  
# def consumer():  
# while True:  
# n=f1.switch()  
# print('消费：',n)  
# if \_\_name\_\_ == '\_\_main\_\_':  
# f1=greenlet(produce)  
# f2=greenlet(consumer)  
# f2.switch()***'''34、写出开头匹配字母和下划线，末尾是数字的正则表达式，键盘录入数据  
'''***# s=input('请输入：')  
# import re  
# f=re.match(r'^[a-zA-Z\_].\*\d$',s)  
# if f:  
# print(f.group())  
# else:  
# print('匹配失败')***'''35、用正则表达式匹配“adhdsa135555543dasdadasd”，拿出字符串中的数字  
'''**x=**'adhdsa135555543dasdadasd'**s=**''  
import** re  
**while True**:  
 q=0  
 **for** i **in** x:  
 **if** i.isdigit():  
 **break** q+=1  
 f = re.search(**'\d+'**,x)  
 **if** f:  
 s+=f.group()  
 x=x[(q+len(f.group())):]  
 **else**:  
 **break**print(s)