Week-3 Graded Assignment (Programming)

Problem 1

Question [8]

Write a program to find the sum of the following series up to n terms

$1 + (1 + 2) + (1 + 2 + 3) + (1 + 2 + 3 + 4) + \ldots\ldots n\ terms$

Sample Input and Outputs

Input	Output
`3`	`10`
`5`	`35`

Answer

```python
n = int(input())
total = 0
for i in range(1, n + 1):
    for j in range(1, i+1):
        total = total + j
print(total)
```

Testcases

Public

Input	Output
`3`	`10`
`5`	`35`

Private

Input	Output	Weight
`0`	`0`	1
`10`	`220`	3
`234`	`2162940`	3
`1`	`1`	1

Solution

The input from the user is parsed into an integer datatype and stored in the variable `n`. `total` is a variable which stores the required output which initialized as 0.

The required sum of the sequence can be mathematically expressed as,

$$1 + (1 + 2) + (1 + 2 + 3) + (1 + 2 + 3 + 4) + \ldots\ldots n\ terms = \sum_{i=1}^{n} \sum_{j=1}^{i} j$$

The above mathematical expression can be written as a nested for-loop of `range(1, n + 1)` and `range(1, i + 1)` respectively. The sum is stored in the variable `total`.

```
for i in range(1, n + 1):
    for j in range(1,i + 1):
        total = total + j
```

Problem 2

Question [8]

Write a program to find all the prime factors of a number which taken as input from user. Assume the accepted input as positive integer which is greater than or equal to 2.

Sample Input 1

```
15
```

Sample Output 1

```
3
5
```

Sample Input 2

```
79
```

Sample Output 2

```
79
```

Sample Input 3

```
78
```

Sample Output 3

```
2
3
13
```

Answer 1

```
n = int(input())
if n == 2:
    print(2)
if n > 2:
    for i in range(2, n+1):
        if (n %  i == 0):
            flag = True
            for j in range(2, i):
                if (i % j == 0):
                    flag = False
                    break
            if flag:
                print(i)
```

Answer 2

```python
n = int(input())
if n == 2:
    print(2)
if n > 2:
    for i in range(2, n+1):
        if (n %  i == 0):
            for j in range(2, i):
                if (i % j == 0):
                    break
            else:
                print(i)
```

Testcases

Public

Input 1

```
15
```

Output 1

```
3
5
```

Input 2

```
79
```

Output 2

```
79
```

Input 3

```
78
```

Output 3

```
2
3
13
```

Private [2 + 2 + 2 + 2 = 8]

Input 1

```
1234
```

Output 1

```
2
617
```

Input 2

```
6217
```

Output 2

```
6217
```

Input 3

```
517
```

Output 3

```
11
47
```

Input 4

```
2
```

Output 4

```
2
```

Solution

The prime factors for the given number is the prime numbers which can be divide the given number.

The prime number starts with 2 and keeps going on. So the prime factor for the number 2 is 2, which is directly printed using the if-statement.

```
if n == 2:
    print(2)
```

If the number from the user is greater than 2, then the control goes to `for` loop with `i` of `range(2, n + 1)` (line-5) which starts at 2 and ends at `n`.

When any `i` divides the input number `n` , we verify if it is a prime number using the nested `for` loop (line-7 onwards).

Problem 3

Question [14]

Accept a string as an input from the user, convert this to lower case and print the string in ascending order using loop.

Note: Ignore non-alphabetic characters in the output

Sample Input 1

```
Bharatanatyam
```

Sample Output

```
aaaaabhmnrtty
```

Sample Input 2

```
monty python
```

Sample Output

```
hmnnoopttyy
```

Answer

```
text = input().lower()
length = len(text)
output_string = ""
for char in "abcdefghijklmnopqrstuvwxyz":
    for i in range(0, length):
        if char == text[i]:
            output_string += char
print(output_string)
```

Testcases

Public

Input	Output
`Bharatanatyam`	`aaaaabhmnrtty`
`monty python`	`hmnnoopttyy`

Private

Input	Output	Weight
`bharatanatyam`	`aaaaabhmnrtty`	4
`montypython`	`hmnnoopttyy`	4
`101 @#$%&`		3
`James Bond Agent 007`	`aabdeegjmnnost`	3

Solution

In the solution, we iterate over all the alphabets in the "English" language arranged in ascending order. In each iteration, we iterate over (nested iteration) each characters in the given input string and check if the alphabets matches any of the characters in the input string. If yes, we add them to the `output_string`, once iterations are finished the ordered characters sequence stored in this variable is printed.

Problem 4

Question [6]

Accept the string from the user and display the characters located on prime index (2,3,5,7..) of a given string.

Sample Input 1

```
hellopython
```

Sample Output 1

```
l
l
p
t
```

Sample Input 2

```
this is python program
```

Sample Output 2

```
i
s
i

h
n
o
r
```

Answer

```
st = input()
length = len(st)
count = 0
for i in range(2, length):
    count = 0
    for j in range(2, i):
        if (i % j == 0):
            count = count + 1
    if (count == 0):
        print(st[i])
```

Testcases

Public

Input 1

```
hellopython
```

Output 1

```
l
l
p
t
```

Input 2

```
this is python program
```

Output 2

```
i
s
i

h
n
o
r
```

Private [2 + 2 + 2 = 6]

Input 1

```
abcdefghijklmnopqrstuvwxyz
```

Output 1

```
c
d
f
h
l
n
r
t
x
```

Input 2

```
0123456789
```

Output 2

```
2
3
5
7
```

Input 3

```
abcdefghijkl
```

Output 3

```
c
d
f
h
l
```

Solution

Accept the input string from the user and assign it to variable `st`. Calculate the length of the string and assign it to variable `length`. Inside the `for` loop, we check each index in the string (`i` iterates from `2` to `length-1`), if it is a prime number then the character at the index `i` which is `st[i]`, is printed.

Problem 5

Question [14]

Accept phone number from user as string.

A valid phone number should satisfy the following constraints.

- The number should start with 6,7,8 and 9.
- Length of the number should be 10.
- Number should not have any digit repeated more then 5 times in consecutive manner.
- Number should not have any digit more than 7 times.

Input	Output
`9852546666`	`valid`
`0984587558`	`invalid`
`85425@6558`	`invalid`

Answer

```
number = input()
valid = True
if len(number) == 10 and number.isdigit() == True and number[0] in '6789':
    for digit_index in range(0,5):
        count = number.count(number[digit_index])
        if count > 7:
            valid = False
            break
        if 6*number[digit_index] in number:
            valid = False
            break
else:
    valid =False
if valid == True:
    print("valid")
else:
    print('invalid')
```

Testcases

Public

Input	Output
`9852546666`	`valid`
`0984587558`	`invalid`
`85425@6558`	`invalid`

Private

Input	Output	Weight
`8181818181`	`valid`	3
`587888888`	`invalid`	2
`8988899998`	`valid`	3
`9898999999`	`invalid`	2
`8989.89898`	`invalid`	2
`9999987999`	`invalid`	2

Solution

- Accept phone number as a string from the user.

- Initialize a variable `valid` with bool value `True`.
- In line 3, check three conditions with the number. If any of them is False then assign `False` Value to `valid` in the else block.
- Otherwise check each digit one by one from 0th to 4th index because only 0th to 4th index digit can be repeated 6 times in 10 digit phone numbers.
- In line 6, check if the counting of any digit is more than 7 then assign `False` value to `valid` and break the loop.
- In line 9, check if the number has digit in consecutive manner 6 times then assign False value to valid and break the loop.
- In the last, according to the value of `valid` , print the output.