Session 1 - Python : Assignment

The Questions are given below followed by the code placeholder for each questions. Write down your solution in the provided cell.

Question 1:

Manhattan distance is the distance between two points in a grid calculated by only taking a vertical and/or horizontal path. Complete the function that accepts two points and returns the Manhattan Distance between the two points. The points are arrays or tuples containing the `x` and `y` coordinate in the grid. You can think of `x` as the row in the grid, and `y` as the column.

Example

Input `[1, 1], [1, 1]` => Output `0` Input `[5, 4], [3, 2]` => Output `4` Input `[1, 1], [0, 3]` => Output `3`

+ Code + Text

```
def manhattan_distance(point1,point2):
    x1,y1=point1
    x2,y2=point2
    return abs(x1-x2)+abs(y1-y2)

#point1
x1=int(input("Enter x coordinate for p1: "))
y1=int(input("Enter y coordinate for p1: "))
point1=(x1, y1)
#point2
x2=int(input("Enter x coordinate for p2: "))
y2=int(input("Enter y coordinate for p2: "))
point2=(x2, y2)

dis=manhattan_distance(point1,point2)
print(dis)
```

```
Enter x coordinate for p1: 1
Enter y coordinate for p1: 1
Enter x coordinate for p2: 0
Enter y coordinate for p2: 3
3
```

Question 2:

A list of integers is sorted in “Wave” order if alternate items are not less than their immediate neighbors (thus the other alternate items are not greater than their immediate neighbors). Thus, the array `[4, 1, 7, 5, 6, 2, 3]` is in Wave order because `4 >= 1`, then `1 <= 7`, then `7 >= 5`, then `5 <= 6`, then `6 >= 2`, and finally `2 <= 3`. The wave-sorted lists has to begin with an element not less than the next, so `[1, 4, 5, 3]` is not sorted in Wave because `1 < 4`. Your task is to implement a function that takes a list of integers and return true if the list is in wave order.

Example

Input `[4, 1, 7, 5, 6, 2, 3]` => Output `True` Input `[1, 4, 5, 3]` => Output `False`

```
def is_wave_sorted(arr):
    n=len(arr)
    if n<2:
        return False

    for i in range(0,n-1,2):
        if arr[i]<arr[i+1]:
            return False
    for i in range(1,n-1,2):
        if arr[i]>arr[i+1]:
            return False
    return True

arr=list(map(int,input("Enter list of integers separated by space: ").split()))
if is_wave_sorted(arr):
    print("True")
else:
    print("False")
```

```
Enter list of integers separated by space: 1 4 5 3
False
```

Question 3:

Balanced number is the number where the sum of all digits to the left of the middle digit(s) and the sum of all digits to the right of the middle digit(s) are equal. Your task is to implement a function that takes a number and return true if it is a balanced number else return False.

Example

Input `959` => Output `True` Input `27102983` => Output `False`

```
def is_balanced_number(number):
    number_str=str(number)
    length=len(number_str)
    if length%2==1:
        middle_index=length//2
        number_str=number_str[:middle_index]+number_str[middle_index+1:]
    number=int(number_str)

    digits=[int(digit) for digit in str(number)]
    split_index=len(digits)//2

    left_sum=sum(digits[:split_index])
    right_sum=sum(digits[split_index:])
    return left_sum==right_sum

num=int(input("Enter a number: "))
if is_balanced_number(num):
    print("True")
else:
    print("False")
```

```
Enter a number: 27102983
False
```

Question 4:

You are given a sequence of valid words and a string. Test if the string is made up by one or more words from the array. Test if the string can be entirely formed by consecutively concatenating words of the dictionary.

Example

valid_words: `["test", "ing"]` Input `"testing"` => Output `True` Input `"tester"` => Output `False`

```
def is_formed_by_words(words,stri):
    n=len(stri)
    word_set=set(words)
    dp=[False]*(n+1)
    dp[0]=True

    for i in range(1,n + 1):
        for j in range(i):
            if dp[j] and stri[j:i] in word_set:
                dp[i] = True
                break

    return dp[n]

words=input("Enter valid words separated by space: ").split()
stri=input("Enter a string to test: ")

if is_formed_by_words(words,stri):
    print("True")
else:
    print("False")
```

```
Enter valid words separated by space: te fd
Enter a string to test: ted
False
```

Question 5:

Write a function that accepts an array of `10` integers (between `0` and `9`), that returns a string of those numbers in the form of a phone number.

Example

Input `[1, 2, 3, 4, 5, 6, 7, 8, 9, 0]` => Output `"(123) 456-7890"`

```
def format_phone_number(numbers):
    number_str=''.join(map(lambda x:f"{x}",numbers))
    formatted_number="({}) {}-{}".format(number_str[:3],number_str[3:6],number_str[6:])
    return formatted_number

numbers=list(map(int,input("Enter 10 integers separated by space: ").split()))
phone_number=format_phone_number(numbers)
print(phone_number)
```

```
Enter 10 integers separated by space: 8 3 1 7 6 5 9 4 8 9
(831) 765-9489
```

Question 6:

Given a dictionary, find mean of all the values present.

Example

Input `{"Gfg": 4, "is": 4, "Best": 4, "for": 4, "Geeks": 4}` => Output `4` Input `{"Gfg": 5, "is": 10, "Best": 15}` => Output `10`

```
def mean_of_values(dictionary):
    total_sum=sum(dictionary.values())
    mean=total_sum/len(dictionary)
    return mean

input_dict=eval(input("Enter a dictionary in the format {'key1': value1, 'key2': value2, ...}: "))
mean=mean_of_values(input_dict)
print(mean)
```

```
Enter a dictionary in the format {'key1': value1, 'key2': value2, ...}: {'gfg':10,'best':20}
15.0
```

Question 7:

Given a list of numbers, remove floats (numbers with decimals). Use list comprehension to solve.

Example

Input `[1, 2.3, 5, 6, 6.89, 7.0]` => Output `[1, 5, 6]`

```
def remove_floats(numbers):
    return [int(num) for num in numbers if int(num)==num]

numbers=list(map(float, input("Enter a list of numbers separated by space: ").split()))
result=remove_floats(numbers)
print(result)
```

```
Enter a list of numbers separated by space: 1 2.3 4 5.6 7 8.9
[1, 4, 7]
```