Q.1 WAP to find a list of integers with exactly two occurrences of nineteen and at least three occurrences of five. Return True otherwise False.

→

```
def test(nums):
    return nums.count(19)==2 and nums.count(5)>=3
nums=[19,19,15,5,3,5,5,2]
print(nums)
print(test(nums))
nums=[19,15,15,5,3,5,5,2]
print(nums)
print(test(nums))
nums=[19,19,5,5,5,5,5]
print(nums)
print(test(nums))
```

Output:-

```
[19,19,15,5,3,5,5,2]
True
[19,15,15,5,3,5,5,2]
False
[19,19,5,5,5,5,5]
True
```

Q.2

→

```
def test(nums):
    return len(nums)==8 and nums.count(nums[4])==3
nums=[19,19,15,5,5,5,1,2]
print(nums)
```

```
print(test(nums))
nums = [19,15,5,7,5,5,2]
print(nums)
print(test(nums))
nums= [11,12,14,13,14,13,15,14]
print(nums)
print(test(nums))
nums= [19,15,11,7,5,6,2]
print(nums)
print(test(nums))
```

Output:-

```
[19,19,15,5,5,5,1,2]
True
[19,15,5,7,5,5,2]
False
[11,12,14,13,14,13,15,14]
True
[19,15,11,7,5,6,2]
False
```

Q.3 WAP that accepts an integer & determine wheather it is greater than 4^4 & which is 4 mod 34.

→

```
def test(n):
    return n%34 == 4 and n > 4**4

n = 922
print(n)
print(test(n))
```

```
n = 914
print(n)
print(test(n))
n = 854
print(test(n))
print(test(n))
```

Output:-

```
922
True
914
False
854
True
```

Q.4

→

```
def test(n):
    return [n+2*i for i in range(n)]
n = 2
print("Number of piles:", n)
print(test(n))
n = 10
print("Number of piles:", n)
print(test(n))
n = 3
print("Number of piles:", n)
print(test(n))
n = 17
print("Number of piles:", n)
print(test(n))
```

Output :-

Number of piles : 2

[2, 4]

Number of piles : 10

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

Number of piles : 3

[3, 5, 7]

Number of piles : 17

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

Q.5 WAP to check the n^{th-1} string is a proper substring of the n^{th} string in a given list of string

→

```
def test(str1):
  return str1[len(str1)-2] in str1[len(str1)-1] and
  str[len(str1)-2] != str1[len(str1)-1]

str1 = ["a", "abb", "sfs", "aa", "de", "sfde"]
print(str1)
print(test(str1))
str1 = ["a", 'abb', "sfs", 'a', "ee", "sfde"]
print(str1)
print(test(str1))
str1 = ['a', 'abb', 'sad', 'aaaa', 'esdfe', 'sfsdfde', 'sfsd', 'sfsdf', 'qw
print(str1)
print(test(str1))
str1['a', 'abb', 'sad', 'aaaa', 'esdfe', 'sfsdfde', 'sfsd', 'sfsdf', 'qwsfsdfr
print(str1)
print(test(str1))
```

Output:-
["a", "abb", "sfs", "oo", "de", "sfde"]
True
["a", "abb", "sfs", "oo", "ee", "sfde"]
False
["a", "abb", "sad", "aaaaesdfe", 'sfsdfde', 'sfsd', 'sfsdf', 'qwrew']
False
['a', 'abb', 'sad', 'aaaaesdfe', 'sfsdfde', 'sfsd', 'sfsdf', 'qwsfsdfre
True

Q.6 WAP to test a list of one hundred integers between 0 and 999, which all differ by ten from one another.

→

```
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(nums)
nums = list(range(0,1000,20))
print(nums)
print(test(nums))
```

Output:-
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120, ... 910
920, 930, 940, 950, 960, 970, 980, 990]
True
[0, 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, ... 900
920, 940, 960, 980]
False

7. WAP to check a given list of integer where the sum of the first i integers is i.

→

```
def test(li,i):
    return sum(li[:i])==i
nums=[0,1,2,3,4,5]
i=1
print(nums)
print(test(nums,1))
i=6
nums=[1,1,1,1,1,1]
print(nums)
print(test(nums,6))
i=2
nums=[2,2,2,2,2]
print(nums)
print(test(nums,2))
```

Output:-

```
[0,1,2,3,4,5]
False
[1,1,1,1,1,1]
True
[2,2,2,2,2]
False
```

Q.8 WAP to split a string of words separated by commas and space into two lists, words and separators.

→

```
def test(string):
    import re
    merged = re.split(r"([, ]+)", string)
    return [merged[::2], merged[1::2]]

s = "W3resource Python, Exercises."
print(s)
print(test(s))
s = "The dance, held in the school gym, ended at midnight"
print(s)
print(test(s))
s = "The colors in my studyroom are blue, green, and yellow"
print(s)
print(test(s))
```

Output:

```
W3resource Python, Exercises.
[['W3resource', 'Python', 'Exercises.'], [' ', ', ']]
The dance, held in the school gym, ended at midnight
[['The', 'dance', 'held', 'in', 'the', 'school', 'gym', 'ended', 'at',
 'midnight'], [' ', ', ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']]
The colors in my studyroom are blue, green, and yellow
[['The', 'dance', 'colors', 'in', 'my', 'studyroom', 'are', 'blue',
 'green', 'and', 'yellow'], [' ', ', ', ' ', ' ', ' ', ' ', ' ', ', ', ' ']]
```

Q.9 WAP to find a list of integers containing exactly four distinct values such that no integer repeats twice consecutively among the first twenty entries.

→

```
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(nums)
print(test(nums))
nums=['1','2','3','3','1','2','3','3','1','2','3','3','1','2','3','3']
print(nums)
print(test(nums))
nums=['1','2','3','1','2','3','1','2','3','1','2','3']
print(nums)
print(test(nums))
```

Output:-

```
['1','2','3','4','1','2','3','4','1','2',3,4,1,2,3,4]
True
[1,2,3,3,1,2,3,3,1,2,3,3,1,2,3,3]
False
[1,2,3,1,2,3,1,2,3,1,2,3]
False
```

Q.10 Given a string consisting of whitspace and groups of matched parenthese, write a python program to split it into groups of perfectly matched parenthese without any whitespace.

→

```
def test(Combined):
    Is = []
    s2 = ""
    for s in Combined.replace(' ', ''):
        s2 += s
        if s2.count("(") == s2.count(")"):
            Is.append(s2)
            s2 = ""
    return Is

Combined = '( ()) ((()()())) (()) ()'
print(Combined)
print(test(Combined))
Combined = '() ((()()( ))) (())'
print(combined)
print(test(combined))
```

```
(()) ((()()())) (()) ()
['(())', '((()()()))', '(())', '()']
() (((()() ( ))) (())
['()', '(((()() ( )))', '(())']
```

PAGE:
DATE: / /

Q.11 WAP to find the indexes of numbers in a given list below a given threshold.

→

```
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 li
      below the given threshold:")
print(test(nums, thresh))
nums = [0, 12, 4, 3, 49, 9, 1, 5, 3]
thresh = 10
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))
```

Output:-

```
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]
```

Original list:
[0, 12, 45, 3, 4923, 322, 105, 29, 15, 39, 55]
Threshold : 100
Check the index of numbers of the said list below the given threshold:
[0, 1, 2, 3, 7, 8, 9, 10]

Q.12 WAP to check wheather the given strings are palindromes or not

→

```
def test(strs):
    return [s==s[::-1] for s in strs]
strs['palindrome', 'madamimadam', '', 'foo', 'eyes']
print(strs)
print(test(strs))
```

Output:
['palindrome', 'madamimadam', '', 'foo', 'eyes']
[False, True, True, False, False]

Q.13 WAP to find strings in a given list starting with a given prefix

→

```
def test(str, prefix):
    return [s for s in strs if s.startswith(prefix)]
strs = ['Cat', 'Car', 'Fear', 'Center']
prefix = "Co"
print(str)
print(test(strs, prefix))
```

```
strs = ['cat', 'dog', 'shatter', 'donut', 'at', 'todo']
prefix = "do"
print(strs)
print(test(strs, prefix))
```

Output:-

```
['cat', 'car', 'fear', 'center']
['cat', 'car']
['cat', 'dog', 'shatter', 'donut', 'at', 'todo']
['dog', 'donut']
```

Q.14 WAP to find the length of a given list of non-empty strings.

→

```
def test(str):
    return [*map(len, strs)]
strs = ['cat', 'car', 'fear', 'center']
print(strs)
print(test(strs))
strs = ['cat', 'dog', 'shatter', 'donut', 'at', 'todo']
print(strs)
print(test(strs))
```

Output:-

```
['cat', 'car', 'fear', 'center']
[3, 3, 4, 6]
['cat', 'dog', 'shatter', 'donut', 'at', 'todo']
[3, 3, 7, 5, 2, 4, 0]
```

Q.15 WAP to Find the longest string in a given list of strings

→

```
def test(words):
    return max(words, key=len)
strs = ['cat', 'car', 'fear', 'center']
print(strs)
print(test(strs))
strs=['cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']
print(strs)
print(test(strs))
```

Output :-

```
['cat', 'car', 'fear', 'center']
Center
['cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']
shatter
```

Q.16 WAP to find strings in a given list containing a given substring.

→

```
def test(strs, substr):
    return[s for s in strs if substr in s]

strs = [[('cat', 'car', 'fear', 'center')]]
print(strs)
substrs = 'ca'
print(test(strs, substrs))
strs = [[('cat', 'dog', 'shatter', 'donut', 'at', 'todo', '')]]
print(strs)
```

```
substrs = 'o'
print(test(strs, substrs))
strs=['Cat', 'dog', 'shatter', 'donut', 'at', 'todo', '']]
print(strs)
substrs = 'oe'
print(test(strs, substrs))
```

Output:-

```
[(o, ('Cat', 'Car', 'Fear', 'Center'))]
['Cat', 'Car']
[(o, ('Cat', 'dog', 'shatter', 'donut', 'at', 'todo', ''))]
['dog', 'donut', 'todo']
[(oe, ('Cat', 'dog', 'shatter', 'donut', 'at', 'todo', ''))]
[]
```

Q.17 WAP to create a string consisting of non-negative integers up to n inclusive

→

```
def test(n):
    return ''.join(map(str, range(n+1)))
n=4
print(n)
print(test(n))
n=15
print(n)
print(test(n))
```

Output:-

```
4
0 1 2 3 4
```

15

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

Q.18

→

```
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(M)
print(test(M,T))
print("\n")
M=[[1,2,3,2],[],[7,9,2,1,4]]
T=2
print(M)
print(test(M,T))
```

Output :-

```
[[1,3,2,32,19],[19,2,48,19],[],[9,35,4],[3,19]]
[[0,4],[1,0],[1,3],[4,1]]

[[1,2,3,2],[],[7,9,2,1,4]]
[[0,1],[0,3],[2,2]]
```

Q.19

→
```
def test(s):
    if " " in s:
        return s.split(" ")
    if "," in s:
        return s.split(",")
    return [c for c in s if c.islower() and ord(c)%2==
```

```
strs = "a b c d"
print(strs)
print(test(strs))
strs = "a,b,c,d"
print(strs)
print("abcd")
print(strs)
print(test(strs))
```

Output:-

```
a b c d
['a', 'b', 'c', 'd']
a,b,c,d
['a', 'b', 'c', 'd']
abcd
['b', 'd']
```

Q.20 WAP to determine the direction ('increasing' or 'decreasing') of monotonic sequence numbers.

→

```
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(nums)
print( test(nums)
nums = [6,5,4,3,2,1]
print(nums)
print( test(nums)
nums = [19,19,5,5,5,5,5]
print(nums)
print(test(nums))
```

Output:

```
[1,2,3,4,5,6]
Increasing
[6,5,4,3,2,1]
Decreasing
[19,19,5,5,5,5,5]
Not a monotonic sequence!
```

PAGE:
DATE: / /

Q.21 WAP to check, for each string in a given list, wheather the last character is an isolated letter or not.

→

```
def test(strs):
    return[ len(s.split(" ")[-1])==1 for s in strs]
strs = ['cat', 'car', 'fear', 'center']
print(strs)
print(test(strs))
strs['ca t', 'car', 'fea r', 'cente r']
print(strs)
print(test(strs))
```

Output:-

```
['cat', 'car', 'fear', 'center']
['false', 'false', 'false', 'false']
['ca t', 'car', 'fea r', 'cente r']
['True', 'false', True, True]
```

Q.22 WAP to Compute the sum of the ASCII values of the upper case chacters in a given string

→

```
def test(strs):
    return sum(map(ord, filter(str.isupper, strs)))

strs = "PytHon ExerciSEs"
print(strs)
print(test(strs))
strs = "JavaScript"
```

```
print(strs)
print(test(strs))
```

Output:

```
Python Exercises
373
JavaScript
157
```

Q.23 WAP to find the indices at which the number in the list drop.

→

```
def test(nums):
    drop_indices = []
    for i in range(1,len(nums)):
        if nums[i] < nums[i-1]:
            drop_indices.append(i)
    return drop_indices
nums = [0,-1,3,8,5,9,8,14,2,4,3,-10,10,17,41,22,-4,-4,-15,0]
print(nums)
print(test(nums),'\n')
nums = [6,5,4,3,2,1]
print(nums)
print(test(nums),'\n')
nums = [1,19,5,15,5,25,5]
print(nums)
print(test(nums))
```

Output:-

[0, -1, 3, 8, 5, 9, 8, 14, 2, 4, 3, -10, 10, 17, 41, 22, -4, -4, -15, -0]
[1, 4, 6, 8, 10, 11, 15, 16, 18]

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

[1, 19, 5, 15, 5, 25, 5]
[0, 2, 4, 6]

Q.24 WAP to create a list whose ith element is the maximum of the first i elements from an input list.

→

```
def test(nums):
    return [max(num[:i]) for r in range(1, len(nums)
nums = [0, -1, 3, 8, 5, 9, 8, 14, 2, 4, 3, -10, 10, 17, 41, 22, -4, -15, 0
print(nums)
print(test(nums))
nums = [6, 5, 4, 3, 2, 1]
print(nums)
print(test(nums))
```

Output:-

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

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

Q.25 WAP to find the XOR of two given strings interpreted as binary numbers.

→

```
def test(nums):
    return bin(int(nums[0],2)^int(nums[1],2))
nums = ["0001","1011"]
print(nums)
print(test(nums))

nums=["1000111011000001","1001011001011110"]
print(nums)
print(test(nums))
```

Output:-

```
['0001','1011']
0b1010
['1000111011000001','1001011001011110']
0b1100010011111
```

Q.26 WAP to find the largest number where Commas or periods are decimal points.

```
def test(str_nums):
    return max(float(s.replace(",",".")) for s in str_nums)
str_nums = ["100","102,1","101.1"]
print(str_nums)
print(test(str_nums))
```

Output:- ["100","102,1","101.1"]

101.1

Q.27 WAP to find x that minimizes the mean squared deviation from a given list of no.

→

```
def test(nums):
    return sum(nums) / len(nums)
nums = [4,-5,17,-9,14,108,-9]
print(nums)
print(test(num))
nums = [12,-2,14,3,-15,10,-45,3,30]
print(nums)
print(test(nums))
```

Output:

```
[4,-5,17,-9,14,108,-9]
17.142857142857142
[12,-2,14,3,-15,10,-45,3,30]
1.1111111111111112
```

Q.28 WAP to select a string from a given list of strings with the most unique characters.

→

```
def test(strs):
    return max(strs, key = lambd x:len(set(x))
strs = ['Cat','Catatatatctsa','abcdefghijklmnopa','1234259
      39185125','','foo','unique']
print(strs)
print(test(strs))
strs = ['Green','Red','orange','Yellow','white']
print(strs)
print(test(strs))
```

Output:-
['Cat', 'abcdefghijklmnop', '12425923918512S', '', 'Foo', 'unique
abcdefghijklmnop
['Green', 'Red', 'orange', 'Yellow', 'white']
orange

Q.29 WAP to find the indices of two number that sum to 0 in a given list of no.

→

```
def test(nums):
    s = set(nums)
    for i in s:
        if -i in s:
            return[nums.index(i),nums.index(-i)]
nums = [1,-4,6,7,4]
print(nums)
print(test(nums))
nums = [1232, -20352, 12547, 12440, 741, 341, 525, 20352, 91, 20]
print(nums)
print(test(nums))
```

Output:-
[1,-4,6,7,4]
[4,1]
[1232, -20352, 12547, 12440, 741, 341, 525, 20352, 91, 20]
[1,7]

Q.30 WAP to find a list of strings that have fewer total characters.

→

```
def test(strs):
    return min(strs,key=lambda x:sum(len(i) for i in x))
strs=[['this','list','is','narrow'],['I','am','shorter
    but wider']]
print(strs)
print(test(strs))
strs = [['Red','Black','Pink'],['Green','Red','white']]
print(strs)
print(test(strs))
```

Output:-

```
[['this','list','is','narrow'],['I','am','shorter but wi
['this','list','is','narrow']
[['Red','Black','Pink'],['Green','Red','white']]
['Red','Black','Pink']
```

Q.31 WAP to find coordinates of a triangle with given side length

→

```
def test(sides):
    a,b,c=sorted(sides)
    s=sum(sides)/2
    area=(s*(s-a)*(s-b)*(s-c))**0.5
    y=2*area/a
    x=(c*2-y*2)**0.5
    return [[0.0,0.0],[a,0.0],[x,y]]
```

```
sides = [3,4,5]
print(sides)
print(test(sides))
sides = [5,6,7]
print(sides)
print(test(sides))
```

Output:

```
[3,4,5]
[[0.0, 0.0], [3, 0.0], [3.0, 4.0]]
[5,6,7]
[[0.0, 0.0], [5, 0.0], [3.8, 5.878775382679628]]
```

Q.32 WAP to rescale and shift number in a given list, so that they cover the range [0,1]

→

```
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(nums)
print(test(nums))
nums = [13.0, 17.0, 17.0, 15.5, 2.94]
print(nums)
print(test(nums))
```

Output:-
[18.5, 17.0, 18.0, 19.0, 18.0]
[0.75, 0.0, 0.5, 1.0, 0.5]
[13.0, 17.0, 17.0, 15.5, 2.94]
[0.7155049786628734, 1.0, 1.0, 0.8933143669985776, 0.0]

Q.33 WAP to find the position of all uppercase vowels in even indices of given string.

→

```
def test(strs):
    return [i for i, c in enumerate(strs) if i%2==0 and
            c in "AEIOU"]
strs = "W3rEsOUrcE"
print(strs)
print(test(strs), "\n")
strs = "AEIOUYW"
print(strs)
print(test(strs))
```

Output:-
W3rEsOUrcE
[6]
AEIOUYW
[0,2,4]

Q.34 WAP to find the sum of the numbers in given list among the first k with more than 2 digits.

→

```
def test(nums, k):
    s=0
    for i in range(len(nums))[:k]:
        if len(str(abs(num[i]))) > 2:
            s = s + nums[i]
    return s
nums = [4,5,17,9,14,108,-9,12,76]
print(nums)
k=4
print("Value of k:", k)
print(test(nums, k))
nums = [4,5,17,9,14,108,-9,12,76]
k=6
print("Value of k:", k)
print(test(nums,k))
nums = [114,215,-117,119,14,108,-9,12,76]
print("\n", nums)
k=5
print("Value of k:", k)
print(test(nums,k))
print("\n", nums)
k=1
print("Value of k:", k)
print(test(nums, k))
```

Output:

[4, 5, 17, 9, 14, 108, -9, 12, 76]
value of k: 4
0

value of k: 6
108

[114, 215, -117, 119, 14, 108, -9, 12, 76]
value of k: 5
331

[114, 215, -117, 119, 14, 108, -9, 12, 76]
value of k: 1
114

Q.35 WAP to Compute the product of the odd digits in a given number, or 0 if there aren't any

→

```
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
n = 123456789
print(n)
print(test(n),'\n')
n = 2468
```

```
print(n)
print(test(n),'\n')
n = 13579
print(n)
print(test(n),'\n')
```

Output :-

```
123456789
945
248
0
13579
945
```

Q.36 WAP to find the largest k numbers from a given list of numbers.

→

```
def test(nums, k):
  if K==0:
    return []
  return sorted(nums, reverse = True)[:k]
nums = [1,2,3,4,5,5,3,6,2]
print(nums)
K=1
print("K:", k)
print(test(nums, K)
print("\n", nums)
K=2
print("K:", k)
print(test(nums, k)
```

```
print("\n", nums)
k=3
print("k:", k)
print(test(nums,k))
print('\n', nums)
k=4
print("k:", k)
print(test(nums, k))
print k=5
print("k:", k)
print(test(nums, k))
```

Output:-

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

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

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

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

K:5
[6,5,5,4,3]

Q.37 WAP to find the largest integer divisor of a number n that is less than n.

→

```
def test(n):
    return next(d for d in range(n-1,0,-1) if n%d==0)
n=18
print(n)
print(test(n))
n=100
print(n)
print(test(n))
n=102
print(n)
print(test(n))
n=500
print(n)
print(test(n))
n=1000
print(n)
print(test(n))
n=6500
print(n)
print(test(n))
```

Output:-

18

9

100

50

102

51

500

250

1000

500

6500

3250

Q.38 WAP to sort the numbers in a given list by the sum of their digits

→

```
def test(nums):
    return sorted(nums, key = lambda n: sum(int(c)
    for c in str(n) if c != "-"))
nums = [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
print(nums)
print(test(nums))
nums = [23, 2, 9, 34, 8, 9, 10, 74]
print(nums)
print(test(nums))
```

Output

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

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

Q.39 WAP to determine which triples sum to zero From a given list of lists

→

```
def test(nums):
    return [sum(t) == 0 for t in nums]
nums = [[1343532, -2920635, 332], [-27, 18, 9], [4, 0, -4], [2, 2, 2],
        [-20, 16, 4]]
print(nums)
print(test(nums))
nums = [[1, 2, -3], [-4, 0, 4], [0, 1, -5], [1, 1, 1], [-2, 4, -1]]
print("\n", nums)
print(test(nums))
```

Output:

[[1343532, -2920635, 332], [-27, 18, 9], [4, 0, -4], [2, 2, 2], [-20, 16, 4]]
[False, True, True, False, True]

[[1, 2, -3], [-4, 0, 4], [0, 1, -5], [1, 1, 1], [-2, 4, -1]]
[True, True, False, False, False]

PAGE:
DATE: / /

Q.40 WAP to find strings that, when case is flipped, given a target where vowels are replaced by character two later

↳

```
def test(str):
    return str.translate({ord(c):ord(c)+2 for c in
                          "aeiouAEIOU"}).swapcase()
strs="Python"
print(strs)
print(test(strs))
strs="aeiou"
print('\n',strs)
print(test(strs))
strs="Hello world!"
print('\n',strs)
print(test(strs))
strs="AEIOU"
print('\n',strs)
print(test(strs))
```

Output:-

```
Python
pYTHON

aeiou
CGKQW

Hello world!
hGLLQ. WQRLD!
```

AEIOU
cgkqw

Q.41 WAP to sort number based on strings.

→

```
def test(strs):
    return ' '.join([x for x in 'one two three four five six
    seven eight nine'.split() if x in strs])
strs = "six one four one two three"
print(strs)
print(test(strs))
strs = "six one four three two nine eight"
print("\n", strs)
print(test(strs))
strs = "nine, eight seven six five four three two one"
print("\n", strs)
print(test(strs))
```

Output:

six one four three one two
one two three four six

six one four three two nine eight
one two three four six eight nine

nine eight seven six five four three two one
one two three four five six seven eight nine

PAGE:
DATE: / /

Q. 42 Write AP to find the set of distinct characters in a given string, ignoring case.

→

```
def test(strs):
    return [*set(strs.lower())]
strs = "HELLO"
print(strs)
print(test(strs))
strs = 'Hello'
print("\n", strs)
print(test(strs))
strs = " Ignoring Case'
print("\n", strs)
print(test(strs))
```

Output:-

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

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

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

Q.43 WAP to find all words in a given string with n consonants.

→

```
def test(strs, n)
  return [w for w in strs.split() if sum([c not in "aeiou"
                                  for c in w.lower()]) == n]
strs = "this is our time"
print (strs)
n=3
print("Number of Consonants:", n)
print (test (strs, n))
n=2
print("Number of Consonant:", n)
print (test (strs, n))
n=1
print("\n Number of Consonant:", n)
print (test (strs, n))
```

Output:-

```
this is our time
Number of Consonant: 3
['this']

Number of Consonant: 2
['time']

Number of Consonant: 1
['is', 'our']
```

Q.44 WAP to find which characters of a a hexadecimal number correspond to prime numbers.

→

```
def test(hn):
    return [c in "2357BD" for c in hn]
hn = "123ABCD"
print(hn)
print(test(hn))
hn = "123456"
print(hn)
print(test(hn))
hn = "FACE"
print(hn)
print(test(hn))
```

Output:-

```
123ABCD
[False, True, True, False, True, False, True]

123456
[False, True, True, False, True, False]

FACE
[False, False, False, False]
```

Q.45 WAP to find all even palindrome up to n.

→

```
def test(n):
    return [i for i in range(0,n,2) if str(i)==str(i)[::-1]]
n=50
print("\nEven palindromes up to",n,"-")
print(test(n))
n=100
print("\nEven palindromes up to",n,"-")
print(test(n))
n=500
print("\nEven palindromes up to",n,"-")
print(test(n))
n=2000
print("\nEven palindromes up to",n,"-")
print(test(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 palindrome up to 2000:
[—11— , 606, 616, 626, 636, 646, 656, 666, 676, 686, 696, 808, 818, 828, 838, 848, 858, 868, 878, 888, 898]

Q.46

→
```
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[]
nums=[1,9,4,6,10,11,14,8]
print(nums)
print(test(nums))
nums=[1,7,4,4,9,2]
print(nums)
print(test(nums))
nums=[1,7,7,5,9]
print('\n',nums)
print(test(nums))
```

Output:-
[1,9,4,6,10,11,14,8)
[4,2]

[1,7,44,9,2]
[2,5]

[1,7,7,5,9]
[]

Q.47

```
def test(nums):
    return[n for n in nums if int(str(n)[:2]) +
    sum(map(int,(n)[2:])) > 0]
nums=[11,-6,-103,-200
print(nums)
print(test(nums))
nums=[1,7,-4,4,-9,2]
print("\n",nums)
print(test(nums))
nums=[10,-11,-71,-31,14,-32]
print(nums)
print(test(nums))
```

Output:

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

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

[10,-11,-71,-33,14,-32]
[10,-13,14]
```

PAGE:
DATE: / /

Q.48

→
```
def test(nums):
    For i in range(len(nums)-1):
        if nums[i] >= nums[i+1]:
            return [i, i+1]
    return[]
nums = [1,2,3,0,4,5,6]
print(nums)
print(test(nums))
nums = [1,2,3,4,5,6]
print(nums)
print(test(nums))
nums = [1,2,3,8,4,6,5,7]
print('\n', nums)
print(test(nums))
nums = [-3,-2,-3,0,2,3,4]
print('\n', nums)
print(test(nums))
```

Output::

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

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

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

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

Q. 49
→
```
def test(nums):
    return max([-1]+[i for i in nums if i>0 and
                    nums.count(i)>=i])
nums=[1,2,2,3,4,4,4,4]
print(nums)
print(test(nums))
nums=[3,1,4,17,5,17,2,1,41,32,2,5,5,5,5]
print('\n',nums)
print(test(nums))
nums=[1,2,2,3,4,5,6]
print('\n',nums)
print(test(nums))
```

Output:-
[1, 2, 2, 3, 4, 4, 4, 4]
4

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

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

Q. 50

→
```
def test(words):
    return sorted([w for w in words if len(w)%2=
                   key=lambda w: (len(w), w))
words=['Red','Black','white','Green','Pink','orange']
print(words)
print(test(words))
words=['The','worm','ate','a','bird','imagine','that','!','Absu
print('\n',words)
print(test(words))
```

Output :-

```
['Red','Black','white','Green','Pink','orange']
['Pink','orange']

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