NAME : Jannat Shaikh
SUBJECT : PYTHON (PRACTICAL)
COURSE: (MSC(CS)-4)
BATCH - 2
ROLL_NO: 103
****
ASSIGNMENT-1
****

```
# 1. Write a python program to add two numbers.

a = int(input("enter the num 1 "))
b = int(input("enter the num 2 "))
sum = a + b
print("sum os two number is  : ", sum)
```

output:

```
enter the num 1 3
enter the num 2 2
sum os two number is  :  5
```

****

```
# 2) Write a python program to find quotient and remainder.

d = int(input("enter the divisor : "))
dd = int(input("enter the dividend : "))
print("quotient is : ", d / dd)
print("remainder is :", d % dd)
```

output:

```
enter the divisor : 13
enter the dividend : 2
quotient is :  6.5
remainder is : 1
```

****

```
# 3) Write a python program to find type of objects in your system.

x = 10
print(type(x))

s = 'abc'
print(type(s))

from collections import OrderedDict
```

```
od = OrderedDict()
print(type(od))

class Data:
    pass

d = Data()
print(type(d))
```

output :

```
<class 'int'>
<class 'str'>
<class 'collections.OrderedDict'>
<class '__main__.Data'>
```

****

4) Write a python program to read two numbers and display the greatest number.

```
num1 = input("enter number 1 : ")
num2 = input("enter number 2 : ")

if num1 <  num2:
    print("num1 is bug",num1)
else:
    print("num2 is big",num2)
```

output :

```
enter number 1 : 10
enter number 2 : 6
num1 is bug 10
```

****

5) Write a python program to check whether number is even or odd.

```
# even like 2,4,6 etc..
# odd like 1,3 etc..


num1 = int(input("enter number 1 : "))

if (num1 % 2) == 0:
    print("number is even")
else:
    print("number is odd")
```

output :
enter number 1 : 2
number is even

enter number 1 : 3
number is odd
****
6) Write a python program to check whether a character is vowel or consonant.

```
c = str(input("enter character : "))

if c == 'a' or c == 'e' or c == 'i' or c == 'o' or c == 'u':
    print("character is vowel")
else:
    print("character is constant")
```

output :

enter character : i
character is vowel

enter character : h
character is constant

****
7) Write a python program to find largest number among three numbers.

```
a = int(input("enter first number : "))
b = int(input("enter second number :"))
c = int(input("enter the second number : "))

if a > b and a > c :
    print("a is a big number : ",a)
elif b > a and b > c :
    print("b is a big number : ",b)
elif c > a and c > b :
    print("c is a big number : ",c)
```

output :
enter first number : 3
enter second number :10
enter the second number : 66

c is a big number : 66
****
#8) Write a python program to find all roots of a quadratic equation.

```
# formula is :---
# x = [-b ± √(b2 - 4ac)]/2a.

#x2 - 7x + 10 = 0 are x = 2 and x = 5

import cmath
a = int(input("enter a number : "))
b = int(input("enter b number :"))
c = int(input("enter c number : "))
s = (b ** 2) - (4 * a * c)
sum1 = (-b + cmath.sqrt(s)) / (2 * a)
print("sum1 in + is : ",sum1)
sum2 = (-b - cmath.sqrt(s)) / (2 * a)
print("sum2 in + is : ",sum2)
```

output :
enter a number : 2
enter b number :7
enter c number : 10
sum1 in + is : (-1.75+1.3919410907075054j)
sum2 in + is : (-1.75-1.3919410907075054j)

****
#9) Write a python program to calculate sum of natural numbers.

```
sum = 0
for i in range(0,100):
   sum = sum + i
   i = i + 1
else:
    print(sum)
```

output :
sum of natural number is : 4950
****
10) Write a python program to check leap year.
note :
if a year is divisible by 400, then it is still considered a leap year.
For example, the year 2000 was a leap year, but the year 2100 will not be.

```
# or (year % 400 == 0) (optional)

year = int(input("enter the year : "))

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        print(year,"is a leap year")
else:
        print(year,"is not a leap year)
```

output :
enter the year : 2000
2000 is a leap year

enter the year : 2001
2001 not a leap year

****

11) Write a python program to find factorial.

```
a = int(input("enter the number : "))
i = 1
count = 1
while a >= i:
    count = count * i
    i = i + 1

print(count)
```

output :
enter the number : 5
120

****

12) Write a python program to generate multiplication table.

```
a = int(input("enter the number : "))
i = 1
while i <= 10:
   print(a , " * ", i , " = ", a * i)
   i = i + 1
```

output :
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20

```
5  *  5  =  25
5  *  6  =  30
5  *  7  =  35
5  *  8  =  40
5  *  9  =  45
5  *  10  =  50
```

****

13) Write a python program to display fibonacci series. (eg. 0, 1, 1, 2, 3, 5, 8, 13, ...)

```
a = 0
b = 1
c = 0
print("fibonacci : ")
while c <= 10:
    print(c)
    a = b
    b = c
    c = a + b
```

output :

```
fibonacci :
0
1
1
2
3
5
8
```

****

14) Write a python program to find gcd.

```
num1 = int(input("enter the num 1 : "))
num2 = int(input("enter the num2 : "))

while(num2 != 0):
    num1,num2 = num2,num1 % num2

gcd = num1
print("gcd is a : ",gcd)
```

output :

```
enter the num 1 : 20
```

```
enter the num2 : 10
gcd is a :  10
************************************************************
```

15) Write a python program to find lcm.

```
num1 = int(input("enter the num 1 : "))
num2 = int(input("enter the num2 : "))

a = num1
b = num2

while(num2 != 0):
    num1,num2 = num2,num1 % num2

gcd = num1
print("gcd is a : ",gcd)

lcm = a * b /gcd

print("lcm is  : ",lcm)
```

output :

```
enter the num 1 : 36
enter the num2 : 70
gcd is a :  2
lcm is  :  1260.0
************************************************************
```

16) Write a python program to reverse a number.

```
i = int(input("enter the number : "))
rev = 0
while(i > 0):
   rev = (rev * 10) + i % 10
   i = i//10

print("reverse the number " , rev)
```

output:

```
enter the number : 6758
reverse the number  8576
*****************************************
```

17) Write a python program to calculate power of a number.

```
#********** first method *******
#import math
#print(math.pow(2,2))

#**********second method **********

a= int(input("enter power"))
b = int(input("enter number"))
i = 1
power = 1
while i <= b:
    power = power*a
    i = i + 1

print(power)
```

output:
enter power3
enter number3
27

****

18) Write a python program to find binary value of a character.

```
def char_to_binary(character):
   binary_value = bin(ord(character))[2:]
   return binary_value.zfill(8)

character = input("Enter a character: ")
binary_representation = char_to_binary(character)
print(f"The binary value of '{character}' is: {binary_representation}")
```

output:
Enter a character: A
The binary value of 'A' is: 01000001

****

19) Write a python program to display two separate strings in single line continuously.

```
str1 = str(input("enter first string : "))
str2 = str(input("enter second stting : "))

print(str1 +" " + str2)
```

output :
enter first string : drashti
enter second string : patoliya
drashti patoliya

****
20) Write a python program to check whether a number is palindrome or not.

```
i = int(input("enter the number : "))
number_original = i
rev = 0
while(i > 0):
  rev = (rev * 10) + i % 10
  i = i//10

print("reverse the number " , rev)

if(number_original == rev):
   print("number is palindrome
else:
   print("number is not palindrome
```

output :
yes
enter the number : 121
reverse the number 121
number is palindrome

not
enter the number : 456
reverse the number 654
number is not palindrome
****
21) Write a python program to check whether a number is prime or not.

```
num = int(input("enter the number : "))
count = 0
for i in range(1,num + 1):
   if(num % i == 0):
      count = count + 1

if(count == 2):
```

```
    print("given number is prime ")
else:
    print("given number is not prime")

output :

enter the number : 2
given number is prime

enter the number : 6
given number is not prime
```
****
22) Write a python program to display prime numbers between two intervals.

```
min = int(input("enter the number : "))
max = int(input("enter the second number : "))

for num in range(min,max+1):
    if(num > 1):
        for i in range(2,num):
            if num % i == 0:
                break
        else:
            print(num)

output :

enter the number : 1
enter the second number : 15
2
3
5
7
11
13
```
****
23) Write a python program to check armstrong number. (eg. 13 + 33 + 53 = 153)

```
num = input("enter the number : ")

sum = 0

for i in num:
```

```
    sum = sum + int(i) ** 3
if sum == int(num):
    print("armstrong number")
else:
    print("armstring number is not")
```

output :

```
enter the number : 125
armstring number is not

enter the number : 153
armstrong number
```

****

24) Write a python program to display armstrong number between two intervals.

```
num = int(input("enter the number : "))
for i in range(1,num +1):
    new = i
    sum = 0
    for j in range(1,i+1):
        rem = new % 10
        sum += rem ** 3
        new //= 10
    if(sum == i):
        print("yes",i)
```

output :

```
enter the number : 1000
yes 1
yes 153
yes 370
yes 371
yes 407
```

****

25) Write a python program to display factors of a number.

```
n = int(input("enter the number : "))
for x in range(1,n +1):
     if n % x == 0:
         print(x)
```

output :

```
enter the number : 3
1
3
```

****

26) Write a python programs to create pyramid and pattern.

```
num = int(input("enter the number"))
row = 0
j = 0
for row in range(row,num+1):
    print(" ")
    for coumn in range(1,row+1):
       print("*",end= " ")
```

output:

```
*
* *
* * *
* * * *
```

****

27) Write a python program to make a simple calculator to add, subtract, multiply or divide

```
ch = str(input("enter the choice"))
a = 1
b = 2
if(ch == "add"):
    print(a + b)
elif(ch == "sub"):
    print(a - b)
elif(ch == "mul"):
    print(a * b)
elif(ch == "divided"):
    print(a / b)
```

output :

```
enter the choice : mul
2
```

****

28) Write a python program to display Simple Number Triangle Pattern

```
def print_number_triangle(n):
    for i in range(1, n + 1):
        for j in range(1, i + 1):
            print(j, end=" ")
        print()

rows = int(input("Enter the number of rows for the triangle: "))
print("Simple Number Triangle Pattern:")
print_number_triangle(rows)
```

output:

```
Simple Number Triangle Pattern:
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

****

29) Write a python program to display inverted pyramid of numbers

```
def inverted_pyramid_numbers(n):
    for i in range(n, 0, -1):
        for j in range(1, i + 1):
            print(j, end=" ")
        print()

rows = int(input("Enter the number of rows for the inverted pyramid: "))
print("Inverted Pyramid of Numbers:")
inverted_pyramid_numbers(rows)
```

Output (for example, if the user enters 5):

output:

```
Inverted Pyramid of Numbers:
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

****

30) Write a python program to display inverted pyramid of descending numbers

```
def inverted_pyramid_descending_numbers(n):
    num = 1
    for i in range(n, 0, -1):
        for j in range(1, i + 1):
            print(num, end=" ")
            num += 1
        print()

rows = int(input("Enter the number of rows for the inverted pyramid: "))
print("Inverted Pyramid of Descending Numbers:")
inverted_pyramid_descending_numbers(rows)
```

Output

```
Inverted Pyramid of Descending Numbers:
1 2 3 4
5 6 7
8 9
10
```

****

31) Write a python program to display inverted pyramid of the same digit

```
def inverted_pyramid_same_digit(n, digit):
    for i in range(n, 0, -1):
        for j in range(1, i + 1):
            print(digit, end=" ")
        print()

rows = int(input("Enter the number of rows for the inverted pyramid: "))
digit = input("Enter the digit to repeat: ")
print("Inverted Pyramid of the Same Digit:")
inverted_pyramid_same_digit(rows, digit)
```

Output

```
Inverted Pyramid of the Same Digit:
8 8 8 8
8 8 8
8 8
8
```

****

32) Write a python program to display reverse pyramid of numbers

```
def reverse_pyramid_numbers(n):
    num = 1
```

```
    for i in range(n, 0, -1):
        for _ in range(n - i):
            print(" ", end=" ")
        for j in range(1, i + 1):
            print(num, end=" ")
            num += 1
        print()

rows = int(input("Enter the number of rows for the reverse pyramid: "))
print("Reverse Pyramid of Numbers:")
reverse_pyramid_numbers(rows)
```

output:

```
Reverse Pyramid of Numbers:
1 2 3 4 5
  6 7 8 9
    10 11 12
      13 14
        15
```

****

33) Write a python program to display inverted half pyramid number pattern

```
def inverted_half_pyramid(n):
    for i in range(n, 0, -1):
        for j in range(i):
            print(i, end=" ")
        print()

rows = int(input("Enter the number of rows for the inverted half pyramid: "))
print("Inverted Half Pyramid Number Pattern:")
inverted_half_pyramid(rows)
```

output:

```
Inverted Half Pyramid Number Pattern:
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```

****

34) Write a python program to display pyramid of natural numbers less than 10

```
def pyramid_natural_numbers():
```

```
    num = 1
    for i in range(1, 5):
        for j in range(1, i + 1):
            print(num, end=" ")
            num += 1
        print()

print("Pyramid of Natural Numbers Less Than 10:")
pyramid_natural_numbers()

output:
Pyramid of Natural Numbers Less Than 10:
1
2 3
4 5 6
7 8 9 10
```

****

35) Write a python program to display reverse pattern of digits from 10

```
def reverse_pattern_digits():
    num = 10
    for i in range(4, 0, -1):
        for j in range(1, i + 1):
            print(num, end=" ")
            num -= 1
        print()

print("Reverse Pattern of Digits from 10:")
reverse_pattern_digits()
output:
Reverse Pattern of Digits from 10:
10 9 8 7
6 5 4
3 2
1
```

****

36) Write a python program to display connected inverted pyramid pattern of numbers

```
def connected_inverted_pyramid(n):
    for i in range(n, 0, -1):
        print(" " * (n - i), end="")
        for j in range(i, 0, -1):
            print(j, end="")
        for j in range(2, i + 1):
            print(j, end="")
```

```
        print()

rows = int(input("Enter the number of rows for the connected inverted pyramid: "))
print("Connected Inverted Pyramid Pattern of Numbers:")
connected_inverted_pyramid(rows)
output:
Connected Inverted Pyramid Pattern of Numbers:
54321
 43212
  321
   21
    1
**************************************************************************
37) Write a python program to display even number pyramid pattern
def even_number_pyramid(n):
    num = 2
    for i in range(1, n + 1):
        for j in range(1, i + 1):
            print(num, end=" ")
            num += 2
        print()

rows = int(input("Enter the number of rows for the even number pyramid: "))
print("Even Number Pyramid Pattern:")
even_number_pyramid(rows)
output:
Even Number Pyramid Pattern:
2
4 6
8 10 12
14 16 18 20
*****************************************************
38) Write a python program to display pyramid of horizontal tables
def horizontal_table_pyramid(n):
    for i in range(1, n + 1):
        for j in range(1, i + 1):
            print(f"{i*j:2}", end=" ")
        print()

rows = int(input("Enter the number of rows for the horizontal table pyramid: "))
print("Pyramid of Horizontal Tables:")
horizontal_table_pyramid(rows)
```

output:
Pyramid of Horizontal Tables:
 1
 2 4
 3 6 9
 4 8 12 16

****

39) Write a python program to display pyramid pattern of alternate numbers

```
def alternate_number_pyramid(n):
    num = 1
    for i in range(1, n + 1):
        for j in range(1, i + 1):
            print(num, end=" ")
            num += 2
        print()

rows = int(input("Enter the number of rows for the alternate number pyramid: "))
print("Pyramid Pattern of Alternate Numbers:")
alternate_number_pyramid(rows)
```

output:
Pyramid Pattern of Alternate Numbers:
1
3 5
7 9 11
13 15 17 19

****

40) Write a python program to display mirrored pyramid (right-angled triangle) pattern of numbers

```
def mirrored_pyramid(n):
    for i in range(1, n + 1):
        print(" " * (n - i) + " ".join(str(j) for j in range(1, i + 1)))

rows = int(input("Enter the number of rows for the mirrored pyramid: "))
print("Mirrored Pyramid Pattern of Numbers:")
mirrored_pyramid(rows)
```

output:
Mirrored Pyramid Pattern of Numbers:
    1
   1 2
  1 2 3

```
 1 2 3 4
1 2 3 4 5

**************************************************************************
```
41) Write a python program to display equilateral triangle with stars (asterisk symbol)
```
def equilateral_triangle_stars(n):
    for i in range(1, n + 1):
        print(" " * (n - i) + "*" * (2*i - 1))

rows = int(input("Enter the number of rows for the equilateral triangle: "))
print("Equilateral Triangle Pattern with Stars:")
equilateral_triangle_stars(rows)
```
output:
```
    *
   ***
  *****
 *******
*********
**********************************************************
```
42) Write a python program to display downward triangle pattern of stars
```
def downward_triangle_stars(n):
    for i in range(n, 0, -1):
        print("*" * i)

rows = int(input("Enter the number of rows for the downward triangle: "))
print("Downward Triangle Pattern of Stars:")
downward_triangle_stars(rows)
```
output:
```
****
***
**
*
************************************************
```
43) Write a python program to display pyramid pattern of stars
```
def pyramid_stars(n):
    for i in range(1, n + 1):
        print(" " * (n - i) + "*" * (2*i - 1))

rows = int(input("Enter the number of rows for the pyramid: "))
print("Pyramid Pattern of Stars:")
pyramid_stars(rows)
```
output:

```
  *
 ***
*****
*******
***************************************
def hourglass_pattern(n):
    for i in range(n, 0, -1):
        print(" " * (n - i) + "*" * (2*i - 1))
    for i in range(2, n + 1):
        print(" " * (n - i) + "*" * (2*i - 1))

rows = int(input("Enter the number of rows for the hourglass pattern: "))
print("Hourglass Pattern Program:")
hourglass_pattern(rows)

output:
*******
 *****
  ***
   *
   *
  ***
 *****
*******
***************************************
```

44) Write a python program to display hourglass pattern program

```
def pascal_triangle(n):
    for line in range(1, n + 1):
        C = 1
        for i in range(1, line + 1):
            print(C, end=" ")
            C = int(C * (line - i) / i)
        print()

rows = int(input("Enter the number of rows for Pascal's triangle: "))
print("Pascal's Triangle Program:")
pascal_triangle(rows)

output:
1
1 1
1 2 1
1 3 3 1
```

1 4 6 4 1
****
45) Write a python program to display pascal's triangle program

```
def generate_pascals_triangle(num_rows):
    triangle = []

    for i in range(num_rows):
        row = [1]  # First element of each row is always 1
        if i > 0:
            # Generate the current row based on the previous row
            for j in range(1, i):
                row.append(triangle[i - 1][j - 1] + triangle[i - 1][j])
            row.append(1)  # Last element of each row is always 1
        triangle.append(row)

    return triangle

def display_pascals_triangle(triangle):
    for row in triangle:
        # Center-align each number in the row
        row_str = ' '.join(str(num) for num in row)
        print(row_str.center(len(row_str) + (len(triangle) - len(row_str))))


num_rows = int(input("Enter the number of rows for Pascal's triangle: "))

triangle = generate_pascals_triangle(num_rows)
display_pascals_triangle(triangle)
```

output:

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
```

****

ASSIGNMENT

****

Strings

_____

1. Write a Python function to read and reverse a string.

```
def reverse_string(input_str):
    return input_str[::-1]
```

```
print(reverse_string("hello"))
```

Output: "olleh"

_____

2. Write a Python function to count the number of vowels in a given string.

```
def count_vowels(input_str):
    vowels = "aeiouAEIOU"
    return sum(1 for char in input_str if char in vowels)

print(count_vowels("hello"))
```

Output: 2

_____

3. Write a Python function to remove duplicate characters from a string.

```
def remove_duplicates(input_str):
    return ''.join(sorted(set(input_str), key=input_str.index))

print(remove_duplicates("hello"))
```

Output: "helo"

_____

4. Write a Python function to check if a given string is a palindrome.

```
def is_palindrome(input_str):
    return input_str == input_str[::-1]

print(is_palindrome("radar"))
```

Output: True

_____

5. Write a Python function to count the number of words in a given string.

```
def count_words(input_str):
    return len(input_str.split())

print(count_words("Hello world"))
```

Output: 2

_____

6. Convert CamelCase string to snake_case

```
import re

def camel_to_snake(input_str):
    return re.sub(r'(?<!^)(?=[A-Z])', '_', input_str).lower()

print(camel_to_snake("CamelCaseString"))
```

Output: "camel_case_string"

---

7. Convert the first letter of each word to uppercase

```
def capitalize_words(input_str):
    return ' '.join(word.capitalize() for word in input_str.split())

print(capitalize_words("hello world"))
```

Output: "Hello World"

---

8. Count the frequency of a substring within a string

```
def count_substring(input_str, substring):
    return input_str.count(substring)

print(count_substring("hello world hello", "hello"))
```

Output: 2

---

9. Swap the case of each character in a string

```
def swap_case(input_str):
    return input_str.swapcase()

print(swap_case("Hello World"))
```

Output: "hELLO wORLD"

---

10. Remove all whitespace characters from a string

```
def remove_whitespace(input_str):
    return ''.join(input_str.split())

print(remove_whitespace("Hello World"))
```

Output: "HelloWorld"

****

slicing

_____

11. Extract even and odd indexed elements from a list using slicing

```
def extract_even_odd(input_list):
    even_elements = input_list[::2]
    odd_elements = input_list[1::2]
    return even_elements, odd_elements

print(extract_even_odd([1, 2, 3, 4, 5]))
```

Output: ([1, 3, 5], [2, 4])

_____

12. Reverse a given list using slicing

```
def reverse_list(input_list):
    return input_list[::-1]

print(reverse_list([1, 2, 3, 4, 5]))
```

Output: [5, 4, 3, 2, 1]

_____

13. Extract a sublist from a given list using slicing

```
def extract_sublist(input_list, start, end):
    return input_list[start:end]

print(extract_sublist([1, 2, 3, 4, 5], 1, 4))
```

Output: [2, 3, 4]

_____

14. Extract alternate elements from a given list using slicing

```
def extract_alternate(input_list):
    return input_list[::2]

print(extract_alternate([1, 2, 3, 4, 5]))
```

Output: [1, 3, 5]

_____

15. Replace a sublist within a list with another sublist using slicing

```
def replace_sublist(main_list, sublist, start, end):
    main_list[start:end] = sublist
    return main_list

print(replace_sublist([1, 2, 3, 4, 5], ['a', 'b'], 1, 4))
```

Output: [1, 'a', 'b', 5]

****

Functions

_____

16. Compute the factorial of a given number recursively

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

print(factorial(5))
```

Output: 120

_____

17. Generate multiplication table of given number by the user

```
def multiplication_table(num):
    for i in range(1, 11):
        print(f"{num} x {i} = {num * i}")

multiplication_table(7)
```

Output:

```
# 7 x 1 = 7
# 7 x 2 = 14
# 7 x 3 = 21
# 7 x 4 = 28
# 7 x 5 = 35
# 7 x 6 = 42
# 7 x 7 = 49
# 7 x 8 = 56
# 7 x 9 = 63
# 7 x 10 = 70
```

_____

18. Check if a given number is prime or not prime

```
def is_prime(num):
    if num <= 1:
        return False
    elif num <= 3:
        return True
    elif num % 2 == 0 or num % 3 == 0:
        return False
    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True
```

```
print(is_prime(7))
```
Output: True

---

19. Generate the Fibonacci sequence up to a specified number of terms

```
def fibonacci(n):
    sequence = [0, 1]
    while len(sequence) < n:
        next_number = sequence[-1] + sequence[-2]
        sequence.append(next_number)
    return sequence

print(fibonacci(10))
```
Output: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

---

20. Convert temperature from Celsius to Fahrenheit and vice versa

```
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5/9

print(celsius_to_fahrenheit(0))
print(fahrenheit_to_celsius(32))
```

Output: 0.0
Output: 32.0

****

Lists

---

21. Calculate the sum of all elements in a list

```
def sum_of_elements(input_list):
    return sum(input_list)

print(sum_of_elements([1, 2, 3, 4, 5]))
```
Output: 15

---

22. Remove duplicates from a list

```
def remove_duplicates(input_list):
    return list(set(input_list))

print(remove_duplicates([1, 2, 2, 3, 3, 4, 5, 5]))
```
Output: [1, 2, 3, 4, 5]

_____
23. Remove duplicates from a list without changing the order
```
def remove_duplicates_ordered(input_list):
    seen = set()
    output = []
    for item in input_list:
        if item not in seen:
            output.append(item)
            seen.add(item)
    return output

print(remove_duplicates_ordered([1, 2, 2, 3, 3, 4, 5, 5]))
```
Output: [1, 2, 3, 4, 5]

_____
24. Generate a list of squares of numbers from 1 to N using list comprehension
```
def squares_list(n):
    return [i**2 for i in range(1, n+1)]

print(squares_list(5))
```
Output: [1, 4, 9, 16, 25]

_____
25. Sort a list of strings in alphabetical order
```
def sort_strings(input_list):
    return sorted(input_list)

print(sort_strings(["apple", "banana", "orange", "grape"]))
```
Output: ['apple', 'banana', 'grape', 'orange']

****
dictionary

_____
26. Merge two dictionaries
```
def merge_dicts(dict1, dict2):
    return {**dict1, **dict2}

dict1 = {'a': 1, 'b': 2}
dict2 = {'c': 3, 'd': 4}
print(merge_dicts(dict1, dict2))
```
Output: {'a': 1, 'b': 2, 'c': 3, 'd': 4}

_____
27. Access the value associated with a given key in a dictionary
```
def access_value(dictionary, key):
    return dictionary.get(key)
```

```
dictionary = {'a': 1, 'b': 2, 'c': 3}
print(access_value(dictionary, 'b'))
Output: 2
```

---

28. Get all keys from a dictionary

```
def get_keys(dictionary):
    return list(dictionary.keys())

dictionary = {'a': 1, 'b': 2, 'c': 3}
print(get_keys(dictionary))
Output: ['a', 'b', 'c']
```

---

29. Update a dictionary with key-value pairs from another dictionary

```
def update_dictionary(dictionary1, dictionary2):
    dictionary1.update(dictionary2)
    return dictionary1

dictionary1 = {'a': 1, 'b': 2}
dictionary2 = {'c': 3, 'd': 4}
print(update_dictionary(dictionary1, dictionary2))
Output: {'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

---

30. Create a dictionary with keys as numbers from 1 to N and values as squares of the keys

```
def squares_dictionary(n):
    return {i: i**2 for i in range(1, n+1)}

print(squares_dictionary(5))
Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

****

ASSIGNMENT-3

---

1. Book class

```
class Book:
    def __init__(self, title, author, no_of_pages):
        self.title = title
        self.author = author
        self.no_of_pages = no_of_pages

    def describe_book(self):
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")
        print(f"Number of Pages: {self.no_of_pages}")
```

```
book1 = Book("Python Programming", "John Smith", "300")
book1.describe_book()
```

2.Student class

```
class Student:
    failed_students = 0

    def __init__(self, first_name, last_name, course, batch_year, result):
        self.first_name = first_name
        self.last_name = last_name
        self.course = course
        self.batch_year = batch_year
        self.result = result

        if result == "FAIL":
            Student.failed_students += 1

    @staticmethod
    def total_failed_students():
        return Student.failed_students

student1 = Student("Alice", "Smith", "Computer Science", "2022", "PASS")
student2 = Student("Bob", "Johnson", "Engineering", "2023", "FAIL")
student3 = Student("Charlie", "Brown", "Mathematics", "2022", "FAIL")

print(Student.total_failed_students())
```

---

3. Surface_Area class with Function Overloading

```
class Surface_Area:
    def calculate_area(self, radius):
        return 3.14 * radius ** 2

    def calculate_area(self, length, width):
        return length * width

    def calculate_area(self, base, height):
        return 0.5 * base * height

area_calculator = Surface_Area()
print("Area of circle:", area_calculator.calculate_area(5))
print("Area of rectangle:", area_calculator.calculate_area(4, 6))
print("Area of triangle:", area_calculator.calculate_area(3, 7))
```

_____

4. Employee class

```
class Employee:
    def __init__(self, emp_id, name, salary):
        self.emp_id = emp_id
        self.name = name
        self.salary = salary

    def calculate_net_salary(self):
        da = 0.15 * self.salary
        ma = 0.05 * self.salary
        hra = 0.20 * self.salary
        pa = 0.12 * self.salary
        net_salary = self.salary + da + ma + hra - pa
        return net_salary

num1 = NUM(3, 4)
num2 = NUM(5, 6)
print(num1 * num2)
```

_____

5. Overloading the multiplication operator

```
class NUM:
    def __init__(self, no1, no2):
        self.no1 = no1
        self.no2 = no2

    def __mul__(self, other):
        return self.no1 * other.no2

employee1 = Employee("EMP001", "John Doe", 5000)
print(employee1.calculate_net_salary())
time1 = Time(3, 45)
time2 = Time(2, 30)
print(time1 + time2)
```

_____

6. Time class with operator overloading

```
class Time:
    def __init__(self, hour, minutes):
        self.hour = hour
        self.minutes = minutes

    def __add__(self, other):
```

```
        total_minutes = self.minutes + other.minutes
        carry = total_minutes // 60
        new_hour = self.hour + other.hour + carry
        new_minutes = total_minutes % 60
        return Time(new_hour, new_minutes)
```

---

7. Bank Account Class with Inheritance

```
class Account:
    def __init__(self, account_no, balance):
        self.account_no = account_no
        self.balance = balance

class BankAccount(Account):
    def deposit(self, amount):
        self.balance += amount

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
            print(f"Withdrawal successful. Remaining balance: {self.balance}")
        else:
            print("Insufficient funds.")

account1 = BankAccount("ACC001", 1000)
account1.deposit(500)
account1.withdraw(200)
```

---

8. CAR abstract class with Maruti subclass

```
from abc import ABC, abstractmethod

class CAR(ABC):
    def __init__(self, model):
        self.model = model

    def concrete_method(self):
        print("Concrete method implementation in CAR class.")

    @abstractmethod
    def abstract_method1(self):
        pass

    @abstractmethod
    def abstract_method2(self):
```

```
        pass

class Maruti(CAR):
    def abstract_method1(self):
        print("Implementation of abstract_method1 in Maruti class.")

    def abstract_method2(self):
        print("Implementation of abstract_method2 in Maruti class.")
```

---

9. Interface Shape

```
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

    @abstractmethod
    def perimeter(self):
        pass

maruti_car = Maruti("Alto")
maruti_car.abstract_method1()
maruti_car.abstract_method2()
```

---

10. Handling ZeroDivisionError exception

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero.")
```