1. Write a python program to input UserName and print welcome message in same line using two print statement.

Code:

```
Name = input("Enter Your Name: ")
print("Welcome", Name)
```

Output :

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\q1 .py"
Enter Your Name: Michal Jecson
Welcome Michal Jecson
```

2. Write a python program to find the area and circumference of a circle.

Code:

```
import math

radius = float(input("Enter Radius: "))

area = math.pi * radius**2
circumference = 2 * math.pi * radius

print(f"Area of circle: {round(area,2)}")
print(f"Circumference of circle: {round(circumference,2)}")
```

Output :

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Enter Radius: 24
Area of circle: 1809.56
Circumference of circle: 150.8
```

3. Write a python program to find the simple interest for the given data.

Code:

```
p=float(input("Enter principle  amount: "))
r=float(input("Enter rate of intrest:"))
t=float(input("Enter time: "))
si=p*r*t/100
print("simple interest  is {}".format(si))
```

Output :

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Enter principle  amount: 50000
Enter rate of intrest:12
Enter time: 15
simple interest  is 90000.0
```

4. Write a python program to swap the content of two variables using third variable .

Code:

```
x = 5
y = 10

temp = x
x = y
y = temp

print("X after swapping: ", x)
print("Y after swapping: ", y)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
X after swapping:  10
Y after swapping:  5
```

5. Write a python program to swap the content of three variables without using third variable

Code :

```
a, b, c = 1, 2, 3
a = a + b + c
b = a - (b + c)
c = a - (b + c)
a = a - (b + c)

print("After swapping:")
print("a = ", a)
print("b = ", b)
print("c = ", c)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
After swapping:
a =  3
b =  1
c =  2
```

6. Write a python program to check whether given year is leap year or not.

Code:

```
def check_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
```

```
year = int(input("Enter a year: "))
if check_leap_year(year):
    print(year, " is a leap year.")
else:
    print(year, " is not a leap year.")
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Enter a year: 25
25  is not a leap year.
```

7. Write a python program to convert Fahrenheit to Centigrade.

Code :

```
def fahrenheit_to_centigrade(fahrenheit):
    return (fahrenheit - 32) * 5.0/9.0

# Input temperature in Fahrenheit
fahrenheit_temp = 98.6

# Convert Fahrenheit to Centigrade
centigrade_temp = fahrenheit_to_centigrade(fahrenheit_temp)

print(f"{fahrenheit_temp} Fahrenheit is equal to {centigrade_temp} Centigrade")
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
98.6 Fahrenheit is equal to 37.0 Centigrade
```

8. Write a python program to accept two integers from user and display addition, Subtraction, Multiplication, Division of two integers.

Code:

```
a = int(input("Enter the first integer: "))
b = int(input("Enter the second integer: "))

add = a + b
sub = a - b
multi = a * b
div = a / b
print("Addition: ", add)
print("Subtraction: ", sub)
print("Multiplication: ", multi)
print("Division: ", div)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Enter the first integer: 23
Enter the second integer: 15
Addition:  38
Subtraction:  8
Multiplication:  345
Division:  1.5333333333333334
```

9. Write a python program to find maximum no from 3 number using if, if..elif and nested if.

Code:

```
# Using if statements
def find_max_if(a, b, c):
    if a >= b and a >= c:
        return a
    if b >= a and b >= c:
        return b
    return c

# Using if...elif statements
def find_max_elif(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

# Using nested if statements
def find_max_nested_if(a, b, c):
    if a >= b:
        if a >= c:
            return a
        else:
            return c
    else:
        if b >= c:
            return b
        else:
            return c
# Test the functions
num1 = 10
num2 = 20
num3 = 15
```

```
print("Using if statements: ", find_max_if(num1, num2, num3))
print("Using if...elif statements: ", find_max_elif(num1, num2, num3))
print("Using nested if statements: ", find_max_nested_if(num1, num2, num3))
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Using if statements:  20
Using if...elif statements:  20
Using nested if statements:  20
```

10. Program to print full pyramid using .

Code:

```
def full_pyramid(n):
    for i in range(1, n + 1):
        # Print leading spaces
        for j in range(n - i):
            print(" ", end="")

        # Print asterisks for the current row
        for k in range(1, 2*i):
            print("*", end="")
        print()
full_pyramid(5)
```

Output :

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
    *
   ***
  *****
 *******
*********
```

11. Write a python program to print the factorial of a inputted number.(Ex. N= 5 O/p: 5*4*3*2*1=120)

Code:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
num = int(input("Enter a number: "))
if num < 0:
    print("Factorial does not exist for negative numbers.")
elif num == 0:
```

```
    print("The factorial of 0 is 1")
else:
    print(f"The factorial of {num} is {factorial(num)}")
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Enter a number: 6
The factorial of 6 is 720
```

12. Write a python program to print the Fibonacci series up to a given number.

Code :

```
def fibonacci_series(n):
    a, b = 0, 1
    count = 0

    if n <= 0:
        print("Please enter a positive integer.")
    elif n == 1:
        print("Fibonacci series up to", n, ":")
        print(a)
    else:
        print("Fibonacci series up to", n, ":")
        while count < n:
            print(a, end=" ")
            nth = a + b
            a  =  b
            b = nth
            count += 1

n = int(input("Enter the number of terms: "))
fibonacci_series(n)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Enter the number of terms: 12
Fibonacci series up to 12 :
0 1 1 2 3 5 8 13 21 34 55 89
```

13. Write a python program to print multiplication tables of given range.

Code:

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

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

# Print multiplication tables from 2 to 5
multiplication_tables(2, 4)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\q13.py"
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

14. Write program to check whether the given number is palindrome or not.

Code:

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

```
number = 12321

if is_palindrome(number):
    print(f"{number} is a palindrome.")
else:
    print(f"{number} is not a palindrome.")
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
12321 is a palindrome.
```

15. Write program to print the Pascal triangle.

Code:

```
def generate_pascals_triangle(num_rows):
    triangle = []
    for i in range(num_rows):
        row = [None for _ in range(i + 1)]
        row[0], row[-1] = 1, 1
        for j in range(1, len(row) - 1):
            row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j]
        triangle.append(row)
    return triangle
def print_pascals_triangle(triangle):
    for row in triangle:
        print(" ".join(map(str, row)).center(len(triangle[-1]) * 2))

num_rows = 5
triangle = generate_pascals_triangle(num_rows)
print_pascals_triangle(triangle)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

16. Write program to find the Armstrong numbers between 100 and 1000.

Code:

```
for num in range(100, 1000):

    order = len(str(num))
    sum = 0
```

```
    temp = num
    while temp > 0:


        digit = temp % 10
        sum += digit ** order
        temp //= 10
    if num == sum:
        print(num)
```

Output :

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
153
370
371
407
```

17. Write a python program to input student details (Roll No, Name , 3 Subjects Marks) and Display Result Sheet with proper format.

Code :

```
# Input student details
roll_no = input("Enter Roll No: ")
name = input("Enter Name: ")
subject1 = float(input("Enter Marks for Subject 1: "))
subject2 = float(input("Enter Marks for Subject 2: "))
subject3 = float(input("Enter Marks for Subject 3: "))

# Calculate total marks and percentage
total_marks = subject1 + subject2 + subject3
percentage = (total_marks / 300) * 100

# Display result sheet
print("\n-------- Result Sheet---------")
print("Roll No:", roll_no)
print("Name:", name)
print("\nSubject 1:", subject1)
print("Subject 2:", subject2)
print("Subject 3:", subject3)
print("\nTotal Marks:", total_marks)
print("Percentage:", percentage)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\q17.py"
Enter Roll No: 1234
Enter Name: Michal Jecson
Enter Marks for Subject 1: 43
Enter Marks for Subject 2: 56
Enter Marks for Subject 3: 67

-------- Result Sheet --------
Roll No: 1234
Name: Michal Jecson

Subject 1: 43.0
Subject 2: 56.0
Subject 3: 67.0

Total Marks: 166.0
Percentage: 55.333333333333336
```

18. Write a python program to check whether inputted string is palindrome or not.

Code:

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

input_string = input("Enter a string: ")
if is_palindrome(input_string):
    print("The inputted string is a palindrome.")
else:
    print("The inputted string is not a palindrome.")
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Enter a string: Hello
The inputted string is not a palindrome.
```

19. Write a python program to count No of Uppercase Characters, Lowercase Characters, Digits, Special Characters and Space from the sentence.

Code:

```
def count_characters(sentence):
    uppercase = 0
    lowercase = 0
    digits = 0
    special_chars = 0
    spaces = 0
```

```
    for char in sentence:
        if char.isupper():
            uppercase += 1
        elif char.islower():
            lowercase += 1
        elif char.isdigit():
            digits += 1
        elif char.isspace():
            spaces += 1
        else:
            special_chars += 1

    return uppercase, lowercase, digits, special_chars, spaces

sentence = "Hello, World! 123"
uppercase, lowercase, digits, special_chars, spaces = count_characters(sentence)

print("Uppercase:", uppercase)
print("Lowercase:", lowercase)
print("Digits:", digits)
print("Special Characters:", special_chars)
print("Spaces:", spaces)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Uppercase: 2
Lowercase: 8
Digits: 3
Special Characters: 2
Spaces: 2
```

20. Write a python program to convert sentence in toggle case without using in-built function.

Code:

```
def toggle_case(sentence):
    toggled_sentence = ''
    for char in sentence:
        if char.islower():
            toggled_sentence += char.upper()
        else:
            toggled_sentence += char.lower()
    return toggled_sentence

# Test the function
input_sentence = "Hello, World!"
toggled_output = toggle_case(input_sentence)
```

```
print(toggled_output)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
hELLO, wORLD!
```

21. Write a python program to display each words along with it's total no of characters and at the end display word which have maximum character.

Code:

```
def display_words_with_char_count(words):
    words_dict = {}
    max_word = ""
    max_char_count = 0

    for word in words:
        char_count = len(word)
        words_dict[word] = char_count

        if char_count > max_char_count:
            max_char_count = char_count
            max_word = word

    for word, char_count in words_dict.items():
        print(f"{word} - {char_count} characters")

    print(f"The word with the maximum characters is: {max_word}")

# Example Usage
words_list = ["apple", "banana", "orange", "strawberry", "kiwi"]
display_words_with_char_count(words_list)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
apple - 5 characters
banana - 6 characters
orange - 6 characters
strawberry - 10 characters
kiwi - 4 characters
The word with the maximum characters is: strawberry
```

22. Write a python program to enter a line and reverse a words from line which have inputted by user.
Ex. String is: : hello girls hello boys hello All!! Enter Word:hello.
O/P: olleh girls olleh boys olleh All!!/
Code:

```
def reverse_words_in_line():
    line = input("Enter a line: ")
    word = input("Enter Word: ")
    words = line.split()
    reversed_line = ' '.join([w[::-1] if w == word else w for w in words])
    print("Output: ", reversed_line)

reverse_words_in_line()
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\q22.py"
Enter a line: hello girls hello boys hello All!!
Enter Word: hello
Output:  olleh girls olleh boys olleh All!!
```

23. Write a program that will calculate summation of numbers stored at even locations and summation of numbers stored at odd locations in list with 10 elements.

Code:

```
n = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

even = sum(n[1::2])
odd = sum(n[::2])

print("Sum of n at even locations:", even)
print("Sum of n at odd locations:", odd)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
Sum of n at even locations: 30
Sum of n at odd locations: 25
```

24. Write a python program to create menu driven program of list which perform insert, update, delete, search, sorting and display element in or from list. Also provide nested menu for delete, sorting.

Code :

```
def insert_element(lst):
    element = input("Enter the element to insert: ")
    lst.append(element)
    print("Element inserted successfully.")

def update_element(lst):
    index = int(input("Enter the index of the element to update: "))
    if 0 <= index < len(lst):
        new_element = input("Enter the new element: ")
        lst[index] = new_element
```

```
        print("Element updated successfully.")
    else:
        print("Invalid index.") def
    delete_element(lst):
    element = input("Enter the element to delete: ")
    if element in lst:
        lst.remove(element)
        print("Element deleted successfully.")
    else:
        print("Element not found in the list.")

def search_element(lst):
    element = input("Enter the element to search: ")
    if element in lst:
        print("Element found in the list.")
    else:
        print("Element not found in the list.")

def sort_list(lst):
    lst.sort()
    print("List sorted successfully.")

def display_list(lst):
    print("Elements in the list:")
    for element in lst:
        print(element)

def main():
    my_list = []
    while True:
        print("\nMenu:")
        print("1. Insert Element")
        print("2. Update Element")
        print("3. Delete Element")
        print("4. Search Element")
        print("5. Sort List")
        print("6. Display List")
        print("7. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            insert_element(my_list)
        elif choice == '2':
            update_element(my_list)
```

```
        elif choice == '3':
            delete_element(my_list)
        elif choice == '4':
            search_element(my_list)
        elif choice == '5':
            sort_list(my_list)
        elif choice == '6':
            display_list(my_list)
        elif choice == '7':
            break
        else:
            print("Invalid choice. Please try again.")

if  name  == "  main  ":
    main()
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"

  Menu:
  1. Insert Element
  2. Update Element
  3. Delete Element
  4. Search Element
  5. Sort List
  6. Display List
  7. Exit
  Enter your choice: 1
  Enter the element to insert: Hello
  Element inserted successfully.
```

25. Write a python program that reads a string and then prints a string that capitalizes every other letter in the string.
(Ex. Enter String: python O/p: pYtHoN)
Code:

```
def capitalize_every_other_letter(input_string):
    result = ""
    for i in range(len(input_string)):
        if i % 2 == 1:
            result += input_string[i].upper()
        else:
            result += input_string[i]
    return result

# Input from user
```

```
input_string = input("Enter a string: ")

# Capitalize every other letter
output_string = capitalize_every_other_letter(input_string)

# Output
print("Output:", output_string)
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\q25.py"
Enter a string: python
Output: pYtHoN
```

26. Write a python program to find minimum element from a list of element along with its index in the list.

Code:

```
def find_min_element_with_index(lst):
    min_element = min(lst)
    min_index = lst.index(min_element)
    return min_element, min_index

# Test the function
elements = [10, 5, 8, 3, 15, 7]
min_element, min_index = find_min_element_with_index(elements)
print(f"The minimum element is {min_element} at index {min_index}.")
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
The minimum element is 3 at index 3.
```

27. Write a python program to count frequency of a given element in a list of numbers.

Code:

```
def count_frequency(lst, element):
    return lst.count(element)

# Example Usage
numbers = [1, 2, 3, 4, 2, 2, 3, 1, 4, 5]
element_to_count = 2
frequency = count_frequency(numbers, element_to_count)
print(f'The frequency of {element_to_count} in the list is: {frequency}')
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"
The frequency of 2 in the list is: 3
```

28. Write a python program to create menu driven program of dictionary which perform insert, update, delete, and display element in or from dictionary. Also provide nested menu for delete element.

Code:

```
def insert_element(dictionary):
    key = input("Enter key: ")
    value = input("Enter value: ")
    dictionary[key] = value
    print("Element inserted successfully!")

def update_element(dictionary):
    key = input("Enter key to update: ")
    if key in dictionary:
        value = input("Enter new value: ")
        dictionary[key] =  value
        print("Element updated successfully!")
    else:
        print("Key not found!")

def delete_element(dictionary):
    key = input("Enter key to delete: ")
    if key in dictionary:
        del dictionary[key]
        print("Element deleted successfully!")
    else:
        print("Key not found!")

def display_elements(dictionary):
    print("Dictionary Elements:")
    for key, value in dictionary.items():
        print(f"{key}: {value}")

dictionary = {}

while True:
    print("\nMenu:")
    print("1. Insert Element")
    print("2. Update Element")
    print("3. Delete Element")
    print("4. Display Elements")
    print("5. Exit")

    choice = input("Enter your choice: ")

    if choice == '1':
        insert_element(dictionary)
```

```
    elif choice == '2':
        update_element(dictionary)
    elif choice == '3':
        delete_element(dictionary)
    elif choice == '4':
        display_elements(dictionary)
    elif choice == '5':
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\tempCodeRunnerFile.py"

  Menu:
  1. Insert Element
  2. Update Element
  3. Delete Element
  4. Display Elements
  5. Exit
  Enter your choice: 1
  Enter key: 2
  Enter value: Nice
  Element inserted successfully!
```

29. Create a dictionary whose keys are month names and values are the number of days in the corresponding months. (a) Ask the user to enter a month name and use the dictionary to tell how many days are in the month.
(b) Print out all the keys in the alphabetical order.
(c) Print out all the months with 31 days.
(d) Print Out the (key-value) pairs sorted by the number of days in each month.

Code:

```
months_dict = {
    "January": 31,
    "February": 28,
    "March": 31,
    "April": 30,
    "May": 31,
    "June": 30,
    "July": 31,
    "August": 31,
    "September": 30,
    "October": 31,
```

```
    "November": 30,
    "December": 31
}

# Ask user for a month name and display the number of days in that month
user_month = input("Enter a month name: ")
print(f"{user_month} has {months_dict.get(user_month, 'unknown')} days")

# Print all keys in alphabetical order
print("Months in alphabetical order:")
for month in sorted(months_dict.keys()):
    print(month)

# Print months with 31 days
print("Months with 31 days:")
for month, days in months_dict.items():
    if days == 31:
        print(month)

# Print key-value pairs sorted by the number of days
print("Months sorted by days:")
sorted_months = sorted(months_dict.items(), key=lambda x: x[1])
for month, days in sorted_months:
    print(f"{month}: {days} days")
```

Output:

```
PS D:\Python\ProblemSheet-1> python -u "d:\Python\ProblemSheet-1\q29.py"
Enter a month name: April
April has 30 days
Months in alphabetical order:
April
August
December
February
January
July
June
March
May
November
October
September
Months with 31 days:
January
March
May
July
August
October
December
Months sorted by days:
February: 28 days
April: 30 days
June: 30 days
September: 30 days
November: 30 days
January: 31 days
March: 31 days
May: 31 days
July: 31 days
August: 31 days
October: 31 days
December: 31 days
```