## Python Programming Puzzles

<https://www.w3resource.com/python-exercises/puzzles/index.php>

#Write a Python program find a list of integers with exactly

# two occurrences of nineteen and at least three occurrences

# of five

#Input: [19, 19, 15, 5, 3, 5, 5, 2]

# Output:True

#Input:[19, 15, 15, 5, 3, 3, 5, 2]

# Output:False

#Input:[19, 19, 5, 5, 5, 5, 5]

# Output:True

import numpy as np

import pandas as pd

ser1=list(np.random.randint(5,20,100))

while ser1.count(19)!=2 or ser1.count(5)<3:

ser1=list(np.random.randint(5,20,100))

print('i got the series as you requested:\n')

print(ser1)

print(ser1.count(19), ser1.count(5))

#Textbook

def test(nums):

return nums.count(19) == 2 and nums.count(5) >= 3

nums = [19,19,15,5,3,5,5,2]

print("Original list:")

print(nums)

print("Check two occurrences of nineteen and at least three occurrences of five in the said list:")

print(test(nums))

nums = [19,15,15,5,3,3,5,2]

print("\nOriginal list:")

print(nums)

print("Check two occurrences of nineteen and at least three occurrences of five in the said list:")

print(test(nums))

nums = [19,19,5,5,5,5,5]

print("\nOriginal list:")

print(nums)

print("Check two occurrences of nineteen and at least three occurrences of five in the said list:")

print(test(nums))

#Write a Python program that accept a list of integers and

# check the length and the fifth element. Return true if the

# length of the list is 8 and fifth element occurs thrice in

# the said list

import numpy as np

import pandas as pd

ser\_len=int(np.random.randint(2,10,1))

ser1=list(np.random.randint(1,20,ser\_len))

while ser\_len!=8 or ser1.count(ser1[4])!=3:

ser\_len=int(np.random.randint(2,10,1))

ser1=list(np.random.randint(1,20,ser\_len))

print(ser1)

#Textbook

def test(nums):

return len(nums) == 8 and nums.count(nums[4]) == 3

nums = [19,19,15,5,5,5,1,2]

print("Original list:")

print(nums)

print("Check whether the length of the said list is 8 and fifth element occurs thrice in the said list. :")

print(test(nums))

nums = [19,15,5,7,5,5,2]

print("\nOriginal list:")

print(nums)

print("Check whether the length of the said list is 8 and fifth element occurs thrice in the said list. :")

print(test(nums))

nums = [11,12,14,13,14,13,15,14]

print("\nOriginal list:")

print(nums)

print("Check whether the length of the said list is 8 and fifth element occurs thrice in the said list. :")

print(test(nums))

nums = [19,15,11,7,5,6,2]

print("\nOriginal list:")

print(nums)

print("Check whether the length of the said list is 8 and fifth element occurs thrice in the said list. :")

print(test(nums))

#Write a Python program that accept an integer test whether

# an integer greater than 4^4 and which is 4 mod 34.

# 4 mod 34: 除以34要餘4

# Input:922

# Output:True

# Input:914

# Output:False

# Input:854

# Output:True

import numpy as np

num1=int(np.random.randint(1,1000))

while num1<=256 or num1%34!=4:

num1=int(np.random.randint(1,1000))

print('the number as desired:',num1)

#We are making n stone piles! The first pile has n stones.

# If n is even, then all piles have an even number of stones.

# If n is odd, all piles have an odd number of stones.

#Each pile must more stones than the previous pile but as few as possible.

# Write a Python program to find the number of stones in each pile

#Input: 2

# Output:

# [2, 4]

# Input: 10

# Output:

# [10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

# Input: 3

# Output:

# [3, 5, 7]

# Input: 17

# Output:

# [17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49]

pile\_num=int(input('input the number of piles:\n'))

pile\_list=[]

for i in range(pile\_num):

pile\_list.append(pile\_num+2\*i)

print(pile\_list)

#Textbook

def test(n):

return [n + 2 \* i for i in range(n)]

n = 2

print("Number of piles:",n)

print("Number of stones in each pile:")

print(test(n))

n = 10

print("\nNumber of piles:",n)

print("Number of stones in each pile:")

print(test(n))

n = 3

print("\nNumber of piles:",n)

print("Number of stones in each pile:")

print(test(n))

n = 17

print("\nNumber of piles:",n)

print("Number of stones in each pile:")

print(test(n))

#Write a Python program to check the nth-1 string is a proper substring of nth string

# in a given list of strings.

# Input:

list1=['a', 'abb', 'sfs', 'oo', 'de', 'sfde']

# Output:

# True

# Input:

list2=['a', 'abb', 'sfs', 'oo', 'ee', 'sfde']

# Output:

# False

# Input:

list3=['a', 'abb', 'sad', 'ooaaesdfe', 'sfsdfde', 'sfsd', 'sfsdf', 'qwrew']

# Output:

# False

# Input:

list4=['a', 'abb', 'sad', 'ooaaesdfe', 'sfsdfde', 'sfsd', 'sfsdf', 'qwsfsdfrew']

# Output:

# True

def substring(lst):

if lst[len(lst)-2] in lst[len(lst)-1]:

print('True')

else: print('False')

substring(list1)

substring(list2)

substring(list3)

substring(list4)

#Write a Python program to test a list of one hundred integers between 0 and 999,

# which all differ by ten from one another.

# Return true or false

import random

list1=[]

list1.append(random.randint(0,1000))

i=0

while i<100:

num=random.randint(0,1000)

if abs(num-list1[i])==100:

list1.append(num)

i+=1

print(list1)

#Textbook

def test(li):

return all(i in range(1000) and abs(i - j) >= 10 for i in li for j in li if i != j) and len(set(li)) == 100

nums = list(range(0, 1000, 10))

print("Original list:")

print(nums)

print("Check whether the said list contains one hundred integers between 0 and 999 which all differ by ten from one another:")

print(test(nums))

nums = list(range(0, 1000, 20))

print("Original list:")

print(nums)

print("Check whether the said list contains one hundred integers between 0 and 999 which all differ by ten from one another:")

print(test(nums))

# Write a Python program to check a given list of integers where

# the sum of the first i integers is i

# Input:

list1=[0, 1, 2, 3, 4, 5]

# Output:

# False

# Input:

list2=[1, 1, 1, 1, 1, 1]

# Output:

# True

# Input:

list3=[2, 2, 2, 2, 2]

# Output:

# False

def first\_i\_sum(lst):

summ=0

i=1

try:

while i<len(lst)+1:

summ+=lst[i]

if summ!=i:

print('False')

break

i+=1

except: #原本的會有'IndexError'

print('True')

first\_i\_sum(list1)

first\_i\_sum(list2)

first\_i\_sum(list3)

#Textbook

def test(nums):

return all([sum(nums[:i]) == i for i in range(len(nums))])

nums = [0,1,2,3,4,5]

print("Original list:")

print(nums)

print("Check the said list, where the sum of the first i integers is i:")

print(test(nums))

nums = [1,1,1,1,1,1]

print("\nOriginal list:")

print(nums)

print("Check the said list, where the sum of the first i integers is i:")

print(test(nums))

nums = [2,2,2,2,2]

print("\nOriginal list:")

print(nums)

print("Check the said list, where the sum of the first i integers is i:")

print(test(nums))

#Write a Python program to split a string of words separated by commas

# and spaces into two lists, words and separators

input1='W3resource Python, Exercises.'

# Output:

# [['W3resource', 'Python', 'Exercises.'], [' ', ', ']]

# Input: The dance, held in the school gym, ended at midnight.

# Output:

# [['The', 'dance', 'held', 'in', 'the', 'school', 'gym', 'ended', 'at', 'midnight.'], [' ', ', ', ' ', ' ', ' ', ' ', ', ', ' ', ' ']]

# Input: The colors in my studyroom are blue, green, and yellow.

# Output:

# [['The', 'colors', 'in', 'my', 'studyroom', 'are', 'blue', 'green', 'and', 'yellow.'], [' ', ' ', ' ', ' ', ' ', ' ', ', ', ', ', ' ']]

print(input1.split()) #僅是隔空格 標點符號還連著

#Textbook

def test(string):

import re

merged = re.split(r"([ ,]+)", string)

return [merged[::2], merged[1::2]]

#re.split syntax: re.split(pattern,string,maxsplit)

# r"([ ,]+)" : 有包含空格 & , 但沒包含 .

s = "W3resource Python, Exercises."

print("Original string:",s)

print("Split the said string into 2 lists: words and separators:")

print(test(s))

s = "The dance, held in the school gym, ended at midnight."

print("\nOriginal string:",s)

print("Split the said string into 2 lists: words and separators:")

print(test(s))

s = "The colors in my studyroom are blue, green, and yellow."

print("\nOriginal string:",s)

print("Split the said string into 2 lists: words and separators:")

print(test(s))

#Write a Python program to find list integers containing exactly

# four distinct values, such that no integer repeats twice

# consecutively among the first twenty entries

#Input:

list1=[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]

# Output:

# True

# Input:

list2=[1, 2, 3, 3, 1, 2, 3, 3, 1, 2, 3, 3, 1, 2, 3, 3]

# Output:

# False

# Input:

list3=[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

# Output:

# False

#test if it's alright

set1=set({list2[0],list2[1],list2[2],list2[3]})

print(set1)

print(len(set1)==len(list1))

chklist1=[True,True,True]

chklist2=[True, False,True]

print(all(chklist1), all(chklist2), '\n')

#start

def no\_repeat(lst):

chklist=[]

for i in range(len(lst)-4):

set2=set({lst[i],lst[i+1],lst[i+2],lst[i+3]})

if len(set2)==4:

chklist.append(True)

else:

chklist.append(False)

print(chklist)

if all(chklist)==True:

print('no integer repeats twice consecutively')

else:

print('oh,there is integer repeats..')

no\_repeat(list1)

no\_repeat(list2)

no\_repeat(list3)

#Textbook

print('Textbook\n')

def test(nums):

return all([nums[i] != nums[i + 1] for i in range(len(nums)-1)]) and len(set(nums)) == 4

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

print("Original list:")

print(nums)

print("Check said list of integers containing exactly four distinct values, such that no integer repeats twice consecutively:")

print(test(nums))

nums = [1, 2, 3, 3, 1, 2, 3, 3, 1, 2, 3, 3, 1, 2, 3, 3]

print("\nOriginal list:")

print(nums)

print("Check said list of integers containing exactly four distinct values, such that no integer repeats twice consecutively:")

print(test(nums))

nums = [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

print("\nOriginal list:")

print(nums)

print("Check said list of integers containing exactly four distinct values, such that no integer repeats twice consecutively:")

print(test(nums))

print('\n')

#my note:

# Textbook怪怪的,只有check相鄰的數字是否相同,並非check if there's repeats in one series

# 只check 數字種類是否為4

nums=[1,2,3,4,5,1,2,3,4,5,1,2,3,4,5]

print(test(nums)) #False

#實際上我好像會錯題目的意思了, 題目就是要數字種類exactly=4

#Given a string consisting of whitespace and groups of

# matched parentheses, write a Python program to split it into groups

# of perfectly matched parentheses without any whitespace

# Input:

str1='( ()) ((()()())) (()) ()'

# Output:

# ['(())', '((()()()))', '(())', '()']

# Input:

str2='() (( ( )() ( )) ) ( ())'

# Output:

# ['()', '((()()()))', '(())']

def split\_pt(str0):

summing=0

cut\_pos=[0]

for i in range(len(str0)):

if str0[i]=='(':

summing+=1

elif str0[i]==')':

summing-=1

else: summing+=0

if summing==0:

cut\_pos.append(i)

print(cut\_pos)

final=[]

for j in range(len(cut\_pos)-1):

final.append(str0[cut\_pos[j]:cut\_pos[j+1]+1])

#delete unwanted

for k in range(len(final)):

if len(final[k])<2:

final=final.pop(index=k)

print(final)

#還是有多出來的 且空格沒縮排

split\_pt(str1)

split\_pt(str2)

#Textbook

print('\nTextbook')

def test(combined):

ls = []

s2 = ""

for s in combined.replace(' ', ''):

s2 += s

if s2.count("(") == s2.count(")"):

ls.append(s2)

s2 = ""

return ls

combined = '( ()) ((()()())) (()) ()'

print("Parentheses string:")

print(combined)

print("Separate parentheses groups of the said string:")

print(test(combined))

combined = '() (( ( )() ( )) ) ( ())'

print("\nParentheses string:")

print(combined)

print("Separate parentheses groups of the said string:")

print(test(combined))

#Write a Python program to find the indexes of numbers of a given list

# below a given threshold

# Original list:

# [0, 12, 45, 3, 4923, 322, 105, 29, 15, 39, 55]

# Threshold: 100

# Check the indexes of numbers of the said list below the given threshold:

# [0, 1, 2, 3, 7, 8, 9, 10]

# Original list:

# [0, 12, 4, 3, 49, 9, 1, 5, 3]

# Threshold: 10

# Check the indexes of numbers of the said list below the given threshold:

# [0, 2, 3, 5, 6, 7, 8]

import random

list1=[]

while len(list1)<21:

list1.append(random.randint(1,1000))

print(list1)

thres=100

for i in range(len(list1)):

if list1[i]<thres: print(i,sep=' ')

#Textbook

def test(nums, thresh):

return [i for i, n in enumerate(nums) if n < thresh]

nums=[0, 12, 45, 3, 4923, 322, 105, 29, 15, 39, 55]

thresh = 100

print("Original list:")

print(nums)

print("Threshold: ",thresh)

print("Check the indexes of numbers of the said list below the given threshold:")

print(test(nums, thresh))

nums=[0, 12, 4, 3, 49, 9, 1, 5, 3]

thresh = 10

print("\nOriginal list:")

print(nums)

print("Threshold: ",thresh)

print("Check the indexes of numbers of the said list below the given threshold:")

print(test(nums, thresh))

#練習enumerate

for i, n in enumerate(list1):

print(i,n)

#Write a Python program to check whether the given strings are palindromes or not.

# Return True, False

# Input:

# list1=['palindrome', 'madamimadam', 'xxx', 'foo', 'eyes']

# Output:

# [False, True, True, False, False]

#Textbook

def test(strs):

return [s == s[::-1] for s in strs]

strs = ['palindrome', 'madamimadam', '', 'foo', 'eyes']

print("Original strings:")

print(strs)

print("\nTest whether the given strings are palindromes or not:")

print(test(strs))

#practice reversed-string

str1='hello'

print(str1[::-1], type(str1[::-1]))

#str1[::-1]本身就是string!

#Write a Python program to find the strings in a given list,

# starting with a given prefix.

# Input:

# [( ca,('cat', 'car', 'fear', 'center'))]

# Output:

# ['cat', 'car']

# Input:

# [(do,('cat', 'dog', 'shatter', 'donut', 'at', 'todo'))]

# Output:

# ['dog', 'donut']

str1=str(input('input the query as the examples:\n'))

str2=str1.strip('[').strip(']').strip('(').strip(')')

#strip功用:only for leading and trailing position of symbol

print(str2)

lst1=list(str2.split(','))

lst1[1]=lst1[1].strip('(')

print(lst1)

new\_list=[]

for element in lst1[1:len(lst1)]:

if lst1[0][1:] in element:

# print(lst1[0][1:])

new\_list.append(element)

print(new\_list)

#Textbook

print('\ntextbook:\n')

def test(strs, prefix):

return [s for s in strs if s.startswith(prefix)]

strs = ['cat', 'car', 'fear', 'center']

prefix = "ca"

print("Original strings:")

print(strs)

print("Starting prefix:", prefix)

print("Strings in the said list starting with a given prefix:")

print(test(strs, prefix))

strs = ['cat', 'dog', 'shatter', 'donut', 'at', 'todo']

prefix = "do"

print("\nOriginal strings:")

print(strs)

print("Starting prefix:", prefix)

print("Strings in the said list starting with a given prefix:")

print(test(strs, prefix))

#Write a Python program to find the lengths of a given list of

# non-empty strings

# Input:

# ['cat', 'car', 'fear', 'center']

# Output:

# [3, 3, 4, 6]

# Input:

# ['cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']

# Output:

# [3, 3, 7, 5, 2, 4, 0]

def find\_len(lst1):

print(list(len(word) for word in lst1))

find\_len(['cat', 'car', 'fear', 'center'])

find\_len(['cat', 'dog', 'shatter', 'donut', 'at', 'todo', ''])

find\_len(['steelers','ravens','bengals','roethlisberger'])

#Textbook

def test(strs):

return [\*map(len, strs)]

# 如果沒有\*, 會output成 [<map object at (位址)>]

# Repetition Operator(\*):

# \*:creates several copies of that object and joins them together

# 格式 : [\*map(函數, iterables)]

# 把函數的作用 imprint(map)到一個iterable(list, tuple..)裡

strs = ['cat', 'car', 'fear', 'center']

print("Original strings:")

print(strs)

print("Lengths of the said list of non-empty strings:")

print(test(strs))

strs = ['cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']

print("\nOriginal strings:")

print(strs)

print("Lengths of the said list of non-empty strings:")

print(test(strs))

#warmup

list1=[1,2,3,4,5]

list2=['a','b','c','d','e']

def add\_10(num):

return num+10

print([\*map(add\_10,list1)]) #[11,12,13,14,15]

print([\*map(ord,list2)]) #[97,98,99,100,101]

#Write a Python program find the longest string of a given list of strings

# Input:

lst1= ['cat', 'car', 'fear', 'center']

# Output:

# center

# Input:

lst2= ['cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']

# Output:

# shatter

def choose\_longest(lst):

max\_len= max(list( len(lst[i]) for i in range(len(lst)) ))

longest\_list=[]

for term in lst:

if len(term)==max\_len:

longest\_list.append(term)

return longest\_list

print(choose\_longest(lst1))

print(choose\_longest(lst2))

lst3=['steelers','pirates','penguins','bengals','reds','ravens']

print(choose\_longest(lst3))

#Textbook

print('\nTextbook:\n')

def test(words):

return max(words, key=len)

strs = ['cat', 'car', 'fear', 'center']

print("Original strings:")

print(strs)

print("Longest string of the said list of strings:")

print(test(strs))

strs = ['cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']

print("\nOriginal strings:")

print(strs)

print("Longest string of the said list of strings:")

print(test(strs))

strs=['steelers','pirates','penguins','bengals','reds','ravens']

print("Longest string of the said list of strings:")

print(test(strs))

#this is not right,because only return one string

#Textbook-2

print('\nTextbook-2\n')

def test(words):

alphabet = "abcdefghijklmnopqrstuvwxyz"

alphabet = alphabet + alphabet.upper()

alphabet\_dict = {}

for k in alphabet:

alphabet\_dict[k] = True

alphabet\_set = set(alphabet)

max\_word = words[0]

for el in words:

If not ((set(el) <= alphabet\_set) and (set(el) == set(el).intersection(alphabet\_dict.keys()))):

continue

#應該是check如果list裡有空字串,就break?

if len(el) >= len(max\_word):

max\_word = el

return max\_word

strs = ['cat', 'car', 'fear', 'center']

print("Original strings:")

print(strs)

print("Longest string of the said list of strings:")

print(test(strs))

strs = ['cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']

print("\nOriginal strings:")

print(strs)

print("Longest string of the said list of strings:")

print(test(strs))

#Write a Python program find the strings in a given list containing a

# given substring

# Input:

# [(ca,('cat', 'car', 'fear', 'center'))]

# Output:

# ['cat', 'car']

# Input:

# [(o,('cat', 'dog', 'shatter', 'donut', 'at', 'todo', ''))]

# Output:

# ['dog', 'donut', 'todo']

# Input:

# [(oe,('cat', 'dog', 'shatter', 'donut', 'at', 'todo', ''))]

# Output:

# []

#Textbook

def test(strs, substr):

return [s for s in strs if substr in s]

strs = ['cat', 'car', 'fear', 'center']

print("Original strings:")

print(strs)

substrs = 'ca'

print("Substring: "+substrs)

print("Strings in the said list containing a given substring:")

print(test(strs, substrs))

strs = ['cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']

print("\nOriginal strings:")

print(strs)

substrs = 'o'

print("Substring: "+substrs)

print("Strings in the said list containing a given substring:")

print(test(strs, substrs))

strs = ['cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']

print("\nOriginal strings:")

print(strs)

substrs = 'oe'

print("Substring: "+substrs)

print("Strings in the said list containing a given substring:")

print(test(strs, substrs))

#這個方法還是需要個別手動輸入 substring & string list!

#但找不到更好的辦法來split raw data like;

#[(ca,('cat', 'car', 'fear', 'center'))] 會說 ca not defined

#Write a Python program to create string consisting of the non-negative

# integers up to n inclusive

# Input:

# 4

# Output:

# 0 1 2 3 4

# Input:

# 15

# Output:

# 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

import math

def n\_inclusive(num):

num=math.floor(num) #if num is not an integer

lst1=list(i for i in range(num+1))

print(' '.join(str(lst1[i]) for i in range(num+1)))

n\_inclusive(3)

n\_inclusive(15)

n\_inclusive(6.7)

n\_inclusive (-8) # output: nil

#Textbook

def test(n):

return ' '.join(map(str,range(n+1)))

#range本身就是一個iterable

n = 4

print("Non-negative integer:")

print(n)

print("Non-negative integers up to n inclusive:")

print(test(n))

n = 15

print("\nNon-negative integer:")

print(n)

print("Non-negative integers up to n inclusive:")

print(test(n))

#An irregular/uneven matrix, or ragged matrix, is a matrix that has a

# different number of elements in each row. Ragged matrices are not

# used in linear algebra, since standard matrix transformations cannot

# be performed on them, but they are useful as arrays in computing.

#Write a Python program to find the indices of all occurrences of

# target in the uneven matrix.

# Input:

# [([1, 3, 2, 32, 19], [19, 2, 48, 19], [], [9, 35, 4], [3, 19]),19]

# Output:

# [[0, 4], [1, 0], [1, 3], [4, 1]]

#(find 19, 最後一個值)

# Input:

# [([1, 2, 3, 2], [], [7, 9, 2, 1, 4]),2]

# Output:

# [[0, 1], [0, 3], [2, 2]]

#(find 2, 最後一個值)

import numpy as np

arr1=np.array([([1, 3, 2, 32, 19], [19, 2, 48, 19], [], [9, 35, 4], [3, 19]),19])

print(type(arr1)) # <class 'numpy.ndarray'>

print(arr1.shape) # (2,) : 第一層有2個, 第二層不規則了

#Textbook-1

def test(M, T):

return [[i, j] for i, row in enumerate(M) for j, n in enumerate(row) if n == T]

M = [[1, 3, 2, 32, 19], [19, 2, 48, 19], [], [9, 35, 4], [3, 19]]

T = 19

print("Matrix:")

print(M)

print("Target value:")

print(T)

print("Indices of all occurrences of the target value in the said uneven matrix:")

print(test(M,T))

M = [[1, 2, 3, 2], [], [7, 9, 2, 1, 4]]

T = 2

print("\nMatrix:")

print(M)

print("Target value:")

print(T)

print("Indices of all occurrences of the target value in the said uneven matrix:")

print(test(M,T))

#Textbook-2

def test(M, T):

indices = []

for i, num in enumerate(M[0]):

if num == T:

indices.append([0, i])

# Search for element starting from the first column and storing the indices of the found elements in a list

for row, row\_num in zip(M[1:], range(1, len(M))):

for i, num in enumerate(row):

if num == T:

indices.append([row\_num, i])

return indices

M = [[1, 3, 2, 32, 19], [19, 2, 48, 19], [], [9, 35, 4], [3, 19]]

T = 19

print("Matrix:")

print(M)

print("Target value:")

print(T)

print("Indices of all occurrences of the target value in the said uneven matrix:")

print(test(M,T))

M = [[1, 2, 3, 2], [], [7, 9, 2, 1, 4]]

T = 2

print("\nMatrix:")

print(M)

print("Target value:")

print(T)

print("Indices of all occurrences of the target value in the said uneven matrix:")

print(test(M,T))

#Write a Python program to split a given string (s) into strings

# if there is a space in the string,

# otherwise split on commas if there is a comma,

# otherwise return the list of lowercase letters with odd order

# (order of a = 0, b = 1, etc.)

# Input:

# a b c d

# Split the said string into strings if there is a space in the string,

# otherwise split on commas if there is a comma,

# Output:

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

# Input:

# a,b,c,d

# Split the said string into strings if there is a space in the string,

# otherwise split on commas if there is a comma,

# Output:

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

# Input:

# abcd

# Split the said string into strings if there is a space in the string,

# otherwise split on commas if there is a comma,

# Output:

# ['b', 'd']

def split(strs):

if ' ' in strs:

return list(strs.split(' '))

if ',' in strs:

return list(strs.split(','))

else:

return list(strs[i].lower() for i in range(len(strs)) if i%2!=0)

print(split('a b c d'))

print(split('a,b,c,d'))

print(split('abcd'))

print(split('Steelers wins today!'))

print(split('GetwithYourCOCKS'))

#Write a Python program to determine the direction

# ('increasing' or 'decreasing') of monotonic sequence numbers

# Input:

# [1, 2, 3, 4, 5, 6]

# Output:

# Increasing.

# Input:

# [6, 5, 4, 3, 2, 1]

# Output:

# Decreasing.

# Input:

# [19, 19, 5, 5, 5, 5, 5]

# Output:

# Not a monotonic sequence!

def direction(lst):

seq1=[(lst[i+1]-lst[i]) for i in range(len(lst)-1)]

check\_list=[]

for num in seq1:

if num>0:

check\_list.append(True)

elif num<0:

check\_list.append(False)

else:

check\_list.append('zero')

print(check\_list)

if all(check\_list):

print('Increasing')

elif not any(check\_list):

print('Decreasing')

else:

print('Not a monotonic sequence')

direction([1, 2, 3, 4, 5, 6])

direction([6, 5, 4, 3, 2, 1])

direction([19, 19, 5, 5, 5, 5, 5])

direction([7,34,66,127,-12,0,-7])

#warm-up

print(all([True,True,True])) # Increasing (if it's True)

print(all([True,False,False])) # Others

print(any([True,False,False])) # Others

print(any([False,False,False])) # Decreasing (if it's False)

#Textbook

print('Textbook:\n')

def test(nums):

return "Increasing." if all(nums[i] < nums[i + 1] for i in range(len(nums) - 1)) else \

"Decreasing." if all(nums[i + 1] < nums[i] for i in range(len(nums) - 1)) else \

"Not a monotonic sequence!"

nums = [1,2,3,4,5,6]

print("Original list:")

print(nums)

print("Check the direction ('increasing' or 'decreasing') of the said list:")

print(test(nums))

nums = [6,5,4,3,2,1]

print("\nOriginal list:")

print(nums)

print("Check the direction ('increasing' or 'decreasing') of the said list:")

print(test(nums))

nums = [19,19,5,5,5,5,5]

print("\nOriginal list:")

print(nums)

print("Check the direction ('increasing' or 'decreasing') of the said list:")

print(test(nums))

#Write a Python program to check, for each string in a given list,

# whether the last character is an isolated letter or not.

# Return True or False.

# Input:

# ['cat', 'car', 'fear', 'center']

# Output:

# [False, False, False, False]

# Input:

# ['ca t', 'car', 'fea r', 'cente r']

# Output:

# [True, False, True, True]

def iso\_check(lst):

res=[not lst[i]==lst[i].split(' ')[0] for i in range(len(lst))]

print (res)

iso\_check(['cat', 'car', 'fear', 'center'])

iso\_check(['ca t', 'car', 'fea r', 'cente r'])

iso\_check(['Fuck','fuck you','shit ']) #中間有space會一起count進去

#Textbook

def test(strs):

return [len(s.split(" ")[-1])==1 for s in strs]

strs = ['cat', 'car', 'fear', 'center']

print("Original strings:")

print(strs)

print("Check, for each string in the said list, whether the last character is an isolated letter:")

print(test(strs))

strs = ['ca t', 'car', 'fea r', 'cente r']

print("\nOriginal strings:")

print(strs)

print("Check, for each string in the said list, whether the last character is an isolated letter:")

print(test(strs))

#Write a Python program to compute the sum of the ASCII values of

# the upper-case characters in a given string

# Input:

# PytHon ExerciSEs

# Output:

# 373

# Input:

# JavaScript

# Output:

# 157

print(ord('A'),ord('Z')) #65~90

def sum\_ASCII(strs):

res=0

for char in strs:

if ord(char) in range(64,91):

res+=ord(char)

print(res)

sum\_ASCII('PytHon ExerciSEs')

sum\_ASCII('JavaScript')

sum\_ASCII('IGotYouBabe')

#warmup

import pandas as pd

print(sum(range(11)))

#Textbook-1

def test(strs):

return sum(map(ord,filter(str.isupper,strs)))

strs = "PytHon ExerciSEs"

print("Original strings:")

print(strs)

print("Sum of the ASCII values of the upper-case characters in the said string:")

print(test(strs))

strs = "JavaScript"

print("\nOriginal strings:")

print(strs)

print("Sum of the ASCII values of the upper-case characters in the said string:")

print(test(strs))

#Textbook-2

def test(strs):

tot = 0

for c in strs:

if c.isupper():

tot += ord(c)

return tot

#(略)

#Textbook-3

def test(strs):

return sum(ord(c) for c in strs if c.isupper())

#(略)

#Write a Python program to find the indices for which the numbers in

# the list drops

#NOTE: You can detect multiple drops just by checking

# if nums[i] < nums[i-1]

# Input:

# [0, -1, 3, 8, 5, 9, 8, 14, 2, 4, 3, -10, 10, 17, 41, 22, -4, -4, -15, 0]

# Output:

# [1, 4, 6, 8, 10, 11, 15, 16, 18]

# Input:

# [6, 5, 4, 3, 2, 1]

# Output:

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

# Input:

# [1, 19, 5, 15, 5, 25, 5]

# Output:

# [0, 2, 4, 6]

def drops(lst):

return list(i for i in range(1,len(lst)) if lst[i]<lst[i-1])

print(drops([0, -1, 3, 8, 5, 9, 8, 14, 2, 4, 3, -10, 10, 17, 41, 22, -4, -4, -15, 0]))

print(drops([6, 5, 4, 3, 2, 1]))

print(drops([1, 19, 5, 15, 5, 25, 5]))

#Write a Python program to create a list whose ith element is the

# maximum of the first i elements from a input list

# Input:

# [0, -1, 3, 8, 5, 9, 8, 14, 2, 4, 3, -10, 10, 17, 41, 22, -4, -4, -15, 0]

# Output:

# [0, 0, 3, 8, 8, 9, 9, 14, 14, 14, 14, 14, 14, 17, 41, 41, 41, 41, 41, 41]

# Input:

# [6, 5, 4, 3, 2, 1]

# Output:

# [6, 6, 6, 6, 6, 6]

# Input:

# [1, 19, 5, 15, 5, 25, 5]

# Output:

# [1, 19, 19, 19, 19, 25, 25]

def max\_till\_now(lst):

max\_list=lst

for i in range(1,len(lst)):

max\_list[i]=max(max\_list[i-1],lst[i])

print(max\_list)

max\_till\_now([0, -1, 3, 8, 5, 9, 8, 14, 2, 4, 3, -10, 10, 17, 41, 22, -4, -4, -15, 0])

max\_till\_now([6, 5, 4, 3, 2, 1])

max\_till\_now([1, 19, 5, 15, 5, 25, 5])

#Textbook

def test(nums):

return [max(nums[:i]) for i in range(1, len(nums) + 1)]

#(以下略)

#Warmup- bitwise operator (運算順序優先於logical operator)

age=18

is\_self\_excluded=True

print(age >= (18 & ~is\_self\_excluded))

# 18> 18 & -2 : True

print(~is\_self\_excluded)

# -2 (~是invert 再-1的意思, -True-1 => -2 )

# based on different language,

# some ~ may mean : -n+1 (-1+1 =>0 ; 0+1=> 1)

#當要表現負整數蒔,電腦用~:invert的方法表現出負數 此時會占用一個bit

# 7:0111 0:0000 -1(0-invert 再-1):1111 8:1000

#Left-shift

print(bin(39)) #100111

print(39 << 3) #100111000

print(0b100111000) #312, = 39\*2^3

print((39<<3) & 255) #56

#因為 mask只限定用8位,變成00111000

print(0b111000)

#Right-shift

print(5 >> 1) #2

print((5>>1) == 5//2) #floor-division

print(-34 >>8) #-1

print(bin(-2)) #這裡是 -0b10

#Write a Python program to find the XOR of two given strings

# interpreted as binary numbers

# Input:

# ['0001', '1011']

# Output:

# 0b1010

# Input:

# ['100011101100001', '100101100101110']

# Output:

# 0b110001001111

#XOR is equivalent to:

# (a^b)-ith element= (a-i + b-i) mod 2

#但我想用straight forward concept(exlclusive)來寫

def xor(lst):

first=lst[0]

second=lst[1]

res\_str=''

for i in range(len(first)):

if first[i]!=second[i]:

res\_str+='1'

else: res\_str+='0'

print('0b'+res\_str)

xor(['0001', '1011'])

xor(['100011101100001', '100101100101110'])

xor(['0101','1111'])

#Textbook-1

def test(nums):

return bin(int(nums[0],2) ^ int(nums[1],2))

#(略)

#syntax for 'int': int(x, base=10) ; x需要是string

print(int('1100011011',2)) #795

print(int('f',16)) #15

print(int('fuck',35)) #可以是任何的進位制

#Textbook-2

def test(nums):

a, b = nums

xor = int(a, 2) ^ int(b, 2)

return bin(xor)[2:].zfill(len(a))

#(略)

#str.zfill : zero filled left and fill to the required length

#Write a Python program to find the largest number where commas or

# periods are decimal points

# Input:

# ['100', '102,1', '101.1']

# Output:

# 102.1

def largest(lst):

lst2=lst

for i in range(len(lst)):

if ',' in lst[i]:

lst2[i]=lst[i].replace(',','.')

print(max(list(map(float,lst2))))

largest(['100', '102,1', '101.1'])

largest(['332','143','-31','566.34','96.41'])

#Textbook-1

def test(str\_nums):

return max(float(s.replace(",", ".")) for s in str\_nums)

#Textbook-2

def test(str\_nums):

numbers = []

for s in str\_nums:

numbers.append(float(s.replace(",", ".")))

numbers.sort()

return numbers[-1]

#Write a Python program to find x that minimizes mean squared deviation

# from a given a list of numbers

# Input:

# [4, -5, 17, -9, 14, 108, -9]

# Output:

# 17.142857142857142

# Input:

# [12, -2, 14, 3, -15, 10, -45, 3, 30]

# Output:

# 1.1111111111111112

import numpy as np

print(np.mean([4, -5, 17, -9, 14, 108, -9]))

#it's the answer, because it's just the average which minimize the deviation

print(np.std([4, -5, 17, -9, 14, 108, -9]))

print(np.mean([12, -2, 14, 3, -15, 10, -45, 3, 30]))

#it's the answer

#Textbook

def test(nums):

return sum(nums) / len(nums)

#Write a Python program to select a string from a given list of strings

# with the most unique characters

# Input:

# ['cat', 'catatatatctsa', 'abcdefhijklmnop', '124259239185125', '', 'foo', 'unique']

# Output:

# abcdefhijklmnop

# Input:

# ['Green', 'Red', 'Orange', 'Yellow', '', 'White']

# Output:

# Orange

def mu(lst):

screened=[]

for word in lst:

if len(set(word))==len(word) and len(word)>0:

screened.append(word)

max\_num= max(list(map(len,screened)))

# print(max\_num)

for result in screened:

if len(result)==max\_num:

print(result)

mu(['cat', 'catatatatctsa', 'abcdefhijklmnop', '124259239185125', '', 'foo', 'unique'])

mu(['Green', 'Red', 'Orange', 'Yellow', '', 'White'])

#Textbook-1

def test(strs):

return max(strs, key=lambda x: len(set(x)))

#Textbook-2

def test(strs):

largest\_set = {}

n = 0

largest = None

for phrase in strs:

diff = len(largest\_set) - len(set(phrase))

if diff < 0:

largest\_set = set(phrase)

largest = phrase

n = len(largest\_set)

elif diff == 0:

if n < len(set(phrase)):

largest\_set = set(phrase)

largest = phrase

n = len(largest\_set)

else:

pass

return largest

#Write a Python program to find the indices of two numbers that

# sum to 0 in a given list of numbers

# Input:

# [1, -4, 6, 7, 4]

# Output:

# [4, 1]

# Input:

# [1232, -20352, 12547, 12440, 741, 341, 525, 20352, 91, 20]

# Output:

# [1, 7]

def sumto0(lst):

res=[]

for i in range(len(lst)):

for j in range(len(lst)):

if lst[i]+lst[j]==0:

res.append(i)

res.append(j)

half\_num\_index=int(len(res)/2)

print(res[0:half\_num\_index])

sumto0([1, -4, 6, 7, 4])

sumto0([1232, -20352, 12547, 12440, 741, 341, 525, 20352, 91, 20])

#Textbook

def test(nums):

s = set(nums)

for i in s:

if -i in s:

return [nums.index(i), nums.index(-i)]

'''

30. Write a Python program to find the list of strings that has fewer

total characters (including repetitions).

Input:

[['this', 'list', 'is', 'narrow'], ['I', 'am', 'shorter but wider']]

Output:

['this', 'list', 'is', 'narrow']

Input:

[['Red', 'Black', 'Pink'], ['Green', 'Red', 'White']]

Output:

['Red', 'Black', 'Pink']

'''

def shorter(lst):

lst0=lst[0]

lst1=lst[1]

count0=sum([len(term) for term in lst[0]])

count1=sum([len(term) for term in lst[1]])

if count0>count1: print(lst[1])

elif count0<count1: print(lst[0])

else: print('it is equal')

shorter([['this', 'list', 'is', 'narrow'], ['I', 'am', 'shorter but wider']])

shorter([['Red', 'Black', 'Pink'], ['Green', 'Red', 'White']])

shorter([['Michael','Jackson','is','dead'],['But','Janet','is still alive']])

shorter([['Charlie'],['Charles']])

#Textbook

def test(strs):

return min(strs, key=lambda x: sum(len(i) for i in x))

#warmup

print(sum([1,2,3,4,5]))

print(sum((1,2,3,4,5)))

#both are all ok (15)

import numpy as np

print(sum(np.array([[1,2],[3,4],[5,6]]))) #[9,12]

'''31. Write a Python program to find the coordinates of a triangle

with the given side lengths.

Input:

[3, 4, 5]

Output:

[[0.0, 0.0], [3, 0.0], [3.0, 4.0]]

Input:

[5, 6, 7]

Output:

[[0.0, 0.0], [5, 0.0], [3.8, 5.878775382679628]]

'''

import math

def coord(len\_lst):

result=[]

#assume first pt is [0,0], second pt is [len\_lst[0],0]

#decide the third pt (x,y):

# sqrt(x\*\*2+y\*\*2)=len\_lst[2]; sqrt((x-len\_lst[0])\*\*2+y\*\*2)=len\_lst[1]

# => 2\*len\_lst[0]\*x+ len\_lst[0]\*\*2= len\_lst[2]\*\*2 -len\_lst[1]\*\*2

result.append([0.0,0.0])

result.append([float(len\_lst[0]),0.0])

x= (len\_lst[2]\*\*2 -len\_lst[1]\*\*2- len\_lst[0]\*\*2)/(2\*len\_lst[0])

y= math.sqrt(len\_lst[2]\*\*2-x\*\*2)

result.append([x,y])

print(result)

coord([3,4,5])

coord([5, 6, 7])

#純數學計算 有點問題

#Textbook (海龍公式)

#https://zh.m.wikipedia.org/zh-hant/%E6%B5%B7%E4%BC%A6%E5%85%AC%E5%BC%8F

# "利用勾股定理和代數式變形來證明"

def test(sides):

a, b, c = sorted(sides)

s = sum(sides) / 2 # semi-perimeter

area = (s \* (s - a) \* (s - b) \* (s - c)) \*\* 0.5 # Heron's formula

y = 2 \* area / a # height

x = (c \*\* 2 - y \*\* 2) \*\* 0.5

return [[0.0, 0.0], [a, 0.0], [x, y]]

sides = [3, 4, 5]

print("Sides of the triangle:",sides)

print("Coordinates of a triangle with the said side lengths:")

print(test(sides))

sides = [5, 6, 7]

print("\nSides of the triangle:",sides)

print("Coordinates of a triangle with the said side lengths:")

print(test(sides))

'''32. Write a Python program to rescale and shift numbers of

a given list, so that they cover the range [0, 1]

Input:

[18.5, 17.0, 18.0, 19.0, 18.0]

Output:

[0.75, 0.0, 0.5, 1.0, 0.5]

Input:

[13.0, 17.0, 17.0, 15.5, 2.94]

Output:

[0.7155049786628734, 1.0, 1.0, 0.8933143669985776, 0.0]

'''

def normalize(lst):

upper=max(lst)

lower=min(lst)

return list(map(lambda i: (i-lower)/(upper-lower), lst))

print(normalize([18.5, 17.0, 18.0, 19.0, 18.0]))

print(normalize([13.0, 17.0, 17.0, 15.5, 2.94]))

print(normalize([55.3,21.67,78.5,-9.3,124.6]))

# print(normalize([3,3,3,3,3])) # not considered min==max

#Textbook

def test(nums):

a = min(nums)

b = max(nums)

if b - a == 0:

return [0.0] + [1.0] \* (len(nums) - 1)

for i in range(len(nums)):

nums[i] = (nums[i] - a) / (b - a)

return nums

nums = [18.5, 17.0, 18.0, 19.0, 18.0]

print("Original list:")

print(nums)

print("Rescale and shift the numbers of the said list so that they cover the range [0, 1]:")

print(test(nums))

nums = [13.0, 17.0, 17.0, 15.5, 2.94]

print("\nOriginal list:")

print(nums)

print("Rescale and shift the numbers of the said list so that they cover the range [0, 1]:")

print(test(nums))

nums=[3,3,3,3,3]

print(test(nums))

# [0.0, 1.0, 1.0, 1.0, 1.0] # vague result!

'''

33. Write a Python program to find the positions of all uppercase vowels

# (not counting Y) in even indices of a given string

Input: w3rEsOUrcE

Output:

[6]

Input: AEIOUYW

Output:

[0, 2, 4]

'''

def AEIOU\_even\_index(strs):

result=[]

for i in range(len(strs)):

if strs[i] in 'AEIOU' and i%2==0:

result.append(i)

print(result)

AEIOU\_even\_index('w3rEsOUrcE')

AEIOU\_even\_index('AEIOUYW')

AEIOU\_even\_index('STeelersIs NumberOne andIHope theyCanGetSUPERBOWLS')

#Textbook

def test(strs):

return [i for i, c in enumerate(strs) if i % 2 == 0 and c in "AEIOU"]

'''

34. Write a Python program to find the sum of the numbers of a given list

among the first k with more than 2 digits.

Input: [4, 5, 17, 9, 14, 108, -9, 12, 76]

Value of K: 4

Output:

0

Input: [4, 5, 17, 9, 14, 108, -9, 12, 76]

Value of K: 6

Output:

108

Input: [114, 215, -117, 119, 14, 108, -9, 12, 76]

Value of K: 5

Output:

331

Input: [114, 215, -117, 119, 14, 108, -9, 12, 76]

Value of K: 1

Output:

114

'''

def sum\_K(lst, K):

if K>len(lst):

K=len(lst)

sums=0

for i in range(K):

if abs(lst[i])>=100:

sums+=int(lst[i])

print(sums)

sum\_K([4, 5, 17, 9, 14, 108, -9, 12, 76],4)

sum\_K([4, 5, 17, 9, 14, 108, -9, 12, 76],8)

sum\_K([4, 5, 17, 9, 14, 108, -9, 12, 76],12)

sum\_K([114, 215, -117, 119, 14, 108, -9, 12, 76],5)

#Textbook-1

def test(nums, k):

s = 0

for i in range(len(nums))[:k]:

if len(str(abs(nums[i])))>2:

s = s + nums[i]

return s

#Textbook-2

def test(nums, k):

return sum(n for n in nums[:k] if len(str(abs(n))) > 2)

'''

35. Write a Python program to compute the product of the odd digits

in a given number, or 0 if there aren't any

Input: 123456789

Output:

945

Input: 2468

Output:

0

Input: 13579

Output:

945

'''

import numpy as np

def odd\_digit\_prod(nums):

assert isinstance(nums,int), "it is not integer!"

prod\_lst=[]

result=1

for i in str(nums):

if i in '13579':

prod\_lst.append(int(i))

if len(prod\_lst)==0:

result=0

for j in prod\_lst:

result\*=int(j)

print(result)

odd\_digit\_prod(13579)

odd\_digit\_prod(9675227619705)

odd\_digit\_prod(2468024)

#Textbook

def test(n):

if any(int(c) % 2 for c in str(n)):

prod = 1

for c in str(n):

if int(c) % 2 == 1:

prod \*= int(c)

return prod

return 0

'''

36. Write a Python program to find the largest k numbers from a given

list of numbers

Input: [1, 2, 3, 4, 5, 5, 3, 6, 2]

Output:

[6]

Input: [1, 2, 3, 4, 5, 5, 3, 6, 2]

Output:

[6, 5]

Input: [1, 2, 3, 4, 5, 5, 3, 6, 2]

Output:

[6, 5, 5]

Input: [1, 2, 3, 4, 5, 5, 3, 6, 2]

Output:

[6, 5, 5, 4]

Input: [1, 2, 3, 4, 5, 5, 3, 6, 2]

Output:

[6, 5, 5, 4, 3]

'''

def k\_lar(lst,k):

result=[]

for i in range(k):

result.append(max(lst))

lst.remove(max(lst))

print(result)

k\_lar([1, 2, 3, 4, 5, 5, 3, 6, 2],1)

k\_lar([1, 2, 3, 4, 5, 5, 3, 6, 2],2)

k\_lar([1, 2, 3, 4, 5, 5, 3, 6, 2],3)

k\_lar([1, 2, 3, 4, 5, 5, 3, 6, 2],4)

k\_lar([1, 2, 3, 4, 5, 5, 3, 6, 2],5)

k\_lar([4.6,7,89,12.3,-23,5,55,78,112],4)

#Textbook

def test(nums, k):

if k == 0:

return []

return sorted(nums, reverse=True)[:k]

print(test([1, 2, 3, 4, 5, 5, 3, 6, 2],4))

'''

37. Write a Python program to find the largest integer divisor of

a number n that is less than n

Input: 18

Output:

9

Input: 100

Output:

50

Input: 102

Output:

51

Input: 500

Output:

250

Input: 1000

Output:

500

Input: 6500

Output:

3250

'''

def lar\_divisor(nums):

# nums / (smallest 因數), If 質數 then none

is\_prime=True

for i in range(2,nums):

if nums % i ==0:

is\_prime=False

print(int(nums/i))

break

if is\_prime:

print('it is a prime number')

lar\_divisor(80)

lar\_divisor(980)

lar\_divisor(9)

lar\_divisor(37)

lar\_divisor(581)

lar\_divisor(97)

#Textbook

def test(n):

return next(d for d in range(n - 1, 0, -1) if n % d == 0)

#有next才會秀值 要不然僅是個object

print(test(39))

print(test(37))

#warmup-1

print(map(lambda x: x\*\*2, [1,2,3,4,5]))

#<map object at 0x7ffff75c0e50>

print (next(map(lambda x: x\*\*2, [3,4,5,6,7])))

# 9

#warmup-2

lst=[1,2,3,4,5]

iterator=iter(lst)

print(next(iterator)) #1

print(next(iterator)) #2

print(next(iterator)) #3

print(next(iterator)) #4

print(next(iterator)) #5

# print(next(iterator)) #StopIteration

'''

38. Write a Python program to sort the numbers of a given list

by the sum of their digits

Input: [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

Output:

[10, 11, 20, 12, 13, 14, 15, 16, 17, 18, 19]

Input: [23, 2, 9, 34, 8, 9, 10, 74]

Output:

[10, 2, 23, 34, 8, 9, 9, 74]

'''

def sum\_digits(lst):

# 154 => 4+5+1 => (154%10) +(154//10->1st)%10 + (1st //10)

#for 'nums':sum of digits=

result\_lst=[]

for nums in lst:

quotient=nums//10

result=nums%10

while quotient>0:

nums=quotient

result+=nums%10

quotient=nums//10

result\_lst.append(result)

dict1=list(zip(lst,result\_lst))

print(dict1)

print(sorted (dict1, key=lambda x:x[1] ))

sort\_result=sorted (dict1, key=lambda x:x[1] )

print('the sorted result is:')

print([sort\_result[i][0] for i in range(len(sort\_result))])

print('\n')

sum\_digits([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20])

sum\_digits([23, 2, 9, 34, 8, 9, 10, 74])

#Textbook-1

def test(nums):

return sorted(nums, key=lambda n: sum(int(c) for c in str(n) if c != "-"))

#Textbook-2

def test(nums):

unordered = nums.copy()

ordered = []

while unordered:

# The number that comes first, since the list is sorted by the primes

smallest = unordered[0]

s = sum(int(c) for c in str(smallest) if c != "-")

for t in unordered:

t\_s = sum(int(c) for c in str(t) if c != "-")

if t\_s < s:

smallest = t

s = t\_s

ordered.append(smallest)

unordered.remove(smallest)

return ordered

'''

39. Write a Python program to determine which triples sum to zero

from a given list of lists

Input: [[1343532, -2920635, 332], [-27, 18, 9], [4, 0, -4], [2, 2, 2], [-20, 16, 4]]

Output:

[False, True, True, False, True]

Input: [[1, 2, -3], [-4, 0, 4], [0, 1, -5], [1, 1, 1], [-2, 4, -1]]

Output:

[True, True, False, False, False]

'''

def sum\_zero(lst):

return list(sum(lst[i])==0 for i in range(len(lst)))

print(sum\_zero([[1343532, -2920635, 332], [-27, 18, 9], [4, 0, -4], [2, 2, 2], [-20, 16, 4]]))

print(sum\_zero([[1, 2, -3], [-4, 0, 4], [0, 1, -5], [1, 1, 1], [-2, 4, -1]]))

#Textbook

def test(nums):

return [sum(t)==0 for t in nums]

'''

40. Write a Python program to find string s that, when case is flipped

gives target where vowels are replaced by chars two later.

Input: Python

Output:

pYTHQN

Input: aeiou

Output:

CGKQW

Input: Hello, world!

Output:

hGLLQ, WQRLD!

Input: AEIOU

Output:

cgkqw

'''

print(ord('a'),ord('z'),ord('A'),ord('Z'))

#lower->upper: -32 ; upper->lower: +32

def flipped(strs):

result=''

for char in strs:

if ord(char) in range(97,123):

char=chr(ord(char)-32)

elif ord(char) in range(65,91):

char=chr(ord(char)+32)

else: char=char

result+=char

result2=''

for char2 in result:

if char2 in 'aeiou':

char2=chr(ord(char2)+2)

elif char2 in 'AEIOU':

char2=chr(ord(char2)+2)

else: char2=char2

result2+=char2

print(result2)

flipped('Python Hello World! aeiou AEIOU')

flipped('what the FUCK')

flipped('what the FUCK'.swapcase())

#Textbook

def test(strs):

return strs.translate({ord(c):ord(c)+2 for c in "aeiouAEIOU"}).swapcase()

'''

The translate() method returns a string where some specified

characters are replaced with the character described in a dictionary,

or in a mapping table.

syntax: string.translate(dict-like table)

Or use 'maketrans' to make the argument as a table

ex:

txt = "Hello Sam!"

mytable = txt.maketrans("S", "P")

print(txt.translate(mytable))

#output: Hello Pam!

'''

'''

41. Write a Python program to sort numbers based on strings.

Input: six one four one two three

Output:

one two three four six

Input: six one four three two nine eight

Output:

one two three four six eight nine

Input: nine eight seven six five four three two one

Output:

one two three four five six seven eight nine

'''

#Textbook-1

def test(strs):

return ' '.join([x for x in 'one two three four five six seven eight nine'.split() if x in strs])

#this method is so clever,only choose the existing one and skip others

print(test('six one four one two three'))

print(test('nine ten one seven eight fuckyou')) #one seven eight nine

#warmup

print((lambda x: [x for x in '123456789' if x in '54321']) (99999))

# ['1', '2', '3', '4', '5'] (99999沒意義)

#Textbook-2

def test(strs):

nums = 'zero zero zero zero zero zero zero zero zero zero one one one one one one one one one one one one two two two two three three three three three four four four four four four five five five five five five five six six six six six six six seven seven seven seven seven seven seven seven eight eight eight eight eight eight eight eight nine nine nine nine nine nine nine nine nine'.split()

return " ".join([nums[i] for i in sorted([nums.index(x) for x in strs.split()])]).rstrip()

#warmup

nums2='one two three four five'

str2='fuck your music Prince 6ix9ine Drake Kesha Jewel'

print(str2.index('Kesha')) #37

#substring 'Kesha' starts from index=37

'''

42. Write a Python program to find the set of distinct characters

in a given string, ignoring case.

Input: HELLO

Output:

['h', 'o', 'l', 'e']

Input: HelLo

Output:

['h', 'o', 'l', 'e']

Input: Ignoring case

Output:

['s', 'n', 'c', 'o', 'e', 'i', 'r', 'g', 'a', ' ']

'''

def dist\_char(strs):

strs=strs.lower()

print(list(set(strs)))

dist\_char('HELLO')

dist\_char('HelLo')

dist\_char('Ignoring case')

'''43. Write a Python program to find all words in a given string

with n consonants

Input: this is our time

Output:

Number of consonants: 3

Words in the said string with 3 consonants:

['this']

Number of consonants: 2

Words in the said string with 2 consonants:

['time']

Number of consonants: 1

Words in the said string with 1 consonants:

['is', 'our']

'''

def consonants(strs):

for term in strs.split(' '):

nc=len(term)-sum([len(x) for x in term if x in 'aeiouAEIOU'])

print('Number of consonamts: ',nc)

print('Words: \n', term)

consonants('this is our time')

consonants('I really cannot understand what Len is talking about his AIM')

#Textbook-1

def test(strs, n):

return [w for w in strs.split() if sum([c not in "aeiou" for c in w.lower()]) == n]

print(test('this is our time',3))

#Textbook-2

def test(strs, n):

return [w for w in strs.split() if sum(c.lower() not in "aeiou" for c in w) == n]

'''44. Write a Python program to find which characters of

a hexadecimal number correspond to prime numbers

Input: 123ABCD

Output:

[False, True, True, False, True, False, True]

Input: 123456

Output:

[False, True, True, False, True, False]

Input: FACE

Output:

[False, False, False, False]

'''

#warmup

print(hex(15)) # 0xf

print(0xf) # 15

print(int('a',base=16)) # 10 (need to add 'base=16' inside of 'int')

def hex\_prime(strs):

prime=[2,3,5,7,11,13]

# new\_list=list(map(lambda y:'0x'+y, strs))

new\_list=list(map(lambda y:int('0x'+y,base=16), strs))

print('converted list is:\n',new\_list)

print([z in prime for z in new\_list])

hex\_prime('123ABCD')

hex\_prime('123456')

hex\_prime('FACE')

#Textbook

def test(hn):

return [c in "2357BD" for c in hn]

'''

Write a Python program to find all even palindromes up to n.

Output:

Even palindromes up to 50 -

[0, 2, 4, 6, 8, 22, 44]

Even palindromes up to 100 -

[0, 2, 4, 6, 8, 22, 44, 66, 88]

Even palindromes up to 500 -

[0, 2, 4, 6, 8, 22, 44, 66, 88, 202, 212, 222, 232, 242, 252, 262, 272, 282, 292, 404, 414, 424, 434, 444, 454, 464, 474, 484, 494]

Even palindromes up to 2000 -

[0, 2, 4, 6, 8, 22, 44, 66, 88, 202, 212, 222, 232, 242, 252, 262, 272, 282, 292, 404, 414, 424, 434, 444, 454, 464, 474, 484, 494, 606, 616, 626, 636, 646, 656, 666, 676, 686, 696, 808, 818, 828, 838, 848, 858, 868, 878, 888, 898]

'''

#palindrome: '夏興夏', '余小余'

def palin\_upto(n):

assert isinstance(n,int)

return list(i for i in range(1,n+1) if i%2==0 and str(i)[0:len(str(i))]==str(i)[-1::-1])

print(palin\_upto(50))

print(palin\_upto(100))

print(len(palin\_upto(1000000))) # 888

#warmup

str1='hello'

print(str1[0:len(str1)]) # hello

print(str1[-1::-1]) # olleh

#Textbook

def test(n):

return [i for i in range(0,n,2) if str(i) == str(i)[::-1]]

'''

46. Given an array of numbers representing a branch on a binary tree,

write a Python program to find the minimum even value and its index.

In the case of a tie, return the smallest index.

If there are no even numbers, the answer is [].

Input:

[1, 9, 4, 6, 10, 11, 14, 8]

Output:

Minimum even value and its index of the said array of numbers:

[4, 2]

Input:

[1, 7, 4, 4, 9, 2]

Output:

Minimum even value and its index of the said array of numbers:

[2, 5]

Input:

[1, 7, 7, 5, 9]

Output:

Minimum even value and its index of the said array of numbers:

[]

'''

#warmup

# lst1=[1]

# lst2=[2]

# print(lst1+lst2) # [1,2]

# print(min(lst1+lst2)) # 1

def min\_even\_index(lst):

even\_list=[nums for nums in lst if nums%2==0]

if even\_list==[]:

smallest\_even=[]

smallest\_index=[]

else:

smallest\_even=[min(even\_list)]

for i in range(len(lst)):

smallest\_index=[]

if lst[i]==smallest\_even[0]:

smallest\_index.append(i)

break

print(smallest\_even+smallest\_index)

min\_even\_index([1, 9, 4, 6, 10, 11, 14, 8])

min\_even\_index([1, 7, 4, 4, 9, 2])

min\_even\_index([1, 7, 7, 5, 9])

min\_even\_index([5,43,66,81,26])

#Textbook

#同時有access index & value的需求時要記得用enumerate

def test(nums):

if any(n % 2 == 0 for n in nums):

return min([v, i] for i, v in enumerate(nums) if v % 2 == 0)

else:

return []

'''

47. Write a Python program to Filter for the numbers in numbers in

a given list whose sum of digits is > 0,

where the first digit can be negative.

Input:

[11, -6, -103, -200]

Output:

[11, -103]

Input:

[1, 7, -4, 4, -9, 2]

Output:

[1, 7, 4, 2]

Input:

[10, -11, -71, -13, 14, -32]

Output:

[10, -13, 14]

'''

#Textbook

def test(nums):

return [n for n in nums if int(str(n)[:2]) + sum(map(int, str(n)[2:])) > 0]

#若是負數,[:2]是為了screen出前面的負值,而如果是正的則沒差,都是正的

print(test([11, -6, -103, -200]))

print(test([1, 7, -4, 4, -9, 2]))

'''

48. Write a Python program to find the indices of two entries

that show that the list is not in increasing order.

If there are no violations (they are increasing), return an empty list

Input:

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

Output:

[2, 3]

Input:

[1, 2, 3, 4, 5, 6]

Output:

[]

Input:

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

Output:

[4, 5]

Input:

[-3, -2, -3, 0, 2, 3, 4]

Output:

[1, 2]

'''

def decr\_order(lst):

return [[i,i+1] for i in range(len(lst)-1) if lst[i+1]-lst[i]<=0]

print(decr\_order([1, 2, 3, 0, 4, 5, 6]))

print(decr\_order([1, 2, 3, 4, 5, 6]))

print(decr\_order([1, 2, 3, 4, 6, 5, 7]))

print(decr\_order([-3, -2, -3, 0, 2, 3, 4]))

#Textbook

def test(nums):

for i in range(len(nums) - 1):

if nums[i] >= nums[i + 1]:

return [i, i + 1]

return []

'''

49. Write a Python program to find the h-index, the largest positive

number h such that h occurs in the sequence at least h times.

If there is no such positive number return h = -1

Input:

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

Output:

4

Input:

[1, 2, 2, 3, 4, 5, 6]

Output:

2

Input:

[3, 1, 4, 17, 5, 17, 2, 1, 41, 32, 2, 5, 5, 5, 5]

Output:

5

'''

# lst1=[3, 1, 4, 17, 5, 17, 2, 1, 41, 32, 2, 5, 5, 5, 5]

# print(lst1.count(5)) #5

# print(lst1.count(17)) #2

def find\_h(lst):

fit\_result=[]

for num in lst:

if lst.count(num)==num:

fit\_result.append(num)

if fit\_result==[]: return -1

else: return max(fit\_result)

print(find\_h([1, 2, 2, 3, 3, 4, 4, 4, 4]))

print(find\_h([1, 2, 2, 3, 4, 5, 6]))

print(find\_h([3, 1, 4, 17, 5, 17, 2, 1, 41, 32, 2, 5, 5, 5, 5]))

print(find\_h([3,3,3,1.1,4,4,4,4,7,7,7,7,7,7,7]))

print(find\_h([5,7,2,2,2,16]))

#Textbook

def test(nums):

return max([-1] + [i for i in nums if i > 0 and nums.count(i) >= i])

#warmup

print(max([-4]+[5])) #5, not 1

print(max([-1]+[])) #-1

'''

50. Write a Python program to find the even-length words from a given

list of words and sort them by length

Original list of words:

['Red', 'Black', 'White', 'Green', 'Pink', 'Orange']

Find the even-length words and sort them by length in the said list of words:

['Pink', 'Orange']

Original list of words:

['The', 'worm', 'ate', 'a', 'bird', 'imagine', 'that', '!', 'Absurd', '!!']

Find the even-length words and sort them by length in the said list of words:

['!!', 'bird', 'that', 'worm', 'Absurd']

'''

def even\_lgth(lst):

return sorted([word for word in lst if len(word)%2==0], reverse=True)

print(even\_lgth(['Red', 'Black', 'White', 'Green', 'Pink', 'Orange']))

print(even\_lgth(['The', 'worm', 'ate', 'a', 'bird', 'imagine', 'that', '!', 'Absurd', '!!']))

#['worm', 'that', 'bird', 'Absurd', '!!']

# why '!!' is in last order?

#Textbook

def test(words):

return sorted([w for w in words if len(w) % 2 == 0], key=lambda w: len(w))

print(test(['The', 'worm', 'ate', 'a', 'bird', 'imagine', 'that', '!', 'Absurd', '!!']))

#['!!', 'bird', 'that', 'worm', 'Absurd']

'''

51. Write a Python program to find the first n Fibonacci numbers

Input: 10

Output:

[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

Input: 15

Output:

[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]

Input: 50

Output:

[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309, 3524578, 5702887, 9227465, 14930352, 24157817, 39088169, 63245986, 102334155, 165580141, 267914296, 433494437, 701408733, 1134903170, 1836311903, 2971215073, 4807526976, 7778742049, 12586269025]

'''

def Fibonacci(n):

result=[1,1]

for i in range(n-2):

result.append(result[i]+result[i+1])

return(result)

print(Fibonacci(10))

print(Fibonacci(15))

print(Fibonacci(10000)[-1]) #incredible large numbers!

#Textbook

def test(n):

a = [1, 1]

while len(a) < n: a += [sum(a[-2:])]

return a[:n]

'''

52. Write a Python program to reverse the case of all strings.

For those strings, which contain no letters, reverse the strings.

Original list:

['cat', 'catatatatctsa', 'abcdefhijklmnop', '124259239185125', '', 'foo', 'unique']

Reverse the case of all strings. For those strings which contain no letters, reverse the strings:

['CAT', 'CATATATATCTSA', 'ABCDEFHIJKLMNOP', '521581932952421', '', 'FOO', 'UNIQUE']

Original list:

['Green', 'Red', 'Orange', 'Yellow', '', 'White']

Reverse the case of all strings. For those strings which contain no letters, reverse the strings:

['gREEN', 'rED', 'oRANGE', 'yELLOW', '', 'wHITE']

Original list:

['Hello', '!@#', '!@#$', '123#@!']

Reverse the case of all strings. For those strings which contain no letters, reverse the strings:

['hELLO', '#@!', '$#@!', '!@#321']

'''

#warmup

str1='Hello'

print(str1.swapcase())

print(str1==str1.swapcase()) #False

print(str1.isalpha()) #True

str2='1234hello'

print(str2.swapcase()) #1234HELLO

print(str2.isalpha()) #False

str3='123$#%^'

print(str3.swapcase()) #(no change) 123$#%^

def rev\_case(lst):

# rev1=list(map(lambda x:x.swapcase(),lst))

result=[]

for term in lst:

if term==term.swapcase():

term=term[-1::-1]

result.append(term)

else:

term=term.swapcase()

result.append(term)

print(result)

rev\_case(['cat', 'catatatatctsa', 'abcdefhijklmnop', '124259239185125', '', 'foo', 'unique'])

rev\_case(['Hello', '!@#', '!@#$', '123#@!'])

rev\_case(['1234world','helloABC^&%\*'])

#Textbook

def test(strs: list[str]) -> list[str]:

return [s.swapcase() if any(c.isalpha() for c in s) else s[::-1] for s in strs]

# list comprehension 可以是 [ expression-1 if condition-1 else expression-2 for-loop]

print(test(['Hello', '!@#', '!@#$', '123#@!']))

#error?? :TypeError: 'type' object is not subscriptable

#probably pynative online editor version is not Python-3

# -> :只是提示该函数 输入参数 和 返回值 的数据类型 方便程序员阅读代码的。

#warmup

lst4=['hELLO', '#@!', '$#@!', '!@#321']

print([term.swapcase() if len(term)==5 else term[-1::-1] for term in lst4])

#若len=5則swapcase,否則顛倒輸出

'''

53. Write a Python program to find the product of the units digits

in the numbers of a given list

Input:

[12, 23]

Output:

6 (2\*3)

Input:

[12, 23, 43]

Output:

18 (2\*3\*3)

Input:

[113, 234]

Output:

12 (3\*4)

Input:

[1002, 2005]

Output:

10 (2\*5)

'''

import numpy as np

lst1=[1,2,3,4]

print(np.multiply.accumulate(lst1)[-1]) #24

def prod\_unitdigit(lst):

print(np.multiply.accumulate([int(str(num)[-1]) for num in lst])[-1])

prod\_unitdigit([12, 23])

prod\_unitdigit([12, 23, 43])

prod\_unitdigit([113, 234])

prod\_unitdigit([1002, 2005])

prod\_unitdigit([24,335,166,24,50])

#Textbook

def test(nums):

return eval('\*'.join([str(x % 10) for x in nums]))

print(test([12, 23, 43]))

'''

eval: eval(expression,global=none, local=none)

expression 須是個string, e.g. '1+1', 'pow(2,3)'

或像是在jupyter notebook的一個expression

'''

print(eval('np.add.accumulate([1,2,3,4])[-1]')) #10

print(eval('pow(2,5)')) #32

#但print後面有加eval跟沒有加 有何不同? eval()的argument是個string; 請看real python

'''

54. Write a Python program to remove duplicates from a list of

integers, preserving order

Input:

[1, 3, 4, 10, 4, 1, 43]

Output:

[1, 3, 4, 10, 43]

Input:

[10, 11, 13, 23, 11, 25, 23, 76, 99]

Output:

[10, 11, 13, 23, 25, 76, 99]

'''

def rm\_dupl(lst):

return list(set(lst))

print(rm\_dupl([1, 3, 4, 10, 4, 1, 43]))

print(rm\_dupl([10, 11, 13, 23, 11, 25, 23, 76, 99]))

#this one not preserving order (why?)

#Textbook

def test(nums):

return list(dict.fromkeys(nums))

print(test([10, 11, 13, 23, 11, 25, 23, 76, 99]))

'''

55. Write a Python program to find the numbers that are greater than

10 and have odd first and last digits

Input:

[1, 3, 79, 10, 4, 1, 39, 62]

Output:

[79, 39]

Input:

[11, 31, 77, 93, 48, 1, 57]

Output:

[11, 31, 77, 93, 57]

'''

def odd\_fl(lst):

return [num for num in lst if num>10 and num%2==1 and int(str(num)[0])%2!=0]

print(odd\_fl([1, 3, 79, 10, 4, 1, 39, 62]))

print(odd\_fl([11, 31, 77, 93, 48, 1, 57]))

#Textbook-1

def test(nums):

return [x for x in nums if x > 10 and x % 10 % 2 and int(str(x)[0]) % 2]

#Textbook-2

def test(nums):

return [n for n in nums if n > 10 and (int(str(n)[0]) \* int(str(n)[-1])) % 2]

'''

56. Write a Python program to find an integer exponent x such that

a^x = n

Input:

a = 2 : n = 1024

Output:

10

Input:

a = 3 : n = 81

Output:

4

Input:

a = 3 : n = 1290070078170102666248196035845070394933441741644993085810116441344597492642263849

Output:

170

'''

#I try to write a code to find the 'closest'

def find\_exp(a,n):

i=0

while a\*\*i<n:

i+=1

return i

print(find\_exp(2,1024))

print(find\_exp(3,81))

print(find\_exp(3,4561465892446896585456))

print(find\_exp(2,1000000))

'''

57. Write a Python program to find the sum of the magnitudes of the

elements in the array with a sign that is equal to the product of the

signs of the entries.

Input:

[1, 3, -2]

Output:

-6

Input:

[1, -3, 3]

Output:

-7

Input:

[10, 32, 3]

Output:

45

Input:

[-25, -12, -23]

Output:

-60

'''

lst1=[1,3,-2]

print(sum(map(abs,lst1)))

def smsp(lst):

mag=sum(map(abs,lst))

prod=1

sign=''

for num in lst:

prod=prod\*num

if prod<0: sign='-'

print(sign+str(mag))

smsp([1,3,-2])

smsp([1, -3, 3])

smsp([10, 32, 3])

smsp([-25, -12, -23])

#Textbook

def test(nums):

tot = sum(abs(i) for i in nums)

if all(nums):

return tot if sum(i < 0 for i in nums) % 2 == 0 else -tot

return 0

#sum(i<0 for i in nums)%2==0: 負數的個數是偶數

'''

58. Write a Python program to find the biggest even number between

two numbers inclusive

Input:

m = 12

n = 51

Output:

50

Input:

m = 1

n = 79

Output:

78

Input:

m = 47

n = 53

Output:

52

Input:

m = 100

n = 200

Output:

200

'''

def ben(m,n):

return max(m,n) if max(m,n)%2==0 else max(m,n)-1

print(ben(12,51))

print(ben(79,1))

print(ben(47,53))

print(ben(200,100))

'''59. A valid filename should end in .txt, .exe, .jpg, .png, or .dll,

and should have at most three digits, no additional periods.

Write a Python program to create a list of True/False that determine

whether candidate filename is valid or not

Input:

['abc.txt', 'windows.dll', 'tiger.png', 'rose.jpg', 'test.py', 'win32.exe']

Output:

['Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes']

Input:

['.txt', 'windows.exe', 'tiger.jpeg', 'rose.c', 'test.java']

Output:

['No', 'Yes', 'No', 'No', 'No']

'''

str1='abc.txt'

print(len(str1.split('.')[-1]))

def vf(lst):

result=[]

for fn in lst:

if len(fn.split('.')[-1])==3:result.append('Yes')

else:result.append('No')

return result

print(vf(['abc.txt', 'windows.dll', 'tiger.png', 'rose.jpg', 'test.py', 'win32.exe']))

print(vf(['.txt', 'windows.exe', 'tiger.jpeg', 'rose.c', 'test.java']))

#Textbook

def test(file\_names):

return ["Yes" if

f.split(".")[1:] in [['txt'], ['png'], ['dll'], ['exe'], ['jpg']] and f[0].isalpha() and sum(c.isdigit() for c in f) < 4

else "No"

for f in file\_names]

'''

60. Write a Python program to find a list of all numbers that are

adjacent to a prime number in the list, sorted without duplicates.

Input:

[2, 17, 16, 0, 6, 4, 5]

Output:

[2, 4, 16, 17]

Input:

[1, 2, 19, 16, 6, 4, 10]

Output:

[1, 2, 16, 19]

Input:

[1, 2, 3, 5, 1, 16, 7, 11, 4]

Output:

[1, 2, 3, 4, 5, 7, 11, 16]

'''

def is\_prime(n):

result=True

if n==0:result=False

if n>2:

for i in range(2,n):

if n%i==0: result=False

return result

def adj\_prime(lst):

result1=[lst[j] for j in range(len(lst)-1) if is\_prime(lst[j+1])]

#只抓後面是質數的條件 沒顧到前面是質數的情況

result2=[lst[k] for k in range(1,len(lst)) if is\_prime(lst[k-1])]

return list(sorted(set(result1+result2)))

# idx\_p=[i for i in range(len(lst)) if is\_prime(lst[i])]

# result1=[]

# for j in idx\_p[1:-1]:

# result1.append(lst[j-1])

# result1.append(lst[j+1])

# result2=list(set(result1))

# return result2

print(adj\_prime([2, 17, 16, 0, 6, 4, 5])) #[2, 4, 16, 17]

print(adj\_prime([1, 2, 19, 16, 6, 4, 10]))

print(adj\_prime([1, 2, 3, 5, 1, 16, 7, 11, 4]))

#Textbook

def test(nums):

return sorted({

n for i, n in enumerate(nums)

if (i > 0 and prime(nums[i - 1])) or (i < len(nums) - 1 and prime(nums[i + 1]))

})

def prime(m):

if (m>0):

return all(m % i for i in range(2, m - 1))

nums = [2, 17, 16, 0, 6, 4, 5]

print("Original list of numbers:")

print(nums)

print("Numbers that are adjacent to a prime number in the said list, sorted without duplicates:")

print(test(nums))

'''

61. Write a Python program to find the number which when appended to

the list makes the total 0

Input:

[1, 2, 3, 4, 5]

Output:

-15

Input:

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

Output:

5

Input:

[10, 42, 17, 9, 1315182, 184, 102, 29, 15, 39, 755]

Output:

-1316384

'''

def amt0(lst):

return -sum(lst)

print(amt0([1, 2, 3, 4, 5]))

print(amt0([-1, -2, -3, -4, 5]))

print(amt0([10, 42, 17, 9, 1315182, 184, 102, 29, 15, 39, 755]))

'''

62. Write a Python program to find the dictionary key whose case is

different than all other keys

Input:

{'red': '', 'GREEN': '', 'blue': 'orange'}

Output:

GREEN

Input:

{'RED': '', 'GREEN': '', 'orange': '#125GD'}

Output:

orange

'''

#warmup

dict1={'red': '', 'GREEN': '', 'blue': 'orange'}

lst1=list(dict1.keys())

chk1=[term.isupper() for term in lst1] #[False,True,False]

chk\_term=True

if chk1.count(False)==1: chk\_term=False

for i,n in enumerate(chk1):

if n==chk\_term:

print(lst1[i])

break

print('result is vague,cannot decide')

def cid(dct):

key\_lst=list(dct.keys())

chk\_lst=[term.isupper() for term in key\_lst]

chk\_result=True

if chk\_lst.count(False)==1: chk\_result=False

print(chk\_result)

for i,n in enumerate(chk\_lst):

if n==chk\_result:

print(key\_lst[i])

break

#print('result is vague,cannot decide') # no-use!

cid({'red': '', 'GREEN': '', 'blue': 'orange'})

cid({'RED': '', 'GREEN': '', 'orange': '#125GD'})

cid({'BEATLES':'Paul','Maroon5':'Adam','BANGLES':'Suzan'})

#Maroon5

cid({'BEATLES':'Paul','MAROOn5':'Adam','BANGLES':'Suzan'})

#MAROOn5

cid({'BEATLES':'Paul','MAROON5':'Adam','BANGLES':'Suzan'})

#BEATLES (because it's the first, and default is True)

#Textbook

def test(dict\_data):

for different in dict\_data:

if all(k.islower() != different.islower() for k in dict\_data if k != different):

return different

'''different跟群組每一個比 '是否是lower case':

結果是不一樣的 才是different

但不能跟自己比, 所以 if k!=different '''

dict\_data = {"red": "", "GREEN": "", "blue": "orange"}

print("Original dictionary key-values:")

print(dict\_data)

print("Find the dictionary key whose case is different than all other keys:")

print(test(dict\_data))

'''

63. Write a Python program to find the sum of the even elements

that are at odd indices in a given list

Input:

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

Output:

12

Input:

[1, 2, 8, 3, 9, 4]

Output:

6

'''

def eeoi(lst):

new\_lst=[n for i,n in enumerate(lst) if n%2==0 and i%2==1]

return sum(new\_lst)

print(eeoi([1, 2, 3, 4, 5, 6, 7]))

print(eeoi([1, 2, 8, 3, 9, 4]))

#Textbook-1

def test(nums):

return sum(i for i in nums[1::2] if i % 2 == 0)

#Textbook-2

def test(nums):

return sum([nums[i] for i in range(len(nums)) if i % 2 == 1 and nums[i] % 2 == 0])

'''

64. Write a Python program to find the string consisting of all the

words whose lengths are prime numbers

Input:

The quick brown fox jumps over the lazy dog.

Output:

The quick brown fox jumps the

Input:

Omicron Effect: Foreign Flights Won't Resume On Dec 15, Decision Later.

Output:

Omicron Effect: Foreign Flights Won't On Dec 15,

'''

def is\_prime(n):

result=True

for i in range(2,n):

if n%i==0:

result=False

break

return result

def LIPN(sentence):

lst1=sentence.split(' ')

final\_strs=''

for term in lst1:

if is\_prime(len(term))==True:

final\_strs+=term+' '

return final\_strs

print(LIPN('The quick brown fox jumps over the lazy dog.'))

print(LIPN("Omicron Effect: Foreign Flights Won't Resume On Dec 15, Decision Later."))

#Textbook

def test(strs):

return " ".join(strs for strs in strs.split() if is\_prime(len(strs)))

def is\_prime(n):

return n > 1 and all(n % j for j in range(2, int(n \*\* 0.5) + 1))

#一個整數在開根號以上到該 整數-1 之間 一定都是不能整除的,所以也不用verify了

'''

65. Write a Python program to shift the decimal digits n places to

the left, wrapping the extra digits around.

If shift > the number of digits of n, reverse the string.

Input:

n = 12345 and shift = 1

Output:

Result = 23451

Input:

n = 12345 and shift = 2

Output:

Result = 34512

Input:

n = 12345 and shift = 3

Output:

Result = 45123

Input:

n = 12345 and shift = 5

Output:

Result = 12345

Input:

n = 12345 and shift = 6

Output:

Result = 54321

'''

def sdd(n,shift):

result=''

if shift<=len(str(n)):

return ''.join(digit for digit in (str(n)[shift:]+str(n)[:shift]))

else: return ''.join(digit for digit in str(n)[-1::-1])

print(sdd(12345,1))

print(sdd(12345,2))

print(sdd(12345,3))

print(sdd(12345,5))

print(sdd(12345,6))

print(sdd('Hello World',8))

#Textbook

def test(n, shift):

shifted\_digits = [int(x) for x in str(n)]

print(shifted\_digits)

for i in range(shift):

shifted\_digits.append(shifted\_digits.pop(0))

#取走(pop)第一個(此時原序列第一個已不見) 再append到後面

print(shifted\_digits)

if shift > len(shifted\_digits):

return str(n)[::-1]

else:

return ''.join(str(x) for x in shifted\_digits)

n = 12345

shift = 2

print("\nn =",n," and shift =",shift)

print("Result = ",test(n, shift))

#warmup

lst2=list('abcde')

print(lst2.pop(0)) #a

print(lst2) #['b','c','d','e']

'''

66. Write a Python program to find the indices of the closest pair

from a list of numbers

Input: [1, 7, 9, 2, 10]

Output:

[0, 3]

Input: [1.1, 4.25, 0.79, 1.0, 4.23]

Output:

[4, 1]

Input: [0.21, 11.3, 2.01, 8.0, 10.0, 3.0, 15.2]

Output:

[2, 5]

'''

#Textbook

def test(nums):

closest\_inds = None

closest\_dist = None

#當有定義一個undecided-now or TBD的需求時 可用None

for ind, num in enumerate(nums):

for other\_ind, num2 in enumerate(nums):

if num != num2 and ((closest\_dist is None) or abs(num - num2) < closest\_dist):

closest\_dist = abs(num - num2)

closest\_inds = [ind, other\_ind]

if num <= num2:

closest\_inds = [ind, other\_ind]

else:

closest\_inds = [other\_ind, ind]

#這裡是以值的大小做index的排序

return closest\_inds

nums = [1, 7, 9, 2, 10]

print("List of numbers:",nums)

print("Indices of the closest pair from the said list of numbers:")

print(test(nums))

#是取第一對發現的

nums=[5,2,6,6,8,10,11]

print(test(nums))

'''

67. Write a Python program to find a string which, when each character

is shifted (ASCII incremented) by shift, gives the result

Input:

Ascii character table

Shift = 1

Output:

@rbhh­bg`q`bsdq­s`akd

Input:

Ascii character table

Shift = -1

Output:

Btdjj!dibsbdufs!ubcmf

'''

def ashift(strs,shift):

return ''.join([chr(ord(char)-shift) for char in strs])

print(ashift('Ascii character table',1))

print(ashift('Ascii character table',-1))

secret='Hitler is a gay' # -> encode: ??? -> try to decode

# shift key: range of a month date

import random

skey=random.randint(1,32)

captured\_code=ashift(secret,shift=skey)

print('the captured code is:\n',captured\_code)

print('try to decode!\n')

#decode:

for guess in range(32):

print(ashift(captured\_code,shift=-guess))

'''

68. Write a Python program to find all 5's in integers less than n

that are divisible by 9 or 15

Input:

Value of n = 50

Output:

[[15, 1], [45, 1]]

Input:

Value of n = 65

Output:

[[15, 1], [45, 1], [54, 0]]

Input:

Value of n = 75

Output:

[[15, 1], [45, 1], [54, 0]]

Input:

Value of n = 85

Output:

[[15, 1], [45, 1], [54, 0], [75, 1]]

Input:

Value of n = 150

Output:

[[15, 1], [45, 1], [54, 0], [75, 1], [105, 2], [135, 2]]

'''

#sublist後面的0,1,2是啥意思? index position

def all5(n):

return [[num for num in range(n) if (num%9==0 or num%15==0) and '5' in str(num)]]

print(all5(50))

#cannot simply combine index position in one line

#Textbook

def test(n):

return [[i,j] for i in range(n) for j in range(len(str(i))) if str(i)[j] == '5' and (i%15==0 or i%9==0)]

print(test(50))

'''69. Write a Python program to create a new string by taking a string,

and word by word rearranging its characters in ASCII order.

Input: Ascii character table

Output:

Aciis aaccehrrt abelt

Input: maltos won

Output:

almost now

'''

def sb\_ascii(strs):

result=''

for word in strs.split(' '):

result+=''.join(sorted(word,key=lambda x:ord(x)))+' '

print(result)

sb\_ascii('Ascii character table')

sb\_ascii('maltos won')

#Textbook

def test(strs):

return " ".join("".join(sorted(w)) for w in strs.split(' '))

#warmup

word1='fuckyou'

print(''.join(sorted(word1)))

'''

70. Write a Python program to find the first negative balance

from a given a list of numbers which represent bank deposits

and withdrawals

Input:

[[12, -7, 3, -89, 14, 88, -78], [-1, 2, 7]]

Output:

[-81, -1]

Input:

[[1200, 100, -900], [100, 100, -2400]]

Output:

[None, -2200]

'''

result=[None for i in range(5)]

print(result)

def nb(lst):

result=[None for j in range(len(lst))]

for i,seq in enumerate(lst):

sums=0

for nums in seq:

sums+=nums

if sums<0:

result[i]=sums

break

print(result)

nb([[12, -7, 3, -89, 14, 88, -78], [-1, 2, 7]])

nb([[1200, 100, -900], [100, 100, -2400]])

#Textbook

def test(balances):

firsts = []

for bals in balances:

total = 0

for b in bals:

total += b

if total < 0:

firsts.append(total)

break

else:

firsts.append(None)

return firsts

balances = [[12, -7, 3, -89, 14, 88, -78], [-1, 2, 7]]

print("Bank deposits and withdrawals:")

print(balances)

print("\nFirst negative balance of deposits and withdrawals:")

print(test(balances))

'''

71. Given a list of numbers and a number to inject, write a Python

program to create a list containing that number in between each pair

of adjacent numbers

Input: [12, -7, 3, -89, 14, 88, -78, -1, 2, 7]

Separator: 6

Output:

[12, 6, -7, 6, 3, 6, -89, 6, 14, 6, 88, 6, -78, 6, -1, 6, 2, 6, 7]

Input: [1, 2, 3, 4, 5, 6]

Separator: 9

Output:

[1, 9, 2, 9, 3, 9, 4, 9, 5, 9, 6]

'''

#warmup

lst1=[1,2,3,4,5,6]

lst1.insert(1,10)

print(lst1) #[1, 10, 2, 3, 4, 5, 6]

lst1.insert(3,10)

print(lst1) #[1, 10, 2, 10, 3, 4, 5, 6]

def insert(lst,n):

i=1

while i<len(lst):

lst.insert(i,n)

i+=2

print(lst)

insert([12, -7, 3, -89, 14, 88, -78, -1, 2, 7],6)

insert([1, 2, 3, 4, 5, 6],9)

#Textbook-1

def test(nums, sep):

ans = [sep] \* (2 \* len(nums) - 1)

print(ans)

ans[::2] = nums

return ans

nums = [1, 2, 3, 4, 5, 6]

separator = 9

print("\nList of numbers:",nums)

print("Separator:",separator)

print("Inject the separator in between each pair of adjacent numbers of the said list:")

print(test(nums,separator))

#Textbook-2

def test(nums, sep):

result = []

for i in range(len(nums)):

if i == len(nums) - 1:

result.append(nums[i])

else:

result.append(nums[i])

result.append(sep)

return result

'''72. Write a Python program to find the indices of three numbers

that sum to 0 in a given list of numbers

Input: [12, -7, 3, -89, 14, 4, -78, -1, 2, 7]

Output:

[1, 2, 5]

Input: [1, 2, 3, 4, 5, 6, -7]

Output:

[2, 3, 6]

'''

def sumto0(lst):

result=[]

for i in range(len(lst)):

for j in range(len(lst)):

for k in range(len(lst)):

if i!=j and j!=k and i!=k and lst[i]+lst[j]+lst[k]==0:

result.append((i,j,k))

print(result)

sumto0([12, -7, 3, -89, 14, 4, -78, -1, 2, 7]) #duplicated!

sumto0([1, 2, 3, 4, 5, 6, -7])

#Textbook

def test(nums):

inv = {n: i for i, n in enumerate(nums)} # note that later duplicates will override earlier entries

for i, n in enumerate(nums):

if inv[n] == i:

del inv[n]

if any((-m - n) in inv for m in nums[:i]): # found solution!

j, m = next((j, m) for j, m in enumerate(nums) if (-m - n) in inv)

k = inv[-m - n]

return sorted([i, j, k])

nums = [12, -7, 3, -89, 14, 4, -78, -1, 2, 7]

print("List of numbers:",nums)

print("Indices of three numbers that sum to 0 in the said list:")

print(test(nums))

#網友:

def test1(nums):

from itertools import combinations

for i, j, k in combinations(nums, 3):

if i + j == -k:

return [nums.index(i), nums.index(j), nums.index(k)]

return []

print(test1([12, -7, 3, -89, 14, 4, -78, -1, 2, 7]))

'''

73. Write a Python program to find a substring in a given string

contains a vowel between two consonants

Input: Hello

Output:

Hel

Input: Sandwhich

Output:

San

Input: Python

Output:

hon

'''

def vbc(strs):

lst1=[i for i,c in enumerate(strs) if c in 'aeiouAEIOU']

# print(lst1)

for idx in lst1:

if idx==0 or idx==len(strs)-1: pass

elif (strs[idx-1] not in 'aeiouAEIOU') and (strs[idx+1] not in 'aeiouAEIOU'):

print (strs[idx-1]+strs[idx]+strs[idx+1])

print(vbc('Hello'))

print(vbc('Sandwhich'))

print(vbc('Python'))

print(vbc('BengalsGoesToSuperBowl'))

#Textbook

def test(s):

cons = "bcdfghjklmnpqrstvwxz"

vows = "aeiou"

return next(s[i - 1:i + 2] for i in range(1, len(s) - 1)

if s[i].lower() in vows and s[i - 1].lower() in cons and s[i + 1].lower() in cons)

#next的用意在於秀出(下一個only)真實的值 而非object

strs = "Sandwhich"

print("\nOriginal string:",strs)

print("Find a vowel between two consonants, contained in said string:")

print(test(strs))

#warmup

def func():

print('\*\*\*\*\*Pickett passes..\*\*\*\*\*\*\*\*')

print('when Friermuth catches the ball..')

yield 'muth !!'

print('\*\*\*referee checks\*\*\*')

yield 'silence.. one..two..three'

print('it is a good catch!')

catch=func()

print(next(catch)) #停在yield 那行

print('wow! what a catch is it good??')

print(next(catch)) #這個next 從yield之後執行下面的

print('nervous')

print(next(catch))

print(next(catch)) #None

'''74. Write a Python program to find a string consisting of

space-separated characters with given counts

Input: {'f': 1, 'o': 2}

Output:

f o o

Input: {'a': 1, 'b': 1, 'c': 1}

Output:

a b c

'''

def sscwgc(dict1):

lst1=list(dict1.keys())

lst2=list(dict1.values())

return ' '.join(lst1[i]\*lst2[i] for i in range(len(lst1)))

print(sscwgc({'f': 1, 'o': 2})) #有bug, oo之間沒空格

print(sscwgc({'a': 1, 'b': 1, 'c': 1}))

#my retry

def sscwgc2(dict2):

enum=dict2.items()

return ' '.join(k for k,v in enum for i in range(v))

print(sscwgc2({'f': 1, 'o': 2}))

#Textbook

def test(counts):

return " ".join(c for c, i in counts.items() for \_ in range(i))

strs = {"f": 1, "o": 2}

print("Original string:",strs)

print("String consisting of space-separated characters with given counts:")

print(test(strs))

#warmup- iter, \*

obj1={'a':'Hamlin','b':['Higgins','Boyd'],'c':'Poyer', 'd':' ', 'e':'stop'}

print(\*iter(obj1)) #a b c d e

'''

75. Write a Python program to reorder numbers from a give array in

increasing/decreasing order based on whether the first plus last

element is odd/even

Reorder numbers of a give array in increasing/decreasing order

based on whether the first plus last element is odd/even.:

Input:

[3, 7, 4]

Output:

[3, 4, 7]

Input:

[2, 7, 4]

Output:

[7, 4, 2]

Input:

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

Output:

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

Input:

[1, 5, 6, 7, 4, 2, 9]

Output:

[9, 7, 6, 5, 4, 2, 1]

'''

#warmup

lst1=[1, 5, 6, 7, 4, 2, 8]

print(sorted(lst1))

print(sorted(lst1,reverse=True))

def resort(lst):

if (lst[0]+lst[-1])%2!=0: return sorted(lst)

return sorted(lst,reverse=True)

print(resort([3, 7, 4]))

print(resort([2, 7, 4]))

print(resort([1, 5, 6, 7, 4, 2, 8]))

print(resort([1, 5, 6, 7, 4, 2, 9]))

#Textbook

def test(nums):

return sorted(nums, reverse=(False if (nums[0] + nums[-1]) % 2 else True))

#argument裡面也可以用if的承述句表達

'''

76. Write a Python program to find the index of the largest prime

in the list and the sum of its digits

Input: [3, 7, 4]

Output:

[1, 7]

Input: [3, 11, 7, 17, 19, 4]

Output:

[4, 10]

Input: [23, 17, 201, 14, 10473, 43225, 421, 423, 11, 10, 2022, 342157]

Output:

[6, 7]

'''

import itertools as it

#warmup

n0=198

print(\*map(int,str(n0))) #1 9 8

print(sum(map(int,str(n0))))

def is\_prime(n):

return n>1 and all(n%j for j in range(2,int(n\*\*0.5)+1))

def lpsd(lst):

max\_prime=max([num for i,num in enumerate(lst) if is\_prime(num)])

return [i for i,nums in enumerate(lst) if nums==max\_prime]+[sum(map(int,str(nums))) for i,nums in enumerate(lst) if nums==max\_prime]

print(lpsd([3, 7, 4]))

print(lpsd([3, 11, 7, 17, 19, 4]))

print(lpsd([23, 17, 201, 14, 10473, 43225, 421, 423, 11, 10, 2022, 342157]))

#Textbook

def test(nums):

n, i = max((n, i) for i, n in enumerate(nums) if is\_prime(n))

return [i, sum(int(c) for c in str(n))]

def is\_prime(n):

return n > 1 and all(n % j for j in range(2, int(n \*\* 0.5) + 1))

#類似list comprehesion回傳多個變數時 包裝成tuple

lst2=[4,7,3,8,9,2]

n,i=max((n,i) for i,n in enumerate(lst2))

print(n,i) #9 4

i,n=max((i,n) for i,n in enumerate(lst2))

print(i,n) #5 2

#所以max是以頭一個變數做為判定標準

'''

77. Write a Python program to convert GPAs to letter grades according

to the following table (GPA:Grade Point Average)

GPAs Grades

4.0: A+

3.7: A

3.4: A-

3.0: B+

2.7: B

2.4: B-

2.0: C+

1.7: C

1.4: C-

below: F

Input:

[4.0, 3.5, 3.8]

Output:

['A+', 'A-', 'A']

Input:

[5.0, 4.7, 3.4, 3.0, 2.7, 2.4, 2.0, 1.7, 1.4, 0.0]

Output:

['A+', 'A+', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'F']

'''

#網友-1

def test(nums):

gpas = {4.0: 'A+', 3.7: 'A', 3.4: 'A-', 3.0: 'B+', 2.7: 'B', 2.4: 'B-', 2.0: 'C+', 1.7: 'C', 1.4: 'C-', 0: 'F'}

grades = []

for num in nums:

for gpa, lett in gpas.items():

if num >= gpa:

grades.append(lett)

break

#加break的意思 在於 從上往下check,一旦符合就break

return grades

print(test([4.0, 3.5, 3.8]))

#網友-2

def test(nums):

gpas = {4.0: 'A+', 3.7: 'A', 3.4: 'A-', 3.0: 'B+', 2.7: 'B',2.4: 'B-', 2.0: 'C+', 1.7: 'C', 1.4: 'C-', 0: 'F'}

return [gpas[max(gpa for gpa in gpas if num >= gpa)] for num in nums]

print(test([4.0, 3.5, 3.8]))

#網友-3

def test(nums):

gpas = {4.0: 'A+', 3.7: 'A', 3.4: 'A-', 3.0: 'B+', 2.7: 'B',2.4: 'B-', 2.0: 'C+', 1.7: 'C', 1.4: 'C-', 0: 'F'}

return [gpas[next(gpa for gpa in gpas if num >= gpa)] for num in nums]

'''

78. Write a Python program to find the two closest distinct numbers

in a given a list of numbers

Input:

[1.3, 5.24, 0.89, 21.0, 5.27, 1.3]

Output:

[5.24, 5.27]

Input:

[12.02, 20.3, 15.0, 19.0, 11.0, 14.99, 17.0, 17.0, 14.4, 16.8]

Output:

[14.99, 15.0]

'''

def cdn(lst):

gap=abs(lst[0]-lst[1])

result=[]

for i in lst:

for j in lst:

if i!=j and abs(i-j)<gap:

gap=abs(i-j)

result=[i,j]

return result

print(cdn([1.3, 5.24, 0.89, 21.0, 5.27, 1.3]))

print(cdn([12.02, 20.3, 15.0, 19.0, 11.0, 14.99, 17.0, 17.0, 14.4, 16.8]))

#Textbook

def test(nums):

s = sorted(set(nums))

return min([[a, b] for a, b in zip(s, s[1:])], key=lambda x: x[1] - x[0])

#warmup

lst1=[5,4,8,2,7]

lst2=[13,16,19,11,10]

print(min([[n1,n2] for n1,n2 in zip(lst1,lst2)],key=lambda x:abs(x[1]-x[0])))

#取出 兩者差異最接近

print(min([[n1,n2] for n1,n2 in zip(lst1,lst2)],key=lambda x:abs(x[1]\*x[0])))

#取出 兩者乘積值最小

'''79. Write a Python program to find the largest negative and

smallest positive numbers (or 0 if none)

Input:

[-12, -6, 300, -40, 2, 2, 3, 57, -50, -22, 12, 40, 9, 11, 18]

Output:

[-6, 2]

Input:

[-1, -2, -3, -4]

Output:

[-1, 0]

Input:

[1, 2, 3, 4]

Output:

[0, 1]

Input:

[]

Output:

[0, 0]

'''

print(list(((i,n) for i,n in enumerate(range(1,3)))))

def lnsp(lst):

# print(min((nn for nn in lst if nn<0),key=lambda x:abs(x))) #-6

if [nn for nn in lst if nn<0]==[]: ln=0

else:ln=min((nn for nn in lst if nn<0),key=lambda x:abs(x))

if [pn for pn in lst if pn>0]==[]: sp=0

else:sp=min((pn for pn in lst if pn>0),key=lambda x:abs(x))

print ([ln,sp])

lnsp([-12, -6, 300, -40, 2, 2, 3, 57, -50, -22, 12, 40, 9, 11, 18])

lnsp([-1, -2, -3, -4])

lnsp([1, 2, 3, 4])

lnsp([])

#Textbook

def test(nums):

pos = [n for n in nums if n > 0]

neg = [n for n in nums if n < 0]

return [max(neg) if neg else 0, min(pos) if pos else 0]

#if [] : False ; if [non-empty iterable]: True

tup1=(1,2,3)

tup2=()

if tup1: print('non-empty')

if not tup2: print('empty')

'''80. Write a Python program to round each float in a given list of

number up to the next integer and return the running total of

the integer squares

Input: [2.6, 3.5, 6.7, 2.3, 5.6]

Output:

[9, 25, 74, 83, 119]

Input: [301.1, 401.4, -23.1, 13554122.0, 10201.0101, 10000000.0]

Output:

[91204, 252808, 253337, 183714223444221, 183714327525025, 283714327525025]

'''

#warmup (題目是無條件進位)

import math

print(math.ceil(3.2758)) #4

print(round(3.2724,3)) #3.272

from itertools import accumulate as ia

def round\_sq(lst):

return list(ia(map(lambda x:x\*\*2,list(map(math.ceil,lst)))))

print(round\_sq([2.6, 3.5, 6.7, 2.3, 5.6]))

print(round\_sq([301.1, 401.4, -23.1, 13554122.0, 10201.0101, 10000000.0]))

'''81. Write a Python program to calculate the average of the numbers

a through b ( b not included ) rounded to nearest integer,

in binary (or -1 if there are no such numbers)

Input:

4 , 7

Output:

0b101

Input:

11 , 19

Output:

0b1110

'''

#warmup

# print(bin(2.54)) #float cannot be interpreted as integer

print(bin(-4))

import statistics as st

def rtniib(a,b):

try:

return bin(round(st.mean(range(a,b))))

except:

return -1

print(rtniib(4,7))

print(rtniib(27,124))

print(rtniib(8,5)) #-1

print(rtniib(3.68,12.37)) #-1

#Textbook

def test(a,b):

r = range(a, b)

if len(r) == 0:

return "-1"

return bin(round(sum(r) / len(r)))

a = 4

b = 7

print("Range:",a,",",b)

print("Average of the numbers",a,"through",b,"rounded to nearest integer, in binary:")

print(test(a, b))

'''

Note:

https://www.geeksforgeeks.org/python-program-to-convert-floating-to-binary/

Python doesn’t provide any inbuilt method to easily convert floating point decimal numbers to binary number. So, Let’s do this manually.

'''

'''

82. Write a Python program to find the sublist of numbers from

a given list of numbers with only odd digits in increasing order.

Input:

[1, 3, 79, 10, 4, 2, 39]

Output:

[1, 3, 39, 79]

Input:

[11, 31, 40, 68, 77, 93, 48, 1, 57]

Output:

[1, 11, 31, 57, 77, 93]

Input:

[9, -2, 3, 4, -2, 0, 2, -3, 8, -1]

Output:

[-3, -1, 3, 9]

'''

def odiio(lst):

return sorted([num for num in lst if num%2!=0])

print(odiio([1, 3, 79, 10, 4, 2, 39]))

print(odiio([11, 31, 40, 68, 77, 93, 48, 1, 57]))

print(odiio([9, -2, 3, 4, -2, 0, 2, -3, 8, -1]))

#Textbook

def test(nums):

return sorted(n for n in nums if all(int(c) % 2 for c in str(abs(n))))

'''

83. A string is happy if every three consecutive characters are

distinct. Write a Python program to find two indices making

a given string unhappy

Input:

Python

Output:

None

Input:

Unhappy

Output:

[4, 5]

Input:

Find

Output:

None

Input:

Street

Output:

[3, 4]

'''

def unhappy(strs):

return [i for i in range(len(strs)-1) if strs[i]==strs[i+1]]+[i+1 for i in range(len(strs)-1) if strs[i]==strs[i+1]]

print(unhappy('Python'))

print(unhappy('Unhappy'))

print(unhappy('Find'))

print(unhappy('Street'))

#未考慮i & i+2 相同讓它成為 unhappy的情況

#Textbook

def test(s):

for i in range(len(s) - 2):

if s[i] == s[i + 1]:

return [i, i + 1]

if s[i] == s[i + 2]:

return [i, i + 2]

'''

84. Write a Python program to find the index of the

matching parentheses for each character in a given string.

Input: ()(())

Output:

[1, 0, 5, 4, 3, 2]

Input: ()()()

Output:

[1, 0, 3, 2, 5, 4]

Input: ((()))

Output:

[5, 4, 3, 2, 1, 0]

'''

'''

algorithm: (:+1 ; ):-1

()(()): 1 0 1 2 1 0 => (i,v) enumerate

=> (0,1) (1,0), (2,1) (3,2) (4,1) (5,0)

整理出的algorithm: 先抓value=0,1 => 1,0,5,2

再抓value=1,2 => insert在(2,1)之前 => 1,0,5,4,3,2

((())) => (0,1) (1,2) (2,3) (3,2) (4,1) (5,0)

value pair: 0&1 ; 1&2 ; 2&3

'''

def match(strs):

chk=[]

v\_acc=0

for i in range(len(strs)):

if strs[i]=='(':

v\_acc+=1

chk.append((i,v\_acc))

elif strs[i]==')':

v\_acc-=1

chk.append((i,v\_acc))

return chk

print(match('()(())'))

print(match('()()()'))

print(match('((()))'))

#return chk #有成功generate出(i,v) tuple

#接下來太複雜 放棄

#Textbook

def test(parens):

a = list(parens)

stack = []

for i, c in enumerate(a):

if c == "(":

stack.append(i)

else:

a[stack[-1]] = i

a[i] = stack.pop()

return a

parens = "()(())"

print("Original parentheses:",parens)

print("Index of the matching parentheses for each character in a given string:")

print(test(parens))

'''

85. Write a Python program to find an increasing sequence

consisting of the elements of the original list

Input:

[1, 3, 79, 10, 4, 2, 39]

Output:

[1, 2, 3, 4, 10, 39, 79]

Input:

[11, 31, 40, 68, 77, 93, 48, 1, 57]

Output:

[1, 11, 31, 40, 48, 57, 68, 77, 93]

Input:

[9, -2, 3, 4, -2, 0, 2, -3, 8, -1]

Output:

[-3, -2, -1, 0, 2, 3, 4, 8, 9]

'''

def inc\_seq(lst):

return sorted(lst)

print(inc\_seq([1, 3, 79, 10, 4, 2, 39]))

print(inc\_seq([11, 31, 40, 68, 77, 93, 48, 1, 57]))

print(inc\_seq([9, -2, 3, 4, -2, 0, 2, -3, 8, -1]))

#Textbook

def test(nums):

result = sorted(set(nums))

return result

'''

86. Write a Python program to find the vowels from each of the original

texts (y counts as a vowel at the end of the word) from a given list

of strings

Input: ['w3resource', 'Python', 'Java', 'C++']

Output:

['eoue', 'o', 'aa', '']

Input: ['ably', 'abruptly', 'abecedary', 'apparently', 'acknowledgedly']

Output:

['ay', 'auy', 'aeeay', 'aaey', 'aoeey']

'''

#warmup

for i in range(7):

for j in range(i):

print(j,end=',')

print([j for i in range(7) for j in range(i)])

#大的迴圈 要寫在前面

def find\_vowels(lst):

# return [c for c in term for term in lst] #error: term is not defined

return [''.join(c for term in lst for c in term if c in 'aeiou' or term[-1] in 'y')]

#沒有切割出來

print(find\_vowels(['w3resource', 'Python', 'Java', 'C++']))

#Textbook

def test(strs):

return ["".join(c for c in text if c.lower() in "aeiou") + (text[-1] if text[-1].lower() == "y" else "")

for text in strs]

strs = ["w3resource", "Python", "Java", "C++"]

print("Original List of strings:",strs)

print("Vowels from each of the original texts (y counts as a vowel at the end of the word:")

print(test(strs))

strs = ["ably", "abruptly", "abecedary", "apparently", "acknowledgedly"]

print("\nOriginal List of strings:",strs)

print("Positions of all uppercase vowels (not counting Y) in even indices:")

print(test(strs))

'''

87. Write a Python program to find a valid substring of a given string

that contains matching brackets, at least one of which is nested.

Input:

]][][[]]]

Output:

[[]]

Input:

]]]]]]]]]]]]]]]]][][][][]]]]]]]]]]][[[][[][[[[[][][][]][[[[[[[[[[[[[[[[[[

Output:

[[][][][]]

'''

#warmup

str1=']][][[]]]'

print(len(str1))

print(str1.find('[]',0,9)) #2

print(str1.find('[]',2+2,9)) #5

print(str1.find('[]',5+2,9)) #-1 (找不到了)

lst1=[1]

lst1.append(2)

print(lst1)

def mb(strs):

i=0

inner\_pos=[]

lgth=len(strs)

while i<lgth:

found=strs.find('[]',i,lgth)

# print(found)

inner\_pos.append((found,found+1))

i+=2

print(inner\_pos)

str1=''

for (j,k) in inner\_pos:

# print(j,k)

if strs[j-1]=='[' and strs[k+1]==']':

str1+=strs[j-1]+strs[j]+strs[k]+strs[k+1]+','

return str1

print(mb(']][][[]]]'))

print(mb(']]]]]]]]]]]]]]]]][][][][]]]]]]]]]]][[[][[][[[[[][][][]][[[[[[[[[[[[[[[[[['))

#只能找出單獨match bracket,無法找出nested

#Textbook

def test(s):

import re

return re.search(r"\[(\[\])+\]", s).group(0)

brackets = "]]]]]]]]]]]]]]]]][][][][]]]]]]]]]]][[[][[][[[[[][][][]][[[[[[[[[[[[[[[[[["

print("\nOriginal List of strings:",brackets)

print("\nFind a valid substring of the said string that contains matching brackets, at least one of which is nested:")

print(test(brackets))

#Special topics -Regex

import re

print(re.search('[Ss]teeler[s]','Steelers eliminated from playoffs,sucks steelers'))

print(re.search('[^0-9]','6ix9ine song is great')) #(1,2) i

print(re.search('[Ss]teeler.','Connor become a Steeler, Go Steelers!'))

# .是只要不是\newline的single character都算, (16,24) Steeler,

print(re.search('\w', '@psc.com.tw')) #(1,2) p

print(re.search(r'\bSteelers\b','I am a Steelers fan, how about you?'))

# \b: word boundary (等同word的category界定, [a-zA-Z0-9\_])

# 要記的用raw text

for i in range(1, 6):

s = f"x{'-' \* i}x"

print(f'{i} {s:10}', re.search('x-{2,4}x', s))

#repeating '-' 2,3,4

'''

88. Write a Python program to find an integer (n >= 0) with the

given number of even and odd digits

Input:

Number of even digits: 2 ,Number of odd digits: 3

Output:

22333

Input:

Number of even digits: 4 ,Number of odd digits: 7

Output:

22223333333

'''

def noeaod(ed,od):

return int(str(2)\*ed +str(3)\*od)

print(noeaod(3,3))

print(noeaod(4,7))

'''

89. Write a Python program to find all integers <= 1000 that are

the product of exactly three primes. Each integer should represent as

the list of its three prime factors

Input: 10

Output:

[[2, 2, 2]]

Input: 50

Output:

[[2, 2, 2], [2, 2, 3], [2, 2, 5], [2, 2, 7], [2, 2, 11], [2, 3, 2], [2, 3, 3], [2, 3, 5], [2, 3, 7], [2, 5, 2], [2, 5, 3], [2, 5, 5], [2, 7, 2], [2, 7, 3], [2, 11, 2], [3, 2, 2], [3, 2, 3], [3, 2, 5], [3, 2, 7], [3, 3, 2], [3, 3, 3], [3, 3, 5], [3, 5, 2], [3, 5, 3], [3, 7, 2], [5, 2, 2], [5, 2, 3], [5, 2, 5], [5, 3, 2], [5, 3, 3], [5, 5, 2], [7, 2, 2], [7, 2, 3], [7, 3, 2], [11, 2, 2]]

'''

def is\_prime(m):

if m>0:

return all(m%i for i in range(2,m-1))

def po3p(n):

result=[]

for i in range(2,n):

for j in range(2,n):

for k in range(2,n):

if i\*j\*k<=n and is\_prime(i) and is\_prime(j) and is\_prime(k):

result.append([i,j,k])

return result

print(po3p(10))

print(po3p(50))

print(po3p(500))

#Textbook

def test(n):

ps = [p for p in range(2,n) if all(p % sat != 0 for sat in range(2, p))]

return [[p, q, r] for p in ps for q in ps for r in ps if p\*q\*r <= n]

'''

90. For each triple of eaten, need, stock write a Python program

to get a pair of total appetite and remaining.

(appetite= eaten + need ; remaining= stock - need)

Input:

[[2, 5, 6], [3, 9, 22]]

Output:

[[7, 1], [12, 13]]

Input:

[[2, 3, 18], [4, 9, 2], [2, 5, 7], [3, 8, 12], [4, 9, 106]]

Output:

[[5, 15], [6, 0], [7, 2], [11, 4], [13, 97]]

Input:

[[1, 2, 3], [4, 5, 6]]

Output:

[[3, 1], [9, 1]]

'''

def taar(lsts):

#return [[lst[0]+lst[1],lst[2]-lst[1]] if lst[2]>=lst[1] for lst in lsts else \ [lst[0]+lst[2],0] if lst[2]<lst[1] for lst in lsts

result=[]

for lst in lsts:

if lst[2]>lst[1]:

result.append([lst[0]+lst[1],lst[2]-lst[1]])

else:

result.append([lst[0]+lst[2],0])

return result

print(taar([[2, 5, 6], [3, 9, 22]]))

print(taar([[2, 3, 18], [4, 9, 2], [2, 5, 7], [3, 8, 12], [4, 9, 106]]))

print(taar([[1, 2, 3], [4, 5, 6]]))

#Textbook

def test(nums):

return [[a+min(b, c), max(0, c-b)] for a, b, c in nums]

nums = [[2, 5, 6], [3, 9, 22]]

print("Original list (triple) of lists:")

print(nums)

print("Each triple of eaten, need, stock return a pair of total appetite and remaining:")

print(test(nums))

'''

91. Write a Python program to find all n-digit integers that start

or end with 2

Input: 1

Output:

[2]

Input: 2

Output:

[12, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 32, 42, 52, 62, 72, 82, 92]

Input: 3

Output:

[102, 112, 122, 132, 142, 152, 162, 172, 182, 192, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 302, 312, 322, 332, 342, 352, 362, 372, 382, 392, 402, 412, 422, 432, 442, 452, 462, 472, 482, 492, 502, 512, 522, 532, 542, 552, 562, 572, 582, 592, 602, 612, 622, 632, 642, 652, 662, 672, 682, 692, 702, 712, 722, 732, 742, 752, 762, 772, 782, 792, 802, 812, 822, 832, 842, 852, 862, 872, 882, 892, 902, 912, 922, 932, 942, 952, 962, 972, 982, 992]

'''

def soew2(n):

return [i for i in range(10\*\*(n-1),10\*\*n) if any((str(i)[0]=='2',str(i)[-1]=='2'))]

print(soew2(2))

print(soew2(3))

#Textbook

def test(n):

ans = []

for i in range(10 \*\* (n - 1), 10 \*\* n):

assert len(str(i)) == n

if str(i).startswith("2") or str(i).endswith("2"):

ans.append(i)

return ans

'''

92. Write a Python program to start with a list of integers,

keep every other element in place and otherwise sort the list.

Input:

[2, 5, 6, 3, 1, 4, 34]

Output:

[1, 5, 2, 3, 6, 4, 34]

Input:

[8, 0, 7, 2, 9, 4, 1, 2, 8, 3]

Output:

[1, 0, 7, 2, 8, 4, 8, 2, 9, 3]

'''

#Textbook- odd index不動, even index做排序

def test(nums):

li = nums.copy()

for i in range(len(li)):

if i % 2 == 0:

for j in range(i+2, len(li), 2):

if li[j] < li[i]:

swap(li, i, j)

return li

def swap(li, i, j):

temp = li[i]

li[i] = li[j]

li[j] = temp

nums = [2, 5, 6, 3, 1, 4, 34]

print("Original list (triple) of lists:")

print(nums)

print("In the said list, keep every other element in place and otherwise sort the list.:")

print(test(nums))

nums = [8, 0, 7, 2, 9, 4, 1, 2, 8, 3]

print("\nOriginal list (triple) of lists:")

print(nums)

print("In the said list, keep every other element in place and otherwise sort the list.:")

print(test(nums))

'''

93. Write a Python program to find the closest palindrome

from a given string

Input:

cat

Output:

cac

Input:

madan

Output:

madam

Input:

radivider

Output:

radividar

Input:

madan

Output:

madam

Input:

abc

Output:

aba

Input:

racecbr

Output:

racecar

'''

def cp(strs):

#pair: (0,-1),(1,-2),(2,-3)...(n,-n-1)

#if len(strs)=odd, pair count=(len(strs)-1)/2

#if len(strs)=even, pair count=len(strs)/2

lsts=list(strs)

for pc in range(len(lsts)//2):

if lsts[pc]!=lsts[(-pc-1)]:

lsts[-pc-1]=lsts[pc]

return ''.join(lsts)

print(cp('racecbr'))

print(cp('cat'))

print(cp('madan'))

print(cp('radividar'))

print(cp('fuckyou'))

print(cp('palindrome'))

print(cp('roethlisberger'))

#no use

def rev(strs1):

return ''.join([strs1[i] for i in range(len(strs1)-1,-1,-1)])

# strs1='hellofuck'

# print(strs1[4])

# strs1[4]='j' #Error:str object not support 'item assignment'

# string is an immutable type !

# print(strs1)

#Textbook

def test(s):

odd = 0

for i, c in enumerate(s):

if c != s[~i]:

odd += 1

if odd % 2 == 1:

half = odd // 2

pal = "".join((s[i] if i < half else s[~i] for i in range(len(s))))

return pal

else:

half = odd // 2

pal = "".join((s[i] if i <= half else s[~i] for i in range(len(s))))

return pal

s = "cat"

print("Original string:",s)

print("Closest palindrome of the said string:")

print(test(s))

'''

94. Given a string consisting of whitespace and groups of

matched parentheses, write a Python program to split it into groups

of perfectly matched parentheses without any whitespace.

Input:

( ()) ((()()())) (()) ()

Output:

['(())', '((()()()))', '(())', '()']

Input:

() (( ( )() ( )) ) ( ())

Output:

['()', '((()()()))', '(())']

'''

import re

#warmup

str1='( ()) ((()()())) (()) ()'

print(re.search("\(+(\(\))+\)+\)+",str1))

#span=(6,16), match='((()()()))' first one

print(re.findall("(\(\))+",str1)) #會是個list 但很複雜

print(re.findall("[( ]+\(\)[) ]+",str1)) #['( ()) ', ' (()) ']

print(re.findall("[( ]+\(\)\*[) ]+",str1)) #['( ()) ', '((()', ' (()) ']

#not knowing how to express the middle part!

#Textbook

def test(combined):

ls = []

s2 = ""

for s in combined.replace(' ', ''):

s2 += s

if s2.count("(") == s2.count(")"):

ls.append(s2)

s2 = ""

return ls

combined = '( ()) ((()()())) (()) ()'

print("Parentheses string:")

print(combined)

print("Separate parentheses groups of the said string:")

print(test(combined))

combined = '() (( ( )() ( )) ) ( ())'

print("\nParentheses string:")

print(combined)

print("Separate parentheses groups of the said string:")

print(test(combined))

#網友

def test(strs: str) -> list[str]:

res, wrd, cnt = [], '', 0

for c in strs.replace(' ', ''):

wrd += c

if c == '(': cnt += 1

else: cnt -= 1

if cnt == 0:

res.append(wrd)

wrd = ''

return res

print(test('( ()) ((()()())) (()) ()'))

'''

95. Write a Python program to find a palindrome of a given length

containing a given string

Input: madam , 7

Output:

madaadam (7+1)/2=4 -> mada, 4\*2=8 ->madaadam

Input: madam , 6

Output:

maddam 6/2=3 -> mad -> maddam

Input: madam , 5

Output:

maaaam why 6?

Input: madam , 3

Output:

maam why 4?

Input: madam , 2

Output:

mm

Input: madam , 1

Output:

aa why 2?

'''

#不是很懂題目的邏輯!

#Textbook

def test(s, length):

s\_index = 0

length\_half = (length - (length % 2)) // 2

ans = ""

while len(ans) < length\_half:

ans += s[s\_index%len(s)]

s\_index += 1

if length % 2 == 1: #若desired lgth為odd則在中間加一個孤零字'a'

ans += "a"

return ans + ans[::-1]

s = 'madam'

length = 7

print("String and length of the palindrome:",s,",",length)

print("Palindrome of the said string and length:")

print(test(s, length))

s = 'madam'

length = 6

print("\nString and length of the palindrome:",s,",",length)

print("Palindrome of the said string and length:")

print(test(s, length))

length = 5

print("\nString and length of the palindrome:",s,",",length)

print("Palindrome of the said string and length:")

print(test(s, length))

length = 3

print("\nString and length of the palindrome:",s,",",length)

print("Palindrome of the said string and length:")

print(test(s, length))

length = 2

print("\nString and length of the palindrome:",s,",",length)

print("Palindrome of the said string and length:")

print(test(s, length))

length = 1

print("\nString and length of the palindrome:",s,",",length)

print("Palindrome of the said string and length:")

print(test(s, length))

'''

96. Write a Python program to get the single digits in numbers

sorted backwards and converted into English words.

Input:

[1, 3, 4, 5, 11]

Output:

['five', 'four', 'three', 'one']

Input:

[27, 3, 8, 5, 1, 31]

Output:

['eight', 'five', 'three', 'one']

'''

#warmup

dict1={1:'one',2:'two',3:'three'}

print(list([dict1[1],dict1[3]]))

print(list(map(lambda x:dict1[x],[1,3]))) #['one','three']

def sbcte(lst):

dict0={'1':'one','2':'two','3':'three','4':'four','5':'five','6':'six','7':'seven','8':'eight','9':'nine','0':'zero'}

# print([c for n in lst for c in str(n)]) #can split into single-digit

# print(sorted(set([c for n in lst for c in str(n)]),reverse=True))

# [1,3,4,5,11] => ['5', '4', '3', '1']

final1=sorted(set([c for n in lst for c in str(n)]),reverse=True)

final2=list(map(lambda x:dict0[x],final1))

return final2

print(sbcte([1, 3, 4, 5, 11]))

print(sbcte([27, 3, 8, 5, 1, 31]))

print(sbcte([2346,1990,6,23]))

#Textbook 原題意是get rid of double-digit numbers

def test(nums):

digits = {"zero": None,

"one": 1,

"two": 2,

"three": 3,

"four": 4,

"five": 5,

"six": 6,

"seven": 7,

"eight": 8,

"nine": 9}

digits\_backwards = {digits[k]: k for k in digits}

print(digits\_backwards)

#{None: 'zero', 1: 'one', 2: 'two',....

digits = [digits[s] for s in digits]

li = [digits[n] for n in nums if n in digits]

return [digits\_backwards[n] for n in sorted(li, reverse=True)]

nums = [1, 3, 4, 5, 11]

print("Original list of numbers:")

print(nums)

print("Return the single digits in nums sorted backwards and converted to English words:")

print(test(nums))

'''

97. Write a Python program to find the following strange sort of list

of numbers: the first element is the smallest,

the second is the largest of the remaining,

the third is the smallest of the remaining,

the fourth is the largest of the remaining, etc.

Input:

[1, 3, 4, 5, 11]

Output:

[1, 11, 3, 5, 4]

Input:

[27, 3, 8, 5, 1, 31]

Output:

[1, 31, 3, 27, 5, 8]

Input:

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

Output:

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

'''

#even index: smallest of remaining

#odd index: largest of remaining

def ssol(lst):

result=[]

while len(lst)>1:

result.append(min(lst))

result.append(max(lst))

lst.pop(lst.index(min(lst)))

lst.pop(lst.index(max(lst)))

if len(lst)==1:

result.append(lst[0])

return result

print(ssol([1, 3, 4, 5, 11]))

print(ssol([27, 3, 8, 5, 1, 31]))

print(ssol([1, 2, 7, 3, 4, 5, 6]))

#Textbook

def test(nums):

if len(nums) < 2:

return nums

result = []

for i in range(len(nums)//2):

result.append(min(nums))

nums.remove(min(nums))

result.append(max(nums))

nums.remove(max(nums))

if len(nums) > 0:

result.append(nums[0])

if len(result) < 2\*len(nums):

result.extend(nums[len(result) // 2 + 1:len(result) // 2 + 1 + len(nums) - len(result)])

return result

nums = [1, 3, 4, 5, 11]

print("Original list of numbers:")

print(nums)

print("Strange sort of list of said numbers:")

print(test(nums))

'''98. Given a string consisting of groups of matched

nested parentheses separated by parentheses, write a Python program

to compute the depth of each group

Input: (()) (()) () ((()()()))

Output:

[2, 2, 1, 3]

Input: () (()) () () () ()

Output:

[1, 2, 1, 1, 1, 1]

Input: (((((((()))))))) () (()) ((()()()))

Output:

[8, 1, 2, 3]

'''

def gomnp(strs):

chk=0

result=[]

maxs=0

final=[]

for p in strs:

if p=='(':

chk+=1

if chk>maxs: maxs=chk

elif p==')':

chk-=1

else:pass

if chk==0:

result.append(maxs)

maxs=0

while 0 in result:

result.remove(0)

return result

print(gomnp('(()) (()) () ((()()()))'))

print(gomnp('() (()) () () () ()'))

print(gomnp('(((((((()))))))) () (()) ((()()()))'))

print(gomnp('(())(()) () ((()()()))')) #still correct!

#Textbook

print()

def test(parens):

print(parens.split())

return [len(s.split(')')[0]) for s in parens.split()]

parentheses = '(()) (()) () ((()()())) '

print("Parentheses strings:",parentheses )

print("\nDepth of groups of matched nested parentheses separated by parentheses:")

print(test(parentheses))

#問題是 如果各組 沒有用space 引開,那就不會是正確的了

parentheses = '(())(()) () ((()()())) '

print(test(parentheses))

'''99. Write a Python program to find a string such that,

when three or more spaces are compacted to a '-' and one or two spaces

are replaced by underscores, leads to the target.

Input: Python-Exercises

Output:

Python Exercises

Input: Python\_Exercises

Output:

Python Exercises

Input: -Hello,\_world!\_\_This\_is-so-easy!-

Output:

Hello, world! This is so easy!

'''

def cts(strs):

str1=strs.replace('-',' ',-1)

str2=str1.replace('\_',' ',-1)

return str2

print(cts('Python-Exercises'))

print(cts('Python\_Exercises'))

print(cts('Input: -Hello,\_world!\_\_This\_is-so-easy!-'))

#Textbook

def test(strs):

return strs.replace("-", " " \* 3).replace("\_", " ")

strs = "-Hello,\_world!\_\_This\_is-so-easy!-"

print("\nOriginal strings:",strs)

print("Depth of groups of matched nested parentheses separated by parentheses:")

print(test(strs))

'''

100. Write a Python program to find four positive even integers

whose sum is a given integer

Input:

n = 100

Output:

[94, 2, 2, 2]

Input:

n = 1000

Output:

[994, 2, 2, 2]

Input:

n = 10000

Output:

[9994, 2, 2, 2]

Input:

n = 1234567890

Output:

[1234567884, 2, 2, 2]

'''

def fpei1k(n):

return [[i,j,k,l] for i in range(1,n) for j in range(1,n) for k in range(1,n) for l in range(1,n) if i+j+k+l==n and all((i%2==0,j%2==0,k%2==0,l%2==0))][0]

print(fpei1k(100))

# time consuming!

#Textbook

def test(n):

for a in range(n, 0, -1):

if not a % 2 == 0:

continue

for b in range(n - a, 0, -1):

if not b % 2 == 0:

continue

for c in range(n - b - a, 0, -1):

if not c % 2 == 0:

continue

for d in range(n - b - c - a, 0, -1):

if not d % 2 == 0:

continue

if a + b + c + d == n:

return [a, b, c, d]

#timeit

import timeit

code1='''

def fpei1k(n):

return [[i,j,k,l] for i in range(1,n) for j in range(1,n) for k in range(1,n) for l in range(1,n) if i+j+k+l==n and all((i%2==0,j%2==0,k%2==0,l%2==0))][0]

print(fpei1k(100))

'''

code2='''

def test(n):

for a in range(n, 0, -1):

if not a % 2 == 0:

continue

for b in range(n - a, 0, -1):

if not b % 2 == 0:

continue

for c in range(n - b - a, 0, -1):

if not c % 2 == 0:

continue

for d in range(n - b - c - a, 0, -1):

if not d % 2 == 0:

continue

if a + b + c + d == n:

return [a, b, c, d]

n = 100

print("Four positive even integers whose sum is",n)

print(test(n))

'''

print("my consuming time is %.2f secs" % timeit.timeit(code1,number=1))

print("Textbook consuming time is %.6f secs" % timeit.timeit(code2,number=1))

#5.29 sec vs. 0.000012 sec

---------(all exercises finished!)-------------------------------------------------------------------