Question 1

1) A number is said to be Harshad if it's exactly divisible by the sum of its digits. Create a function that determines whether a number is a Harshad or not. Examples is_harshad(75) ➞ False 7 + 5 = 12 75 is not exactly divisible by 12 is_harshad(171) ➞ True 1 + 7 + 1 = 9 9 exactly divides 171 is_harshad(481) ➞ True is_harshad(89) ➞ False is_harshad(516) ➞ True is_harshad(200) ➞ True

In [1]:
```
def is_harshad(n):

    num = 0
    n1 = n
    while n1!=0:
        temp = n1%10
        num+=temp

        n1 = n1//10
    if(n%num==0):
        return True
    return False
```

In [3]:
```
n = int(input("Enter a number: "))
if(is_harshad(n)):
    print(f"{n} is a harshad number")
else:
    print(f"{n} is not a harshad number")
```

```
Enter a number: 999
999 is a harshad number
```

In [ ]:
```
```

Question 2

2) A fruit juice company tags their fruit juices by concatenating the first three letters of the words in a flavor's name, with its capacity. Create a function that creates product IDs for different fruit juices. Examples get_drink_ID("apple", "500ml") ➞ "APP500" get_drink_ID("pineapple", "45ml") ➞ "PIN45" get_drink_ID("passion fruit", "750ml") ➞ "PASFRU750"

In [4]:
```
def get_drink_ID(fruit, quantity):

    l = fruit.split(' ')
    fId = ''
    for i in l:
        fId += i[:3].upper()

    return fId+quantity[:-2]
```

In [5]:
```
print(get_drink_ID("apple", "500ml"))
print(get_drink_ID("pineapple", "45ml"))
print(get_drink_ID("passion fruit", "750ml"))
```

```
APP500
PIN45
PASFRU750
```

In [ ]:

Question 3

3) Write a function that takes a list of numbers and returns a list with two elements: 1. The first element should be the sum of all even numbers in the list. 2. The second element should be the sum of all odd numbers in the list. Example sum_odd_and_even([1, 2, 3, 4, 5, 6]) ➞ [12, 9] # 2 + 4 + 6 = 12 and 1 + 3 + 5 = 9 sum_odd_and_even([-1, -2, -3, -4, -5, -6]) ➞ [-12, -9]) sum_odd_and_even([0, 0]) ➞ [0, 0]

In [6]:
```
def sum_odd_and_even(data):

    even = 0
    odd = 0
    for i in data:

        if i%2==0:
            even+=i
        else:
            odd+=i

    return [even, odd]
```

In [7]:
```
print(sum_odd_and_even([1, 2, 3, 4, 5, 6]))
print(sum_odd_and_even([-1, -2, -3, -4, -5, -6]))
print(sum_odd_and_even([0, 0]))
print(sum_odd_and_even([i for i in range(10,31)]))
print(sum_odd_and_even([i for i in range(20,51) if i%3==0]))
```

```
[12, 9]
[-12, -9]
[0, 0]
[220, 200]
[180, 165]
```

In [ ]:

Question 4

Q4) Create a function that takes a list of positive and negative numbers. Return a list where the first element is the count of positive numbers and the second element is the sum of negative numbers. Examples sum_neg([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -11, -12, -13, -14, -15]) ➞ [10, -65] # There are a total of 10 positive numbers. # The sum of all negative numbers equals -65. sum_neg([92, 6, 73, -77, 81, -90, 99, 8, -85, 34]) ➞ [7, -252] sum_neg([91, -4, 80, -73, -28]) ➞ [2, -105]

In [8]:
```
def sum_neg(data):

    count = 0
    total = 0

    for i in data:

        if(i>0):
            count+=1
        else:
            total+=i
```

```
    return [count,total]
```

In [9]:
```
print(sum_neg([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -11, -12, -13, -14, -15]) )
print(sum_neg([92, 6, 73, -77, 81, -90, 99, 8, -85, 34]))
print(sum_neg([91, -4, 80, -73, -28]))
print(sum_neg([]))
print(sum_neg([i for i in range(-10,11)]))
```

```
[10, -65]
[7, -252]
[2, -105]
[0, 0]
[10, -55]
```

In [ ]:
```

```

Question 5

Q5) Create a function that returns the frequency distribution of a list. This function should return an object, where the keys are the unique elements and the values are the frequency in which those elements occur. Examples get_frequencies(["A", "B", "A", "A", "A"]) ➞ { "A" : 4, "B" : 1 } get_frequencies([1, 2, 3, 3, 2]) ➞ { 1: 1, 2: 2, 3: 2 } get_frequencies([True, False, True, False, False]) ➞ { True: 2, False: 3 } get_frequencies([]) ➞ {}

In [10]:
```
def get_frequencies(data):

    d = {}

    for i in data:

        if i in d.keys():     #Method 1

            d[i]+=1
        else:
            d[i]=1

#         d[i] = data.count(i)  #Method 2

    return d
```

In [11]:
```
print(get_frequencies(["A", "B", "A", "A", "A"]))
print(get_frequencies([1, 2, 3, 3, 2]))
print(get_frequencies([True, False, True, False, False]))
print(get_frequencies([]))
```

```
{'A': 4, 'B': 1}
{1: 1, 2: 2, 3: 2}
{True: 2, False: 3}
{}
```

In [ ]:
```

```

Question 6

Q6) Create a function that takes a dictionary representing student information and calculates the Grade Point Average using the formula explained below, and returns the following string: sol_format = a a = " {student_name from dict} GPA for {student_semester from dict} is calculated_gpa" # Formula Grade Point

Average # Quality Points : A -> 4 B -> 3 C -> 2 D -> 1 F -> 0 # Example : grades =["A","A"] credit_hrs = [3,3] # gpa = totalpoints / totalcredithrs = (4 * 3 + 4 * 3) / (3 + 3) -> 4.00 # Rounded to 2 Decimal Places Examples gpa_calculator({"name": "Ansha Mandal", "courses": [{"name": "cal1", "credit_hours": 5, "grade": "A"}, {"name": "kin1", "credit_hours": 3, "grade": "A"}], "semester": "Spring 2018"}) ➞ "Ansha Mandal GPA for Spring 2018 is 4.00" Notes GPA is rounded to two decimal places

In [12]:
```
def gpa_calculator(data):

    quality_points = {
        "A": 4,
        "B": 3,
        "C": 2,
        "D": 1,
        "F": 0
    }

    calculated_gpa = 0
    totalpoints = 0

    grades = []
    credit_hrs = []
    courses = data['courses']

    for i in courses:

        grades.append(i['grade'])
        credit_hrs.append(i['credit_hours'])

    for i,j in zip(grades,credit_hrs):

        totalpoints += quality_points[i]*j
    totalcredithrs = sum(credit_hrs)
    calculated_gpa = totalpoints / totalcredithrs

    print(f"{data['name']} GPA for {data['semester']} is {calculated_gpa:0.2f}")
```

In [13]:
```
gpa_calculator({"name": "Ansha Mandal", "courses": [{"name": "cal1",
"credit_hours": 5, "grade": "A"}, {"name": "kin1", "credit_hours": 3,
"grade": "A"}], "semester": "Spring 2018"})
```

```
Ansha Mandal GPA for Spring 2018 is 4.00
```

In [ ]:
```
```

Question 7

Q7) Create a function that takes a dictionary of objects like { "name": "John", "notes": [3, 5, 4] } and returns a dictionary of objects like { "name": "John", "top_note": 5 }. Examples top_note({ "name": "John", "notes": [3, 5, 4] }) ➞ { "name": "John", "top_note": 5 } top_note({ "name": "Max", "notes": [1, 4, 6] }) ➞ { "name": "Max", "top_note": 6 } top_note({ "name": "Zygmund", "notes": [1, 2, 3] }) ➞ { "name": "Zygmund", "top_note": 3 }

In [14]:
```
def top_note(data):

    d = {}

    d['name'] = data['name']
```

```
    d['top_note'] = max(data['notes'])

    return d
```

In [15]:
```
print(top_note({ "name": "John", "notes": [3, 5, 4] }))
print(top_note({ "name": "Max", "notes": [1, 4, 6] }))
print(top_note({ "name": "Zygmund", "notes": [1, 2, 3] }))
```

```
{'name': 'John', 'top_note': 5}
{'name': 'Max', 'top_note': 6}
{'name': 'Zygmund', 'top_note': 3}
```

In [ ]:
```

```

Question 8

Q8) Create a function that converts a date formatted as MM/DD/YYYY to YYYYDDMM. Examples format_date("11/12/2019") ➞ "20191211" format_date("12/31/2019") ➞ "20193112" format_date("01/15/2019") ➞ "20191501" Notes Return value should be a string.

In [16]:
```
def format_date(date):

    date_lst = date.split('/')

    fdate = ''

    for i in date_lst[::-1]:

        fdate+=i

    return fdate
```

In [17]:
```
print(format_date("11/12/2019"))
print(format_date("12/31/2019"))
print(format_date("01/15/2019"))
```

```
20191211
20193112
20191501
```

In [ ]:
```

```

Question 9

Q9) The median of a group of numbers is the middle number when the group is sorted. If the size of the group is even, the median is the average of the middle two numbers. Given a sorted list of numbers, return the median (rounded to one decimal place if the median isn't an integer). Examples median([1, 2, 4, 5, 6, 8, 8, 8, 10]) ➞ 6 median([2, 2, 6, 8, 8, 10, 10]) ➞ 8 median([1, 2, 2, 4, 7, 8, 9, 10]) ➞ 5.5

In [18]:
```
def median(data):

    data.sort()

    if(len(data)%2==0):
        return (data[len(data)//2]+data[(len(data)//2)-1])/2
    else:
```

```
        return data[len(data)//2]
```

In [19]:
```
print(median([1, 2, 4, 5, 6, 8, 8, 8, 10]))
print(median([2, 2, 6, 8, 8, 10, 10]))
print(median([1, 2, 2, 4, 7, 8, 9, 10]))
```

```
6
8
5.5
```

In [ ]:

Question 10

Q10) Create a function that returns the number of times a character appears in each word in a sentence. Treat upper and lower case characters of the same letter as being identical (e.g. a exists in Anna twice, not once). Examples char_appears("She sells sea shells by the seashore.", "s") ➞ [1, 2, 1, 2, 0, 0, 2] # "s" shows up once in "She", twice in "sells", once in "sea", twice in "shells", etc. char_appears("Peter Piper picked a peck of pickled peppers.", "P") ➞ [1, 2, 1, 0, 1, 0, 1, 3] # "p" shows up once in "Peter", ... 3 times in "peppers". char_appears("She hiked in the morning, then swam in the river.", "t") ➞ [0, 0, 0, 1, 0, 1, 0, 0, 1, 0]

In [20]:
```
def char_appears(data,char):

    char = char.lower()
    data_lst = data.lower().split(' ')

    count_lst = []

    for i in data_lst:

        count_lst.append(i.count(char))

    return count_lst
```

In [21]:
```
print(char_appears("She sells sea shells by the seashore.", "s"))
print(char_appears("Peter Piper picked a peck of pickled peppers.", "P"))
print(char_appears("She hiked in the morning, then swam in the river.", "t"))
```

```
[1, 2, 1, 2, 0, 0, 2]
[1, 2, 1, 0, 1, 0, 1, 3]
[0, 0, 0, 1, 0, 1, 0, 0, 1, 0]
```

In [ ]:

Question 11

Q11) Create a function that takes a number as an argument and returns True or False depending on whether the number is symmetrical or not. A number is symmetrical when it is the same as its reverse. Examples is_symmetrical(7227) ➞ True is_symmetrical(12567) ➞ False is_symmetrical(44444444) ➞ True is_symmetrical(9939) ➞ False is_symmetrical(1112111) ➞ True

In [22]:
```
def is_symmetrical(num):

    str_num = str(num)
```

```
    if(str_num == str_num[::-1]):
        return True
    return False
```

In [23]:
```
print(is_symmetrical(7227))
print(is_symmetrical(12567))
print(is_symmetrical(44444444))
print(is_symmetrical(9939))
print(is_symmetrical(1112111))
```

```
True
False
True
False
True
```

In [ ]:

Question 12

Q12) Create a function that takes three parameters where: • x is the start of the range (inclusive). • y is the end of the range (inclusive). • n is the divisor to be checked against.Return an ordered list with numbers in the range that are divisible by the third parameter n . Return an empty list if there are no numbers that are divisible by n . Examples list_operation(1, 10, 3) ➞ [3, 6, 9] list_operation(7, 9, 2) ➞ [8] list_operation(15, 20, 7) ➞ []

In [24]:
```
def list_operation(start,end,divisor):

    l = []

    for i in range(start,end+1):

        if(i%divisor==0):
            l.append(i)

    return l
```

In [25]:
```
print(list_operation(1, 10, 3))
print(list_operation(7, 9, 2))
print(list_operation(15, 20, 7))
```

```
[3, 6, 9]
[8]
[]
```

In [ ]:

Question 13

Q13) write a program which count and print the numbers of each character in a string input by console. Example: If the following string is given as input to the program: abcdefgabc Then, the output of the program should be: {'a':2 'c':2 ,'b':2 'e':1 'd':1, 'g':1 'f':1}

In [26]:
```
def string_count(data):

    d = {}
```

```
    for i in data:

        d[i] = data.count(i)

    return d
```

In [27]:
```
string_count(input("Enter anything: "))
```

```
Enter anything: adinafisnfgwoo
```

Out[27]:
```
{'a': 2, 'd': 1, 'i': 2, 'n': 2, 'f': 2, 's': 1, 'g': 1, 'w': 1, 'o': 2}
```

In [ ]:

Question 14

Q14) With a given list [12,24,35,24,88,120,155,88,120,155], write a program to print this list after removing all duplicate values with original order reserved.

In [3]:
```
def remove_duplicates(data):

#     for i in range(len(data)):

#         for j in range(i+1,len(data)):

#             if data[i]==data[j]:
#                 data.remove(data[j])


    s = set(data)

    data = list(s)

    data.sort()

    return data[::-1]
```

In [4]:
```
print(remove_duplicates([12,24,35,24,88,120,155,88,120,155]))
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In [4], line 1
----> 1 print(remove_duplicates([12,24,35,24,88,120,155,88,120,155]))

Cell In [3], line 10, in remove_duplicates(data)
      7         else:
      8             for j in range(i+1,len(data)):
---> 10                if data[i]==data[j]:
     11                    data.remove(data[j])
     14 #     s = set(data)
     15
     16 #     data = list(s)
     17
     18 #     data.sort()

IndexError: list index out of range
```

In [ ]:

Question 15

Q15) In this challenge, a farmer is asking you to tell him how many legs can be counted among all his animals. The farmer breeds three species: •chickens = 2 legs •cows = 4 legs •pigs = 4 legs The farmer has counted his animals and he gives you a subtotal for each species. You have to implement a function that returns the total number of legs of all the animals. Examples animals(2, 3, 5) ⟶ 36 animals(1, 2, 3) ⟶ 22 animals(5, 2, 8) ⟶ 50

In [30]:
```
def animals(chickens, cows, pigs):

    return chickens*2 + cows*4 + pigs*4
```

In [31]:
```
print(animals(2, 3, 5))
print(animals(1, 2, 3))
print(animals(5, 2, 8))
```

```
36
22
50
```

In [ ]:

In [ ]: