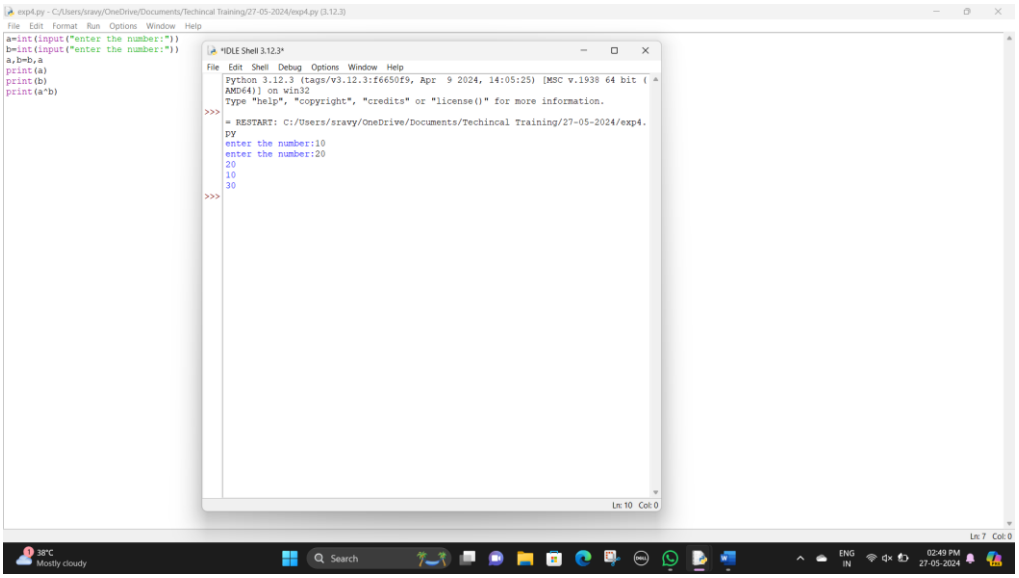
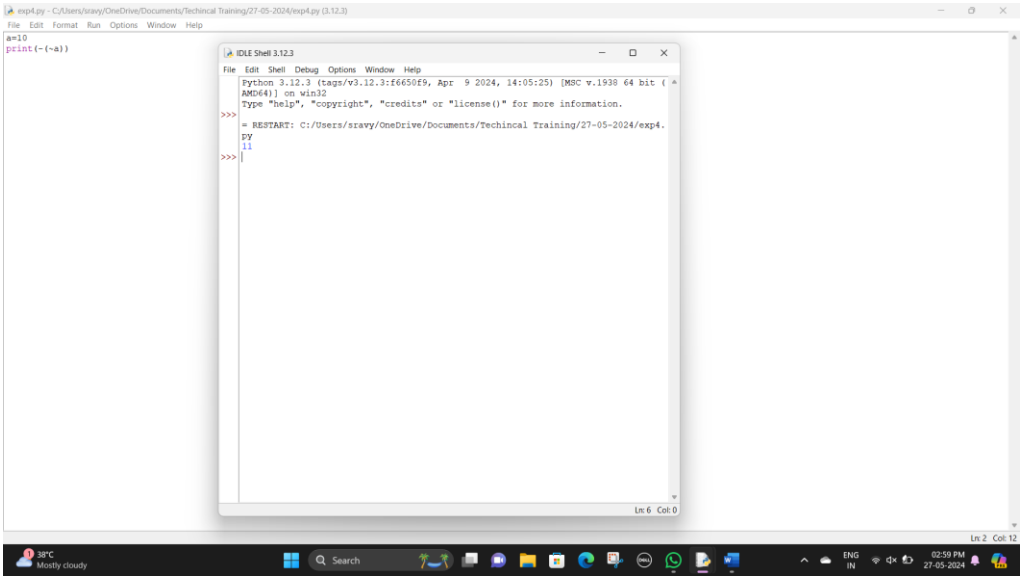
PYTHON PROGRAMMING

Dt: 27-05-2024

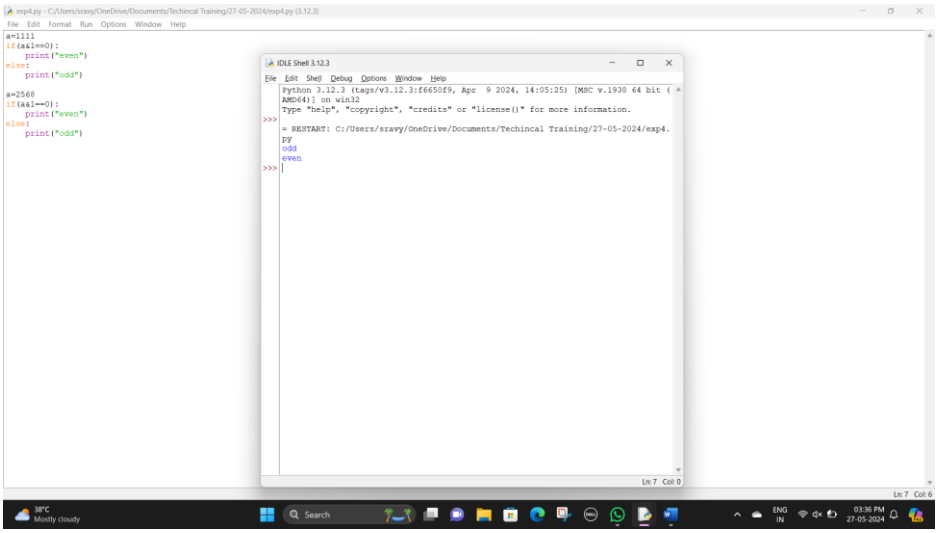
1. Swap 2 numbers using XNOR.



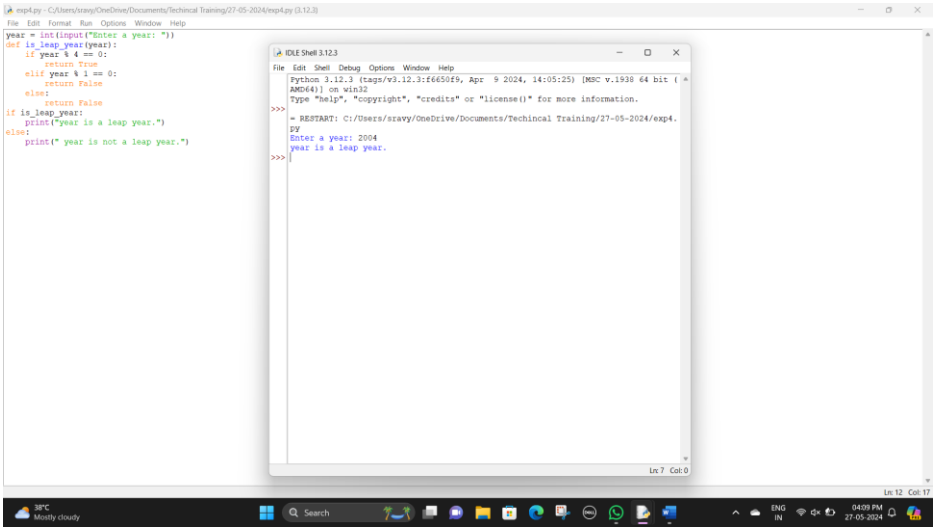
2. Given input is 10 should get output as 11 using NOT operators.



3. **Check whether the given number is odd or even using Bitwise operators.**



4. **Write a program whether the year is leap year or not.**

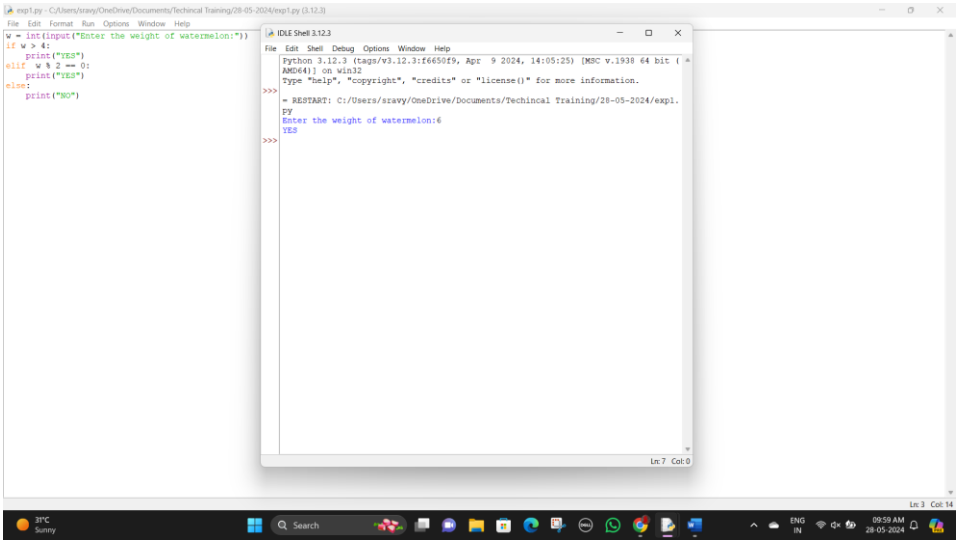


Dt:28-05-2024

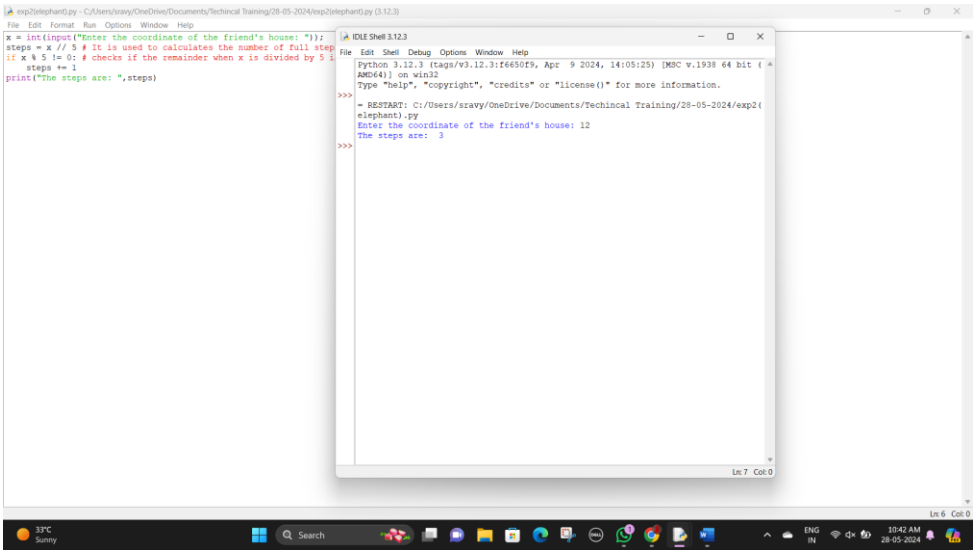
CODEFORCE

1. **One hot summer day Pete and his friend Billy decided to buy a watermelon. They chose the biggest and the ripest one, in their opinion. After that the watermelon was weighed, and the scales showed w kilos. They rushed home, dying of thirst, and decided to divide the berry, however they faced a hard problem.**
   Pete and Billy are great fans of even numbers, that's why they want to divide the watermelon in such a way that each of the two parts weighs even number of kilos, at the same time it is not obligatory that the parts are equal. The boys are extremely tired and want to start their meal as soon as possible, that's why you should help them and find out, if they can divide the watermelon in the way they want. For sure, each of them should get a part of positive weight.

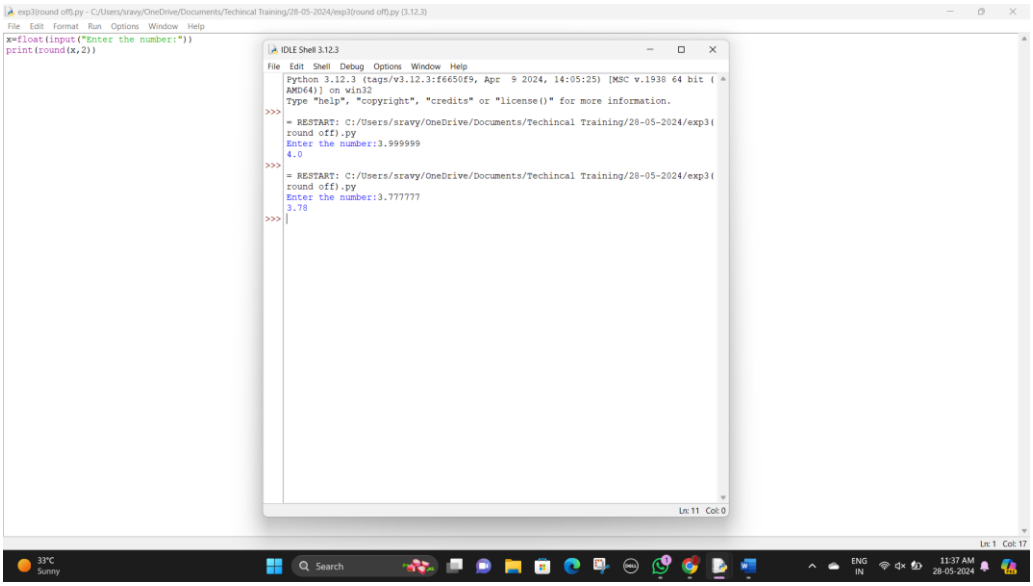
Input: The first (and the only) input line contains integer number w (1 ≤ w ≤ 100) — the weight of the watermelon bought by the boys.
Output: Print YES, if the boys can divide the watermelon into two parts, each of them weighing even number of kilos; and NO in the opposite case.



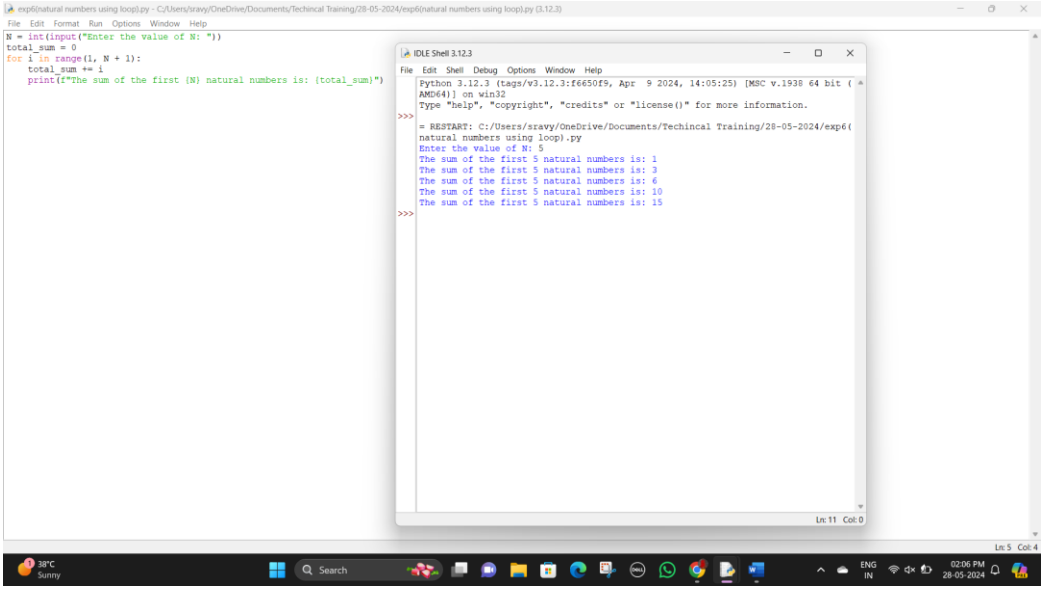
2. **An elephant decided to visit his friend. It turned out that the elephant's house is located at point 0 and his friend's house is located at point x(x > 0) of the coordinate line. In one step the elephant can move 1, 2, 3, 4 or 5 positions forward. Determine, what is the minimum number of steps he need to make in order to get to his friend's house.**
Input: The first line of the input contains an integer x (1 ≤ x ≤ 1 000 000) — The coordinate of the friend's house.
Output: Print the minimum number of steps that elephant needs to make to get from point 0 to point x.



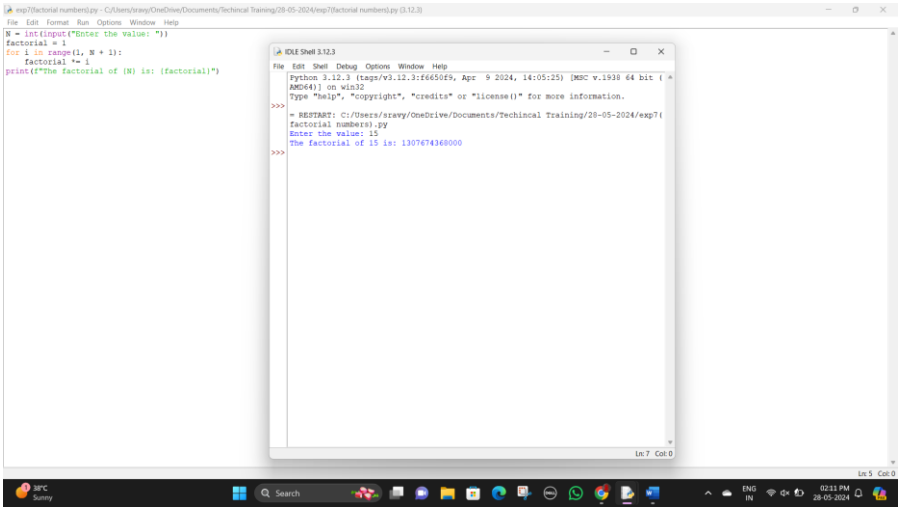
3. **Round-off:**



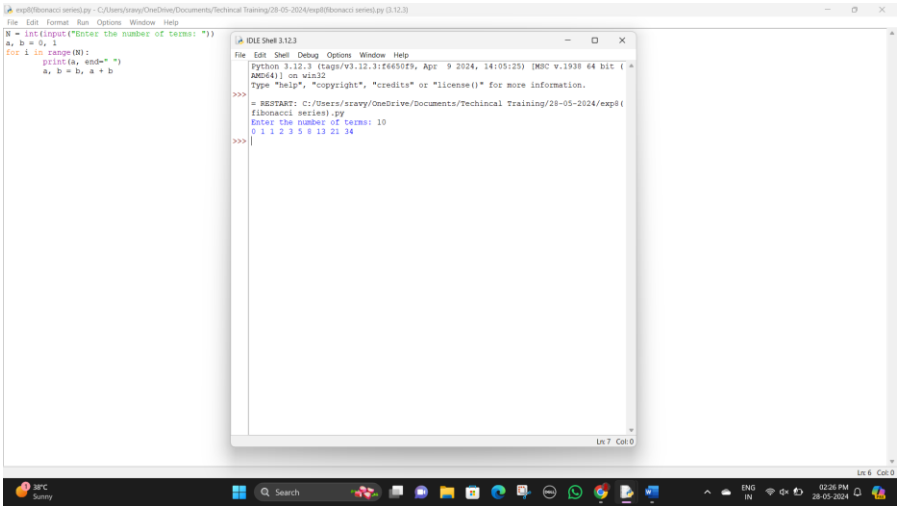
4. **Write a python program to find sum of the first N natural numbers using a loop.**



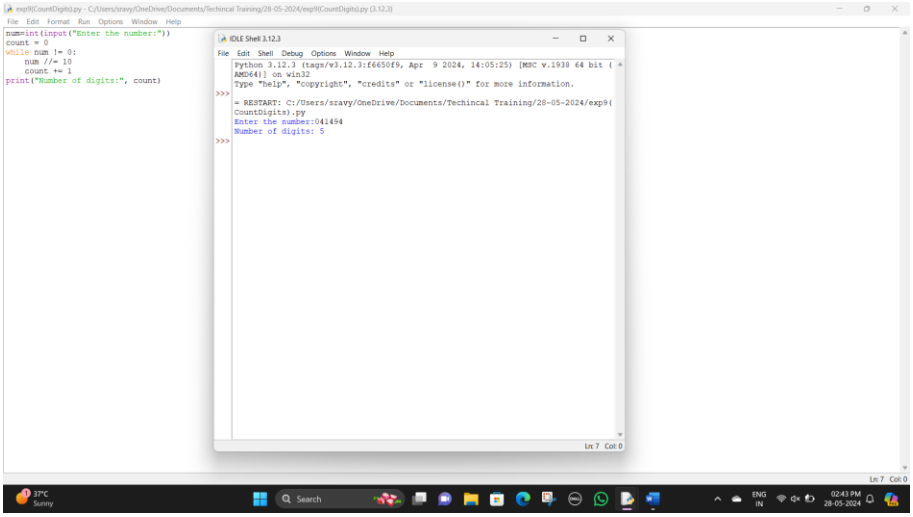
5. **Write a python program to find the factorial of a given number using a loop.**



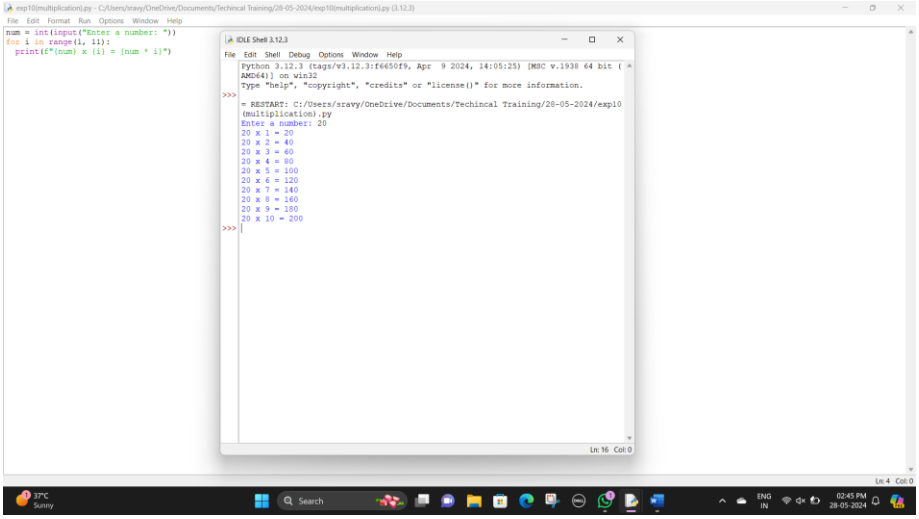
6. **Write a python program to print the Fibonacci series up to N terms.**



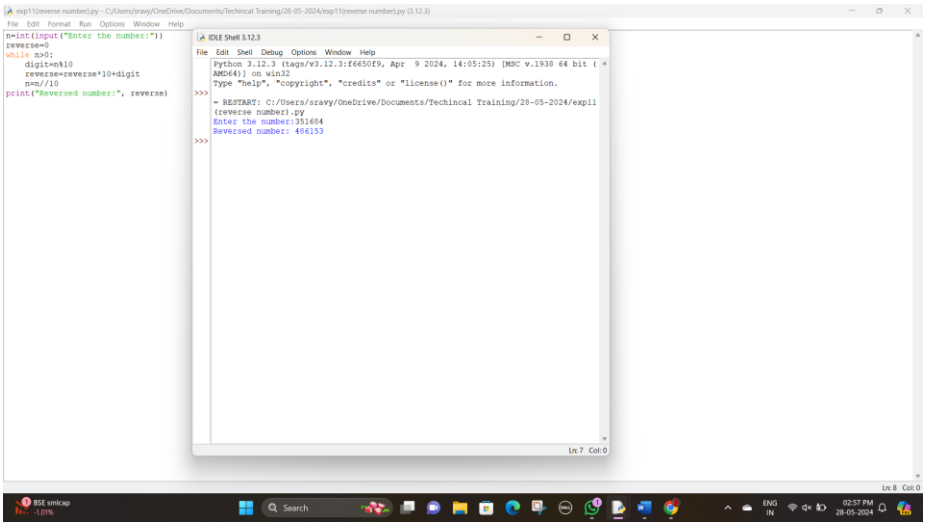
7. **Write a python program to count the number of digits in a given numbers.**



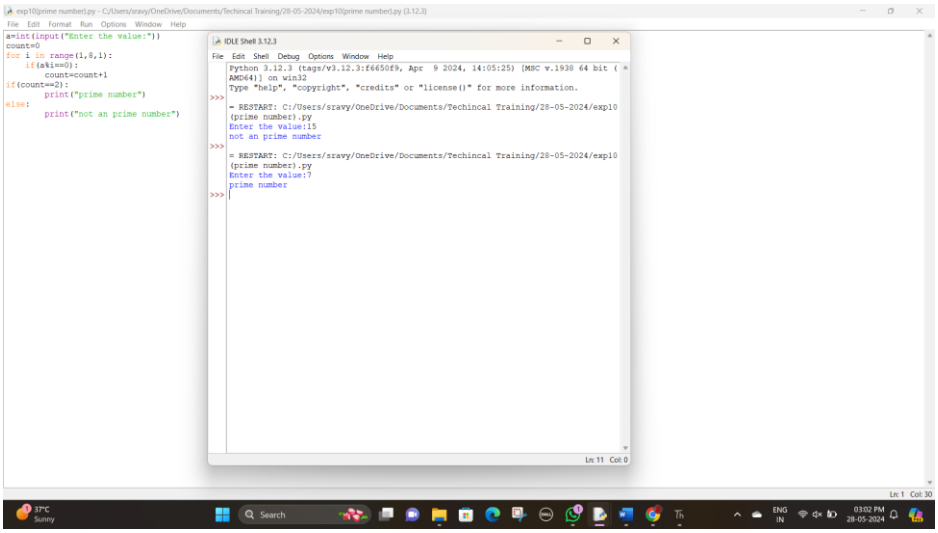
8. **Write a python program to print the multiplication table of a given number.**



9. **Write a python program to reverse a given number.**



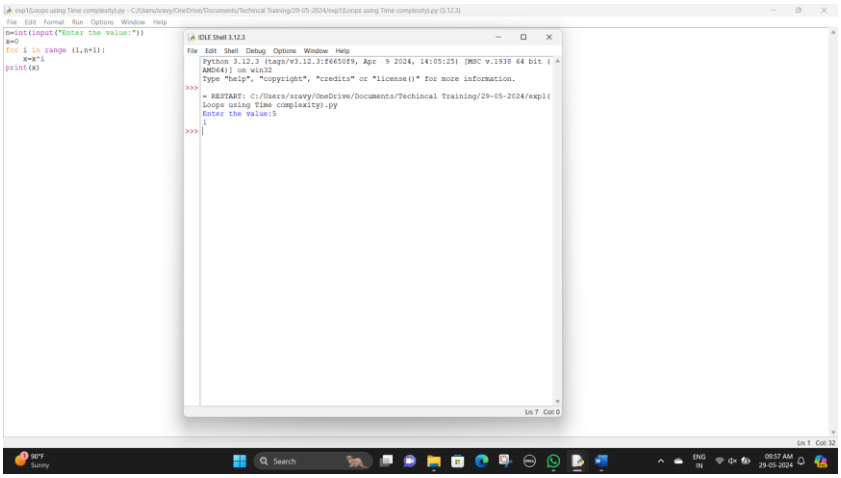
10. Write a python program to check if a given number is prime or not.

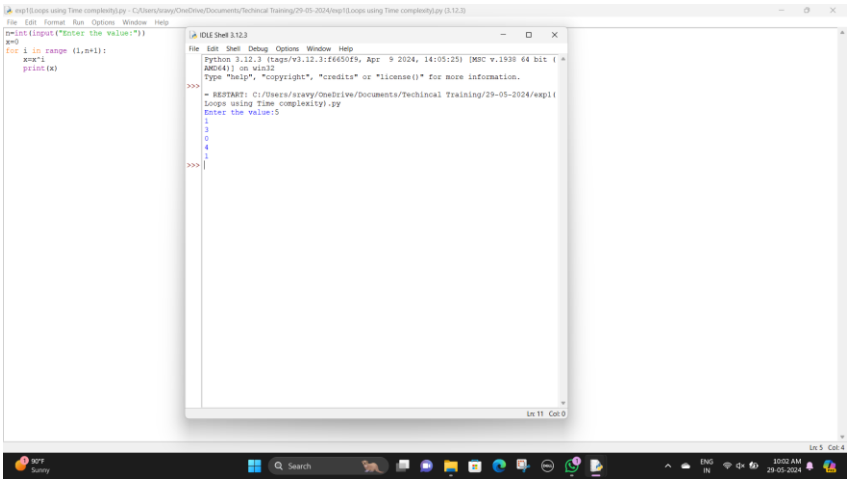


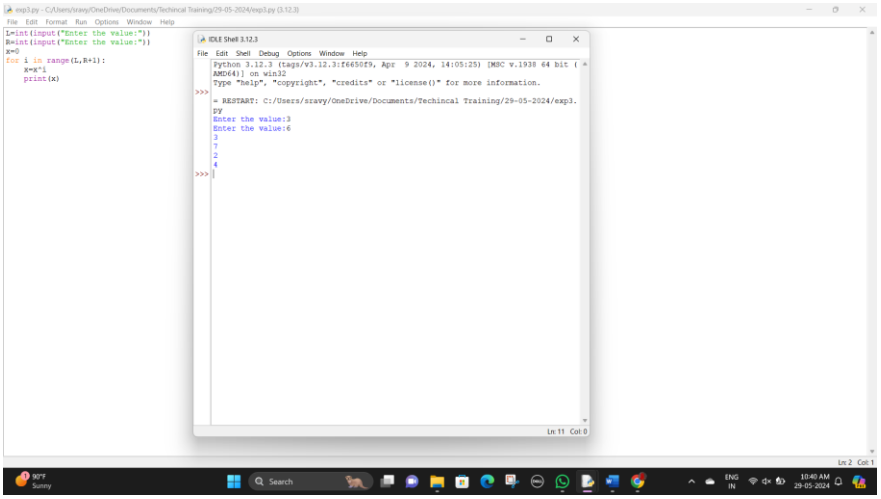
Dt:29-05-2024

LOOPS

Loops using Time Complexity with XOR

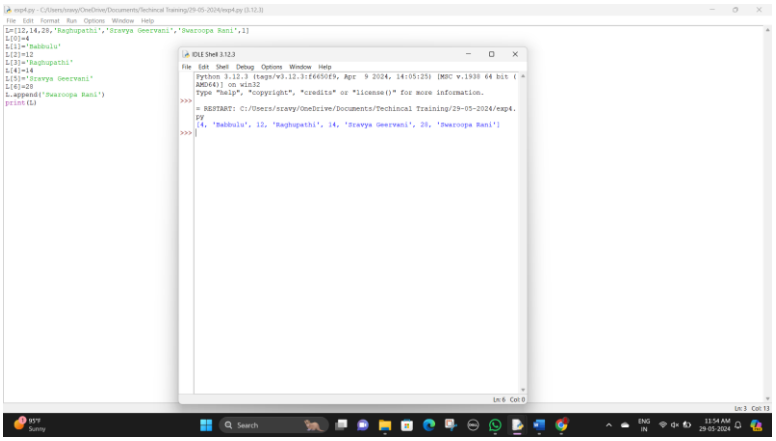




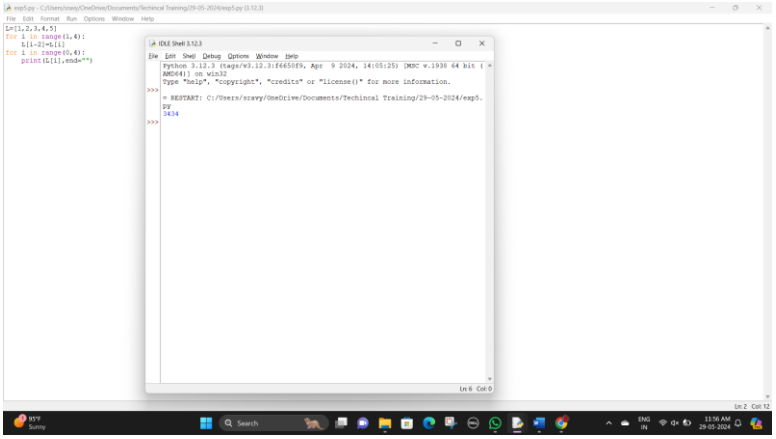


Programs related to List using Loops

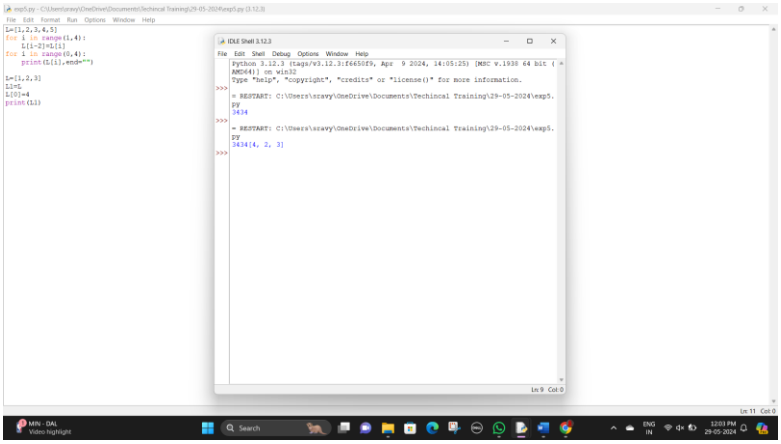
1.



2.



3.



LIST METHODS:

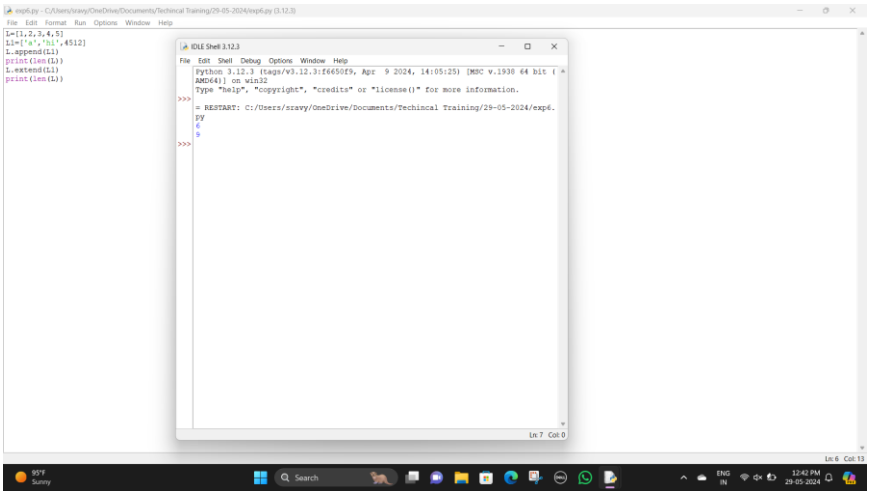
- **Append():** Adding element at the end of the list.
- **Extend():** Adding no. of elements to the end of the list.
- **Insert():** Inserting an item at the define index**. [Syntax:**{L.insert(2,'program')}**]**
- **Remove():** Removing an item by its value from the list**. [Syntax:**{L.remove()}**]**
- **Pop():** Removes the last item from the list or removes an item by its index from the list.
- **Clear():** Removes all items from the list.

- **Index():** Returns the index of the frist matched item. **[Syntax:**{print(L.index())}**]**
- **Count():** Returns the count of number of items passed as an argument. **[Syntax:**{print(L.count())}**]**
- **Copy():** Returns a copy of list. **[Syntax:**{L!=L.copy()}**]**
- **Sort():** Sort items in a list in ascending order. **[Syntax:**{L.sort()}]
- **Reverse():** Reverse the order of items in the list. [**Syntax:**{L>reverse}]

LIST BUILT-IN FUNCTIONS:

- **Sum():** Sums up the numbers in the list.
- **Max():** Return maximum element of given.
- **Min():** Return minimum element of given list.
- **All():** Returns true if all elements are true or if list is empty.
- **Any():** Returns true if any elements are true. If list is empty, return false.
- **Len():** Returns length of the list or size of the list.

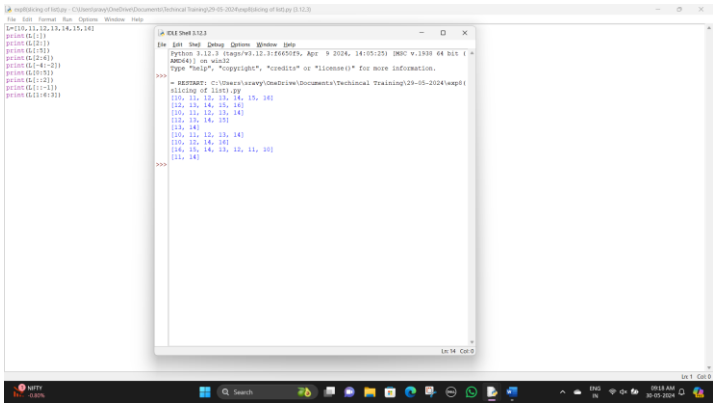
4.



OPERATIONS ON LIST:

1. Concatenation of list (+)
2. Repetition of list (*)
3. Membership on list (in, not in)

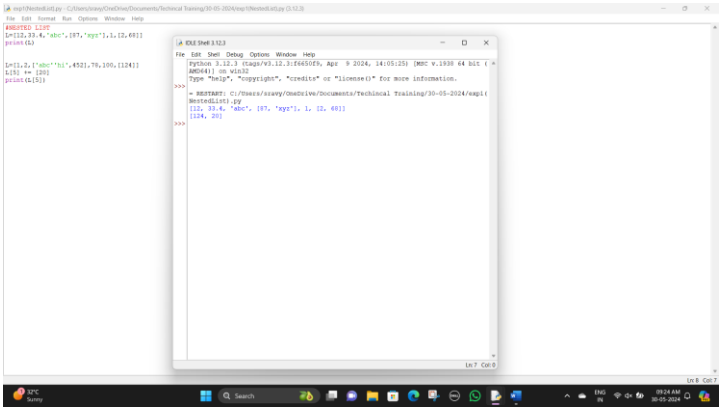
SLICING OF LIST:



Dt:30-05-2024

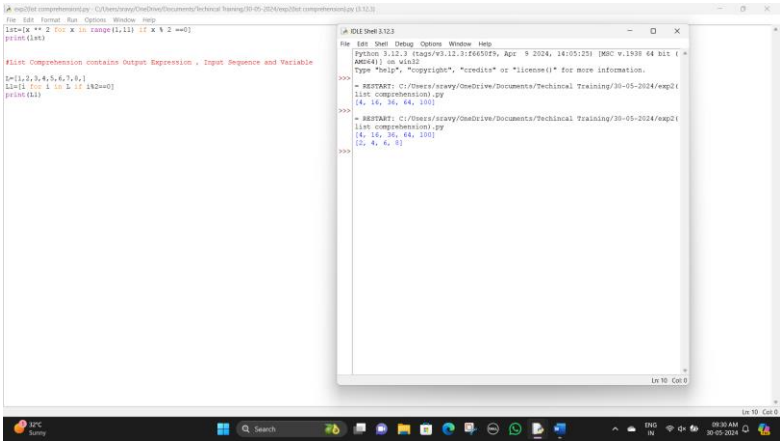
NESTED LIST

- List inside another list is Nested List.
- Nested List is also called as Multidimensional List.

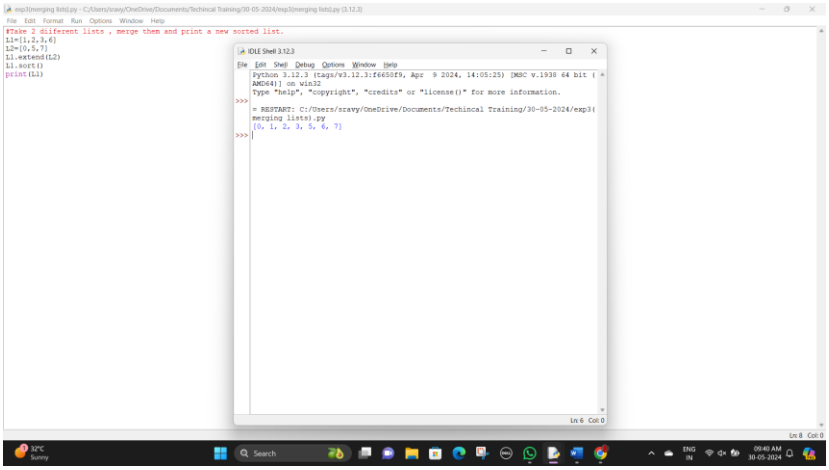


List Comprehension

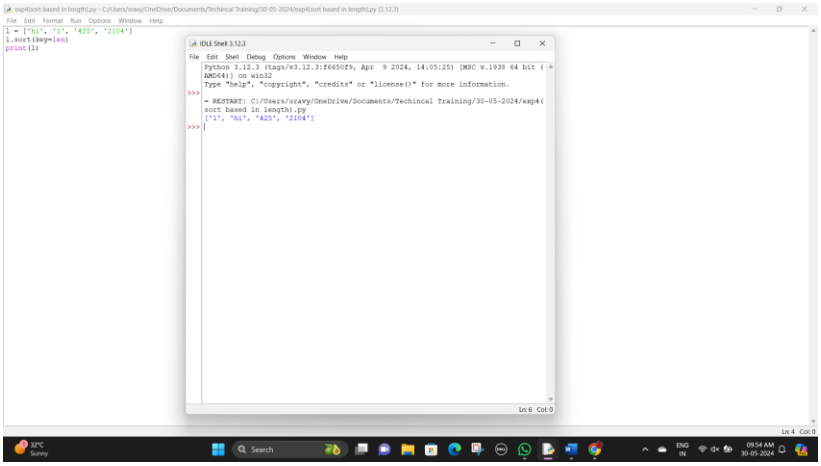
- List comprehension is an elegant way to define and create list of python .



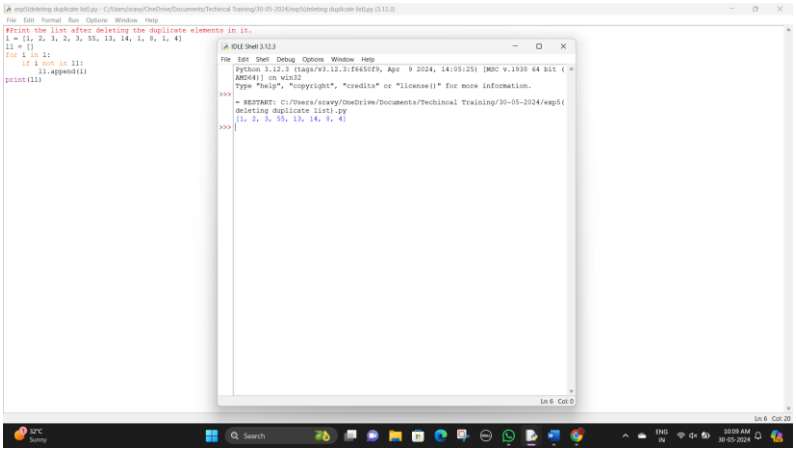
Take two different lists, merge them and print a new sorted list.



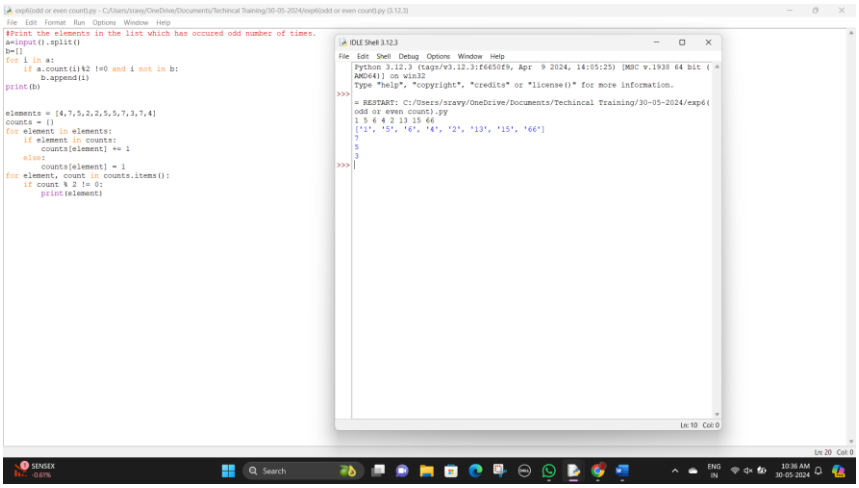
Input a list and sort the elements in the list based on its length.



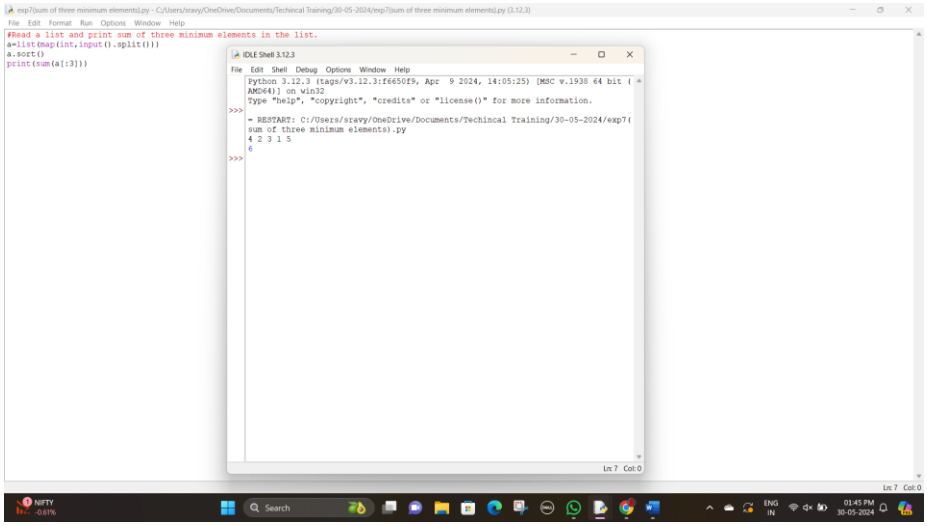
Print the list after deleting the duplicate elements in it.



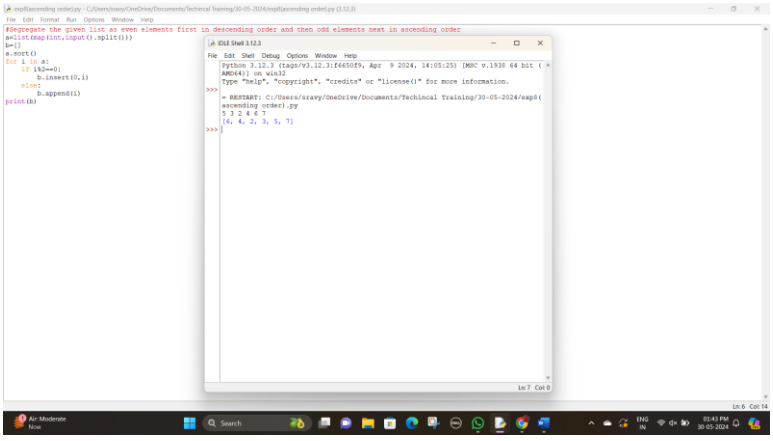
Print the elements in the list which has occurred odd number of times.



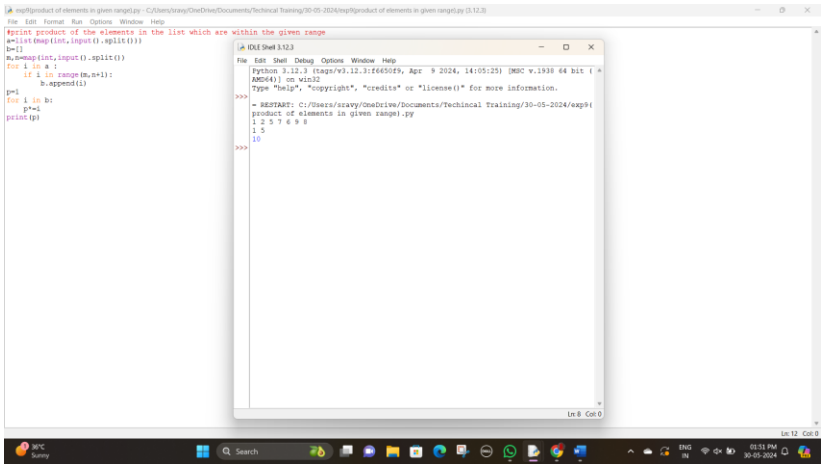
Read a list and print sum of three minimum elements in the list.



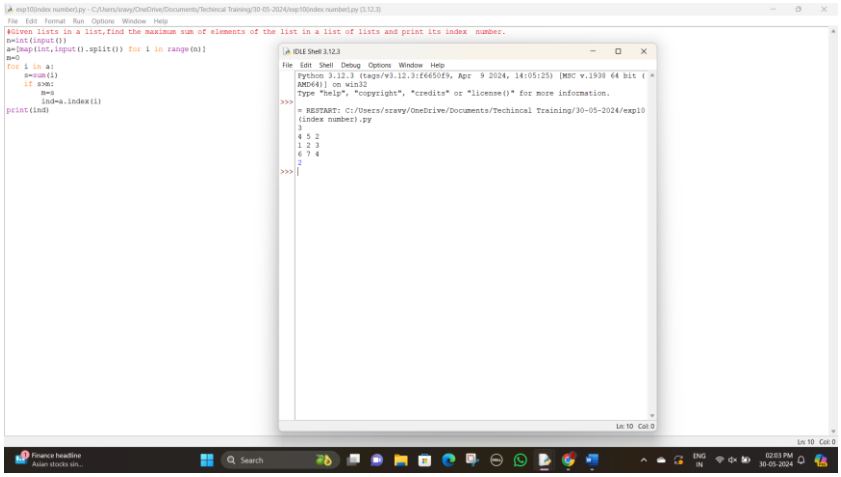
Segregate the given list as even elements first in descending order and then odd elements next in ascending order



print product of the elements in the list which are within the given range



Given lists in a list, find the maximum sum of elements of the list in a list of lists and print its index number .



TUPLE

- In tuple we can't update the elements just as list.

TUPLE BUILT-IN FUNCTIONS:

- **Sum():** Sums up the numbers in the tuple.
- **Max():** Return maximum element of given.
- **Min():** Return minimum element of given tuple.
- **All():** Returns true if all elements are true or if tuple is empty.
- **Any():** Returns true if any elements are true. If tuple is empty, return false.
- **Len():** Returns length of the list or size of the tuple.

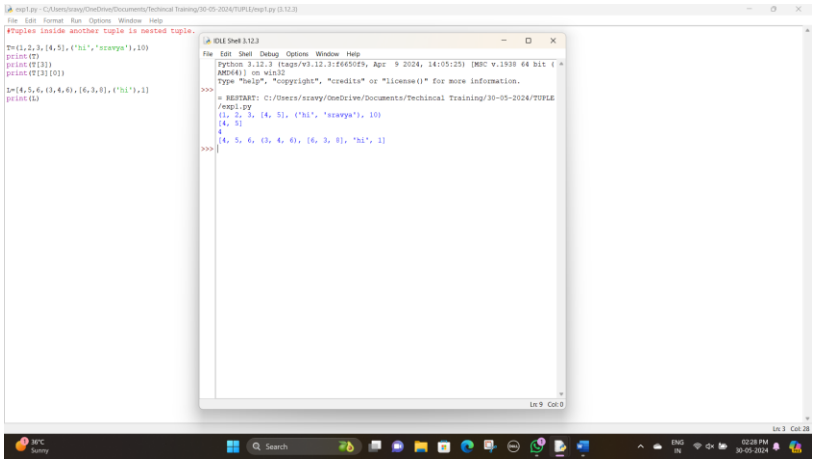
OPERATIONS ON TUPLE:

1. Concatenation of tuple (+)
2. Repetition of tuple (*)
3. Membership on tuple (in, not in)

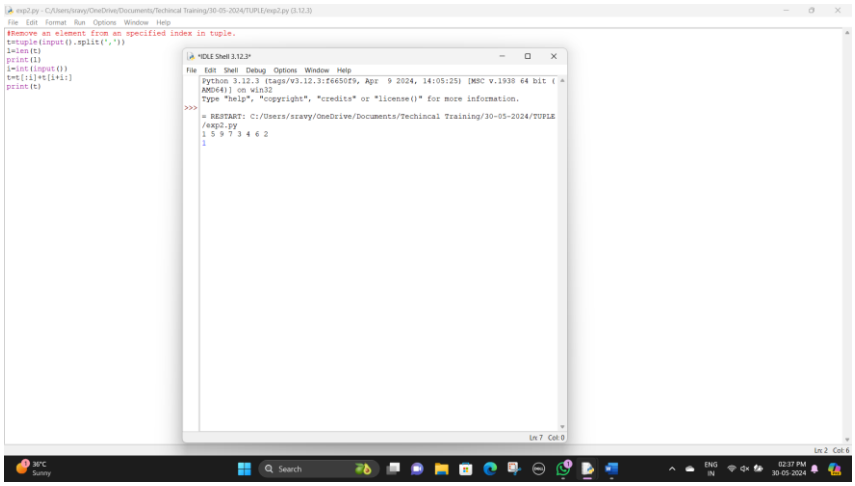
SLICING IN TUPLE:

- slice--> (:) using colon
- T[:]-->print elements from tuple
- T[2:]-->print all elements from 2 index in tuple
- T[:5]--> print all elements from tuple till ending index 4
- T[2:6]--> print elements from index 2 till 5
- [T[::2]--> 2(index+2elements)

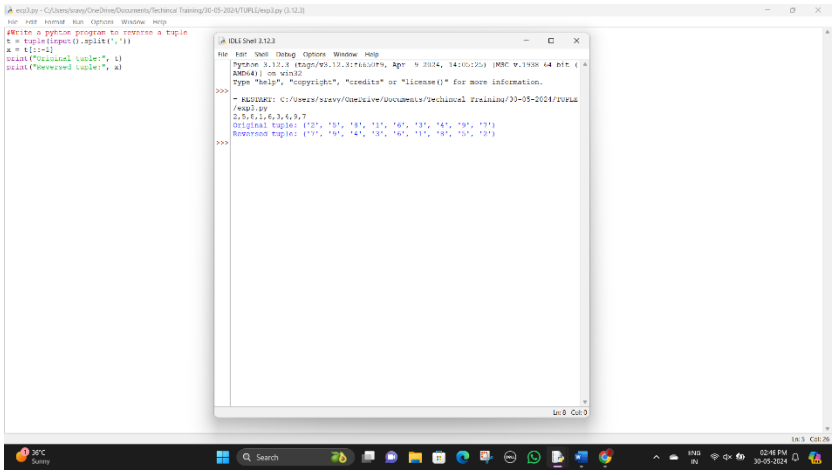
NESTED TUPLE:



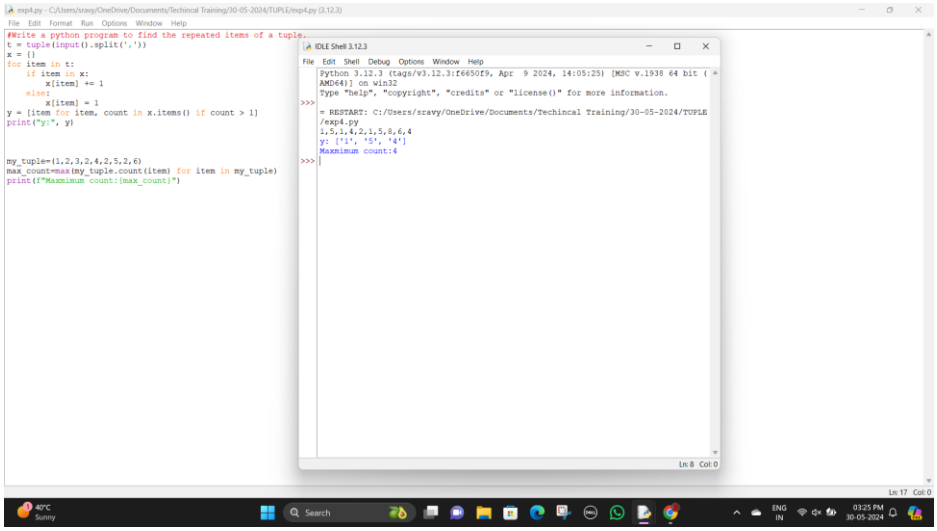
Remove an element from an specified index in tuple.



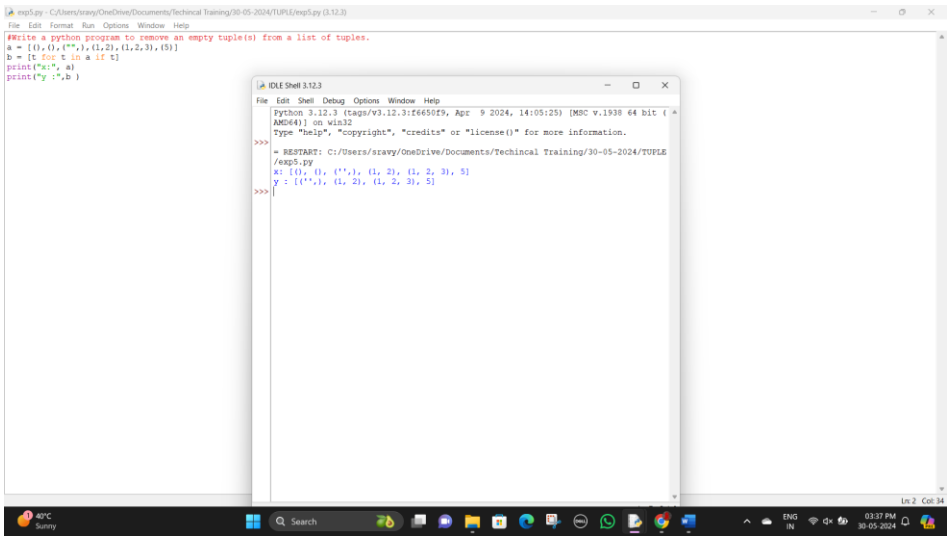
Write a python program to reverse a tuple.



Write a python program to find the repeated items of a tuple.

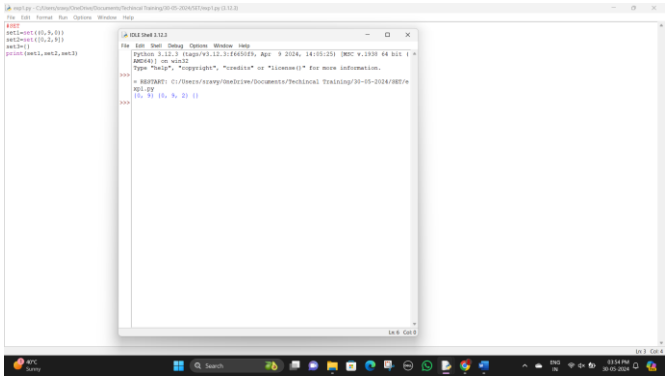


Write a python program to remove an empty tuple(s) from a list of tuples.



SET

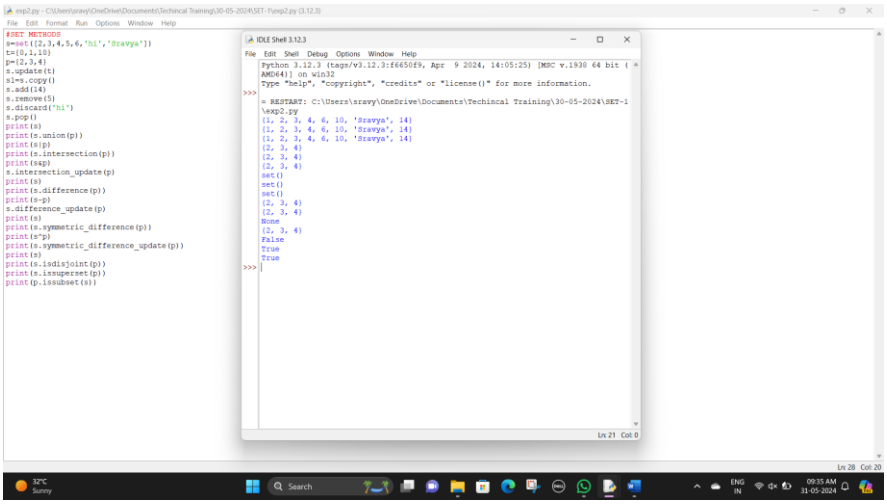
- A set is an unordered collection data type that is iterable, mutable and has no duplicate elements.
- Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.



SET METHODS:

- **Add():** Adds an elements to the set. [**Syntax:** s.add()]
- **Update():** Updates the set with the union of itself and others. [**Syntax:** s.update()]
- **Copy():** Returns a copy of the set. [**Syntax:** s.copy()]
- **Clear():** Removes all elements from the ser. [**Syntax:** s.clear()]
- **Remove():** Removes an element from the set. If the element is not a member, raise a KeyError. [**Syntax:** s.remove()]
- **Pop():** Removes and returns an arbitrary set element. Raise KeyError if the set is empty. [**Syntax:** print(s.pop())]
- **Discard():** Removes an element from the set if it is a member.(Do nothing if not in set) [**Syntax:** s.discard()]
- **Union():** Returns the union of sets in a new set. [**Syntax:** print(a.union(b))]
- **Intersection():** Retuen the intersection of two sets as a new set. [**Syntax:** print(a.intersection(b))]
- **Intersection_update():** Updates the set with the intersection of itself and another.[**Syntax:** a.intersection_update*b)]
- **Diiference():** Returns the difference of two or more sets as a new set. [**Syntax:** print(a.difference(b)]
- **Difference_update():** Removes all elements of another set from this set. [**Syntax:** a.difference_update(b)]
- **Symmetric_difference():** Returns the symmetric difference of two sets as a new set. [**Syntax:** print(a.symmetric_difference(b)]
- **Symmetric_difference_update():** Updates a set with the symmetric difference of itself and another. [**Syntax:** a. Symmetric_difference_update(b)]
- **Isdisjoint():** Returns True if two sets have a null intersection. [**Syntax:** print(a.isdisjoint(b))]
- **Issubset():** Returns True if another set contains this set. [**Syntax:** print(a.issubset(a)]

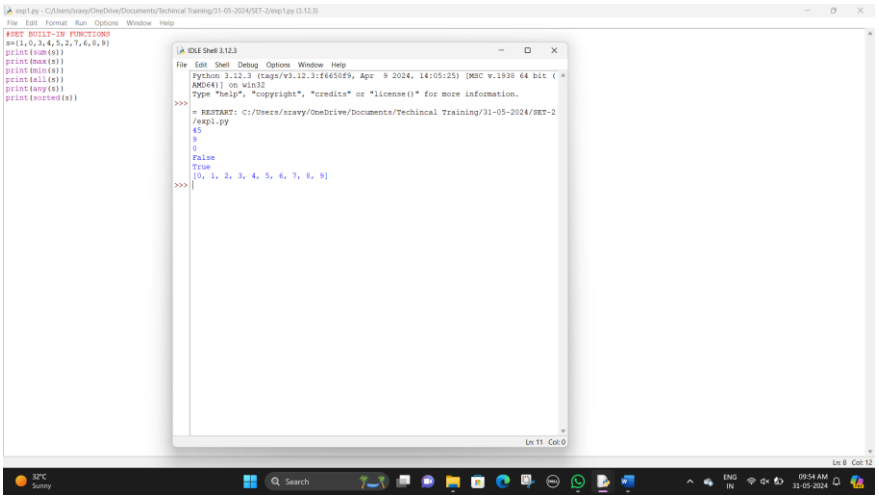
- **Issuperset():** Returns True if this set contains another set. [**Syntax:** print(b.issuperset(a))]



Dt:31-05-2024

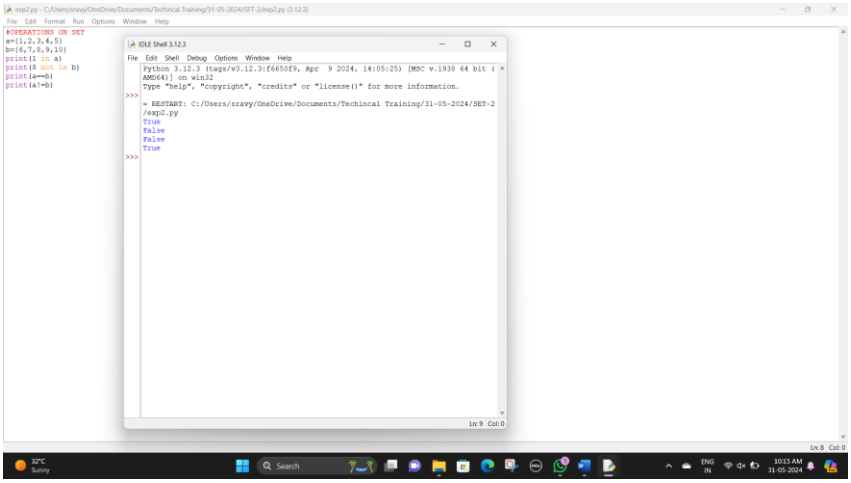
SET IN-BUILT FUNCTIONS:

- **Sum():** Sums of all elements in the set. [**Syntax:** print(sum())]
- **Max():** Return largest item of given. [**Syntax:** print(max())]
- **Min():** Return smallest item of given set. [**Syntax:** print(min())]
- **All():** Returns true if all elements of the set are true. [**Syntax:** print(all())]
- **Any():** Returns true if any elements are true. If set is empty, return false. [**Syntax:** print(any())]
- **Len():** Returns the length in the set. [**Syntax:** print(len())]
- **Sorted():** Return a new sorted list from elements in the set. [**Syntax:** print(sorted())]



OPERATIONS ON SET:

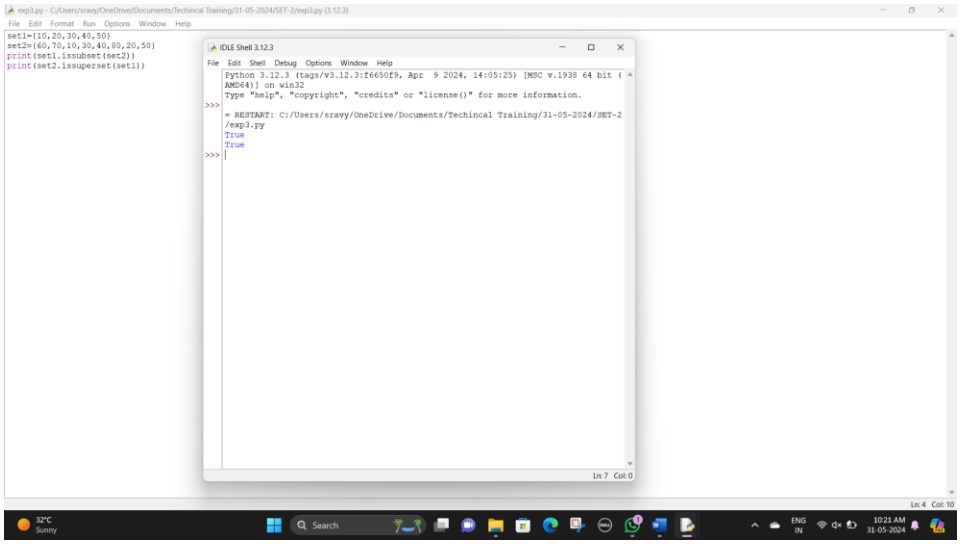
- Membership on tuple(in, not in).
- Equal(==), Not Equal(!=).
- Subset(<=), Proper Subset(<).
- Superset(>=), Proper Superset(>).
- Union(|), Intersection(&).
- Difference(-), Symmetric_Difference(^).



FROZEN SET:

- They are immutable sets.
- They can be created using function frozenset().
- This datatype supports methods like copy(), difference(),etc

Code:



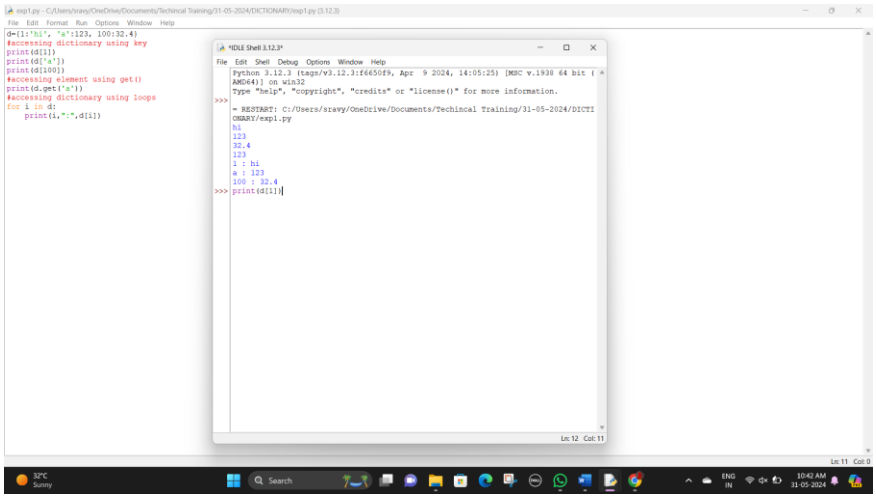
SET COMPREHENSION:

- Returns a set based on existing iterable.
- Syntax: {expression(variable) for variable in input_set[predicate][,…]}

DICTIONARY

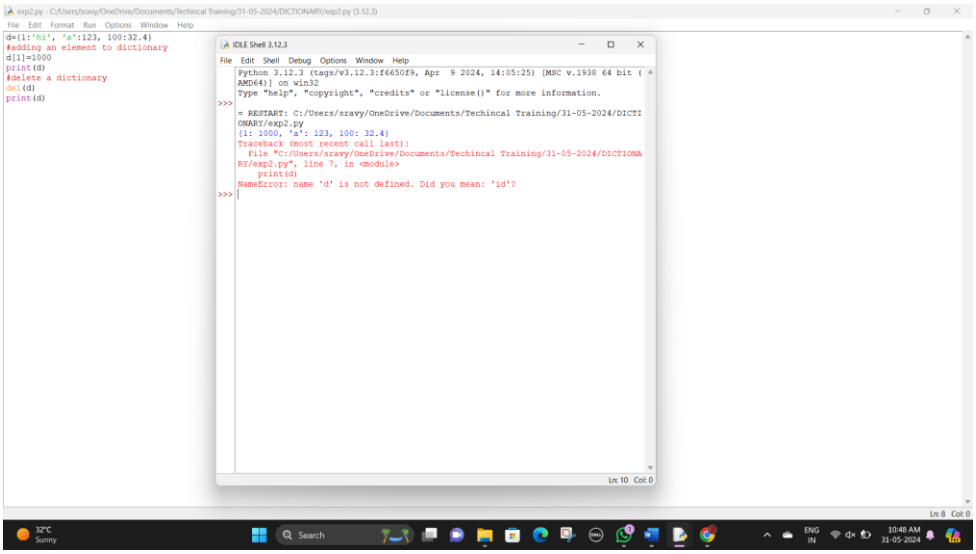
- A dictionary is a collection which is unordered , changeable and indexed.
- In python dictionaries are written in curly brackets.

ACCESSING ELEMENTS IN DICTIONARY:



MODIFY AND DELETE A DICTIONARY:

- Only values in the dictionary can be changed but keys should be immutable.



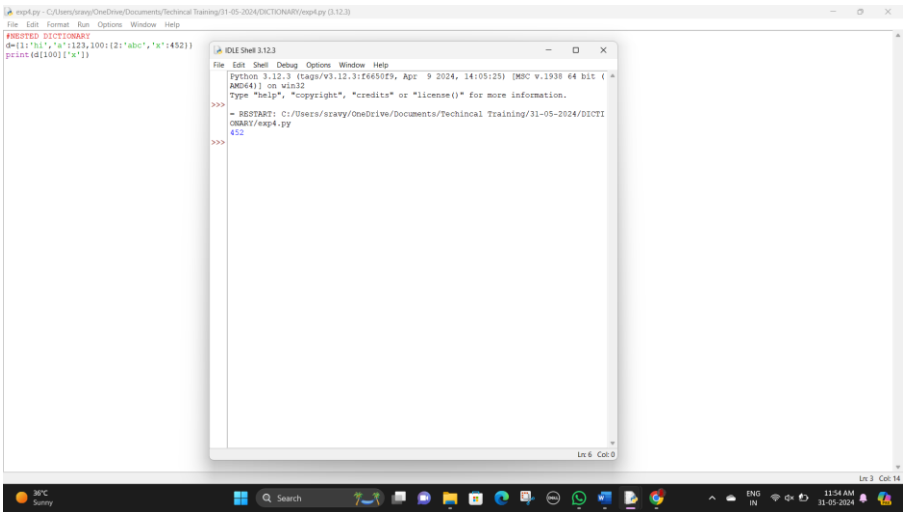
DICTIONARY METHODS:

- **Get():**
- **Update():** Adds two or more dictionaries.
- **Pop():** Removes and returns an element from a dictionary having the given key.
- **Popitem():** Removes the arbitrary key value pair from the dictionary and returns it as tuple.
- **Clear():** The clear() method removes all items from the dictionary.
- **Keys():** returns list of dictionary dictionaries keys.
- **Values():** returns a list of all the values available in a given dictionary.
- **Items():** returns a list of dictionaries(key, value) tuple pairs.

- **Fromkeys():** Create a new dictionary with keys from sequence and values set to value.
- **Setdefault():** Set dictionary[key]=default if key is not already in dictionary.

NESTED DICTIONARY:

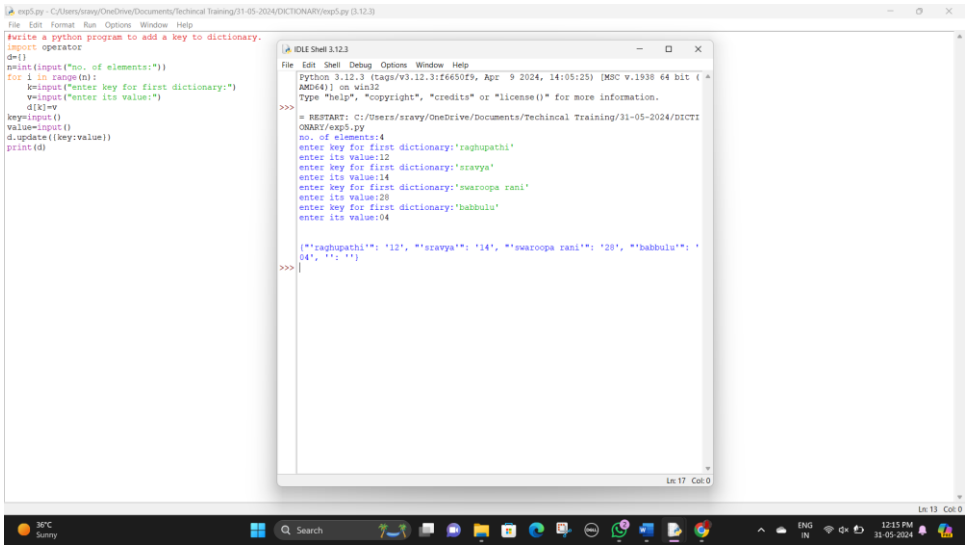
- Dictionary inside another dictionary is nested dictionary.



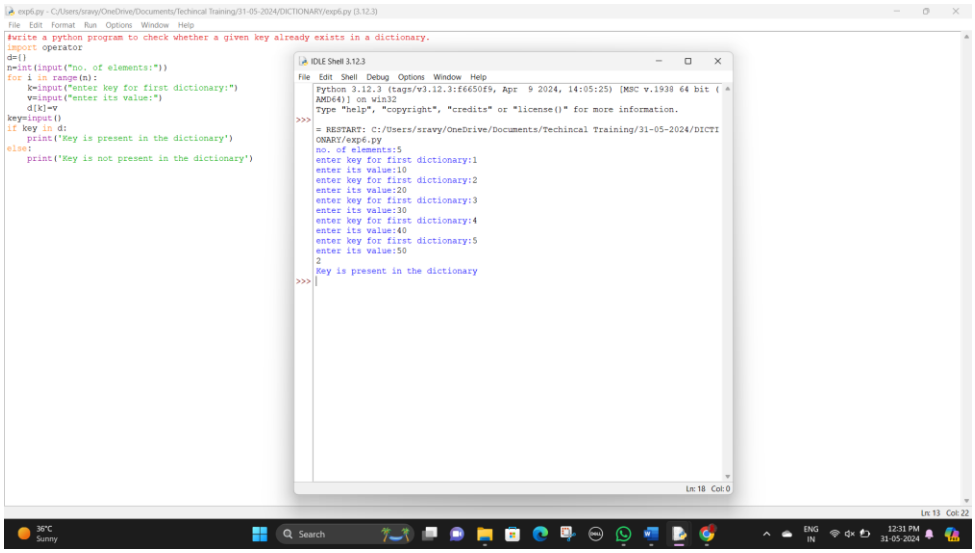
DICTIONARY COMPREHENSION:

- Dictionary comprehension is a method for transforming one dictionary into another dictionary.

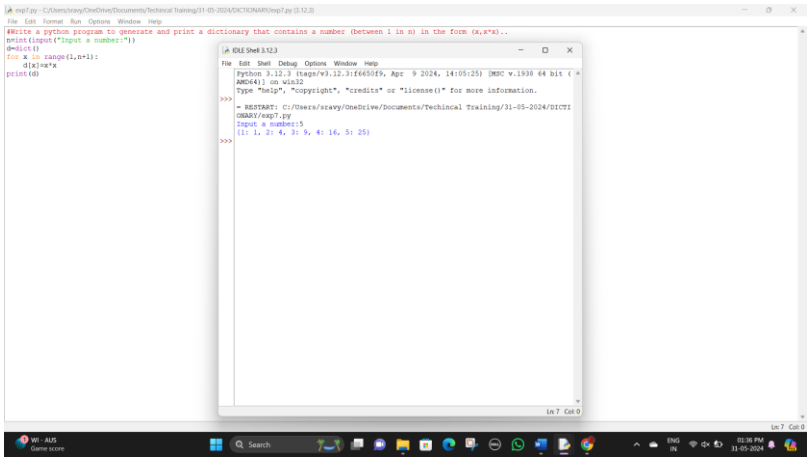
Write a python program to add a key to dictionary.



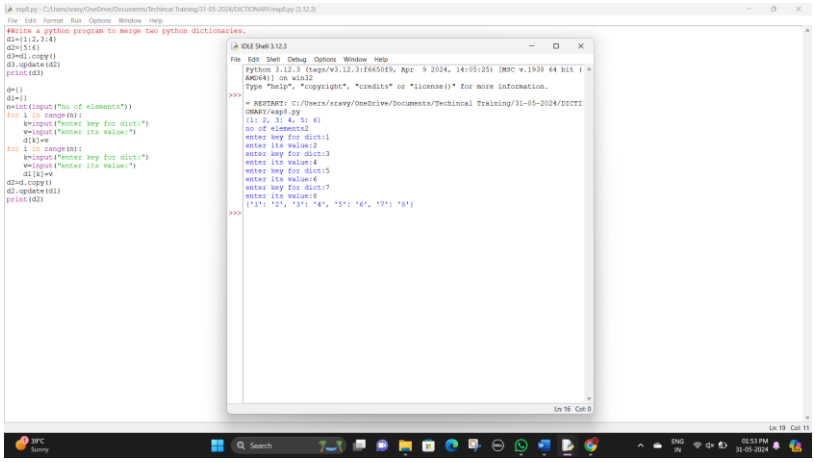
Write a python program to check whether a given key already exists in a dictionary.



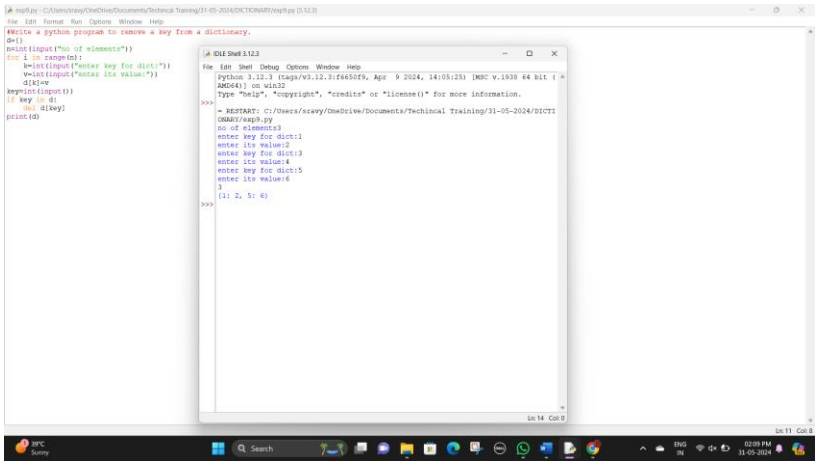
Write a python program to generate and print a dictionary that contains a number (between 1 in n) in the form (x,x*x)..



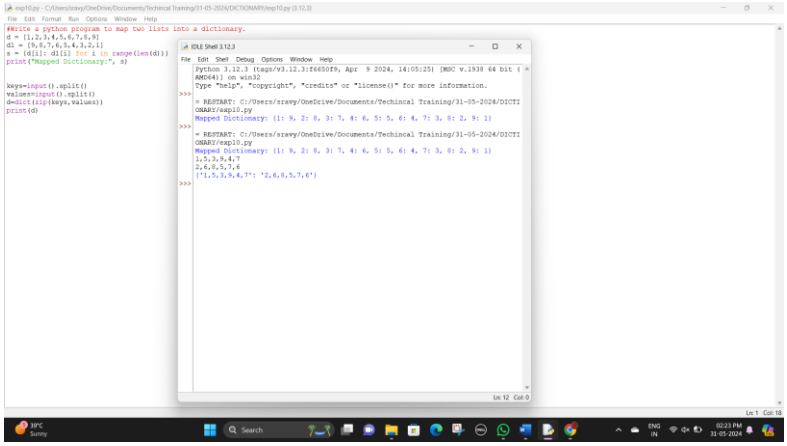
Write a python program to merge two python dictionaries.



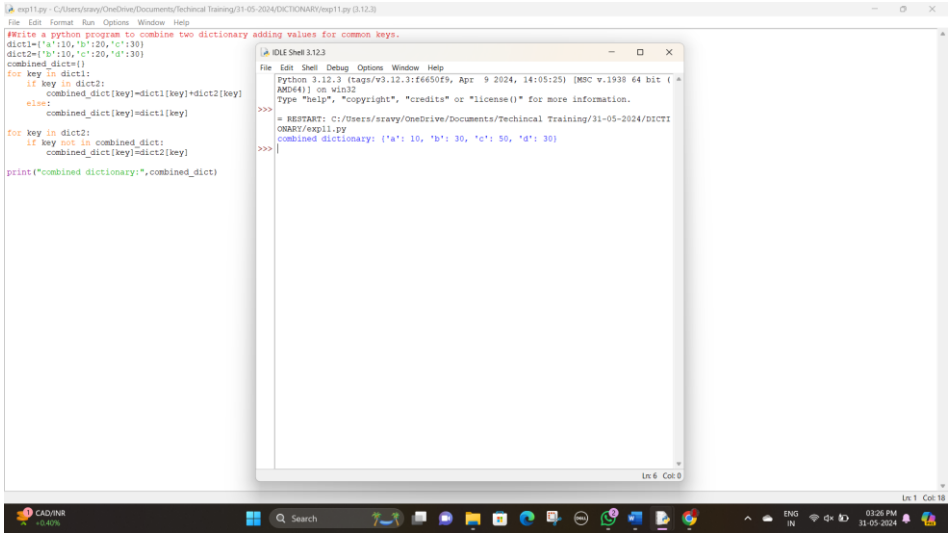
Write a python program to remove a key from a dictionary.



Write a python program to map two lists into a dictionary.



Write a python program to combine two dictionary adding values for common keys.



STRINGS

- Sequences of characters.

MODIFY A STRING:

- Strings are immutable, hence elements of a string cannot be changed once it has been assigned.
- Only new strings can be reassigned to same name.

DELETING A STRING:

- We can directly delete a string using del keyword.

STRING METHODS AND FUNCTIONS:

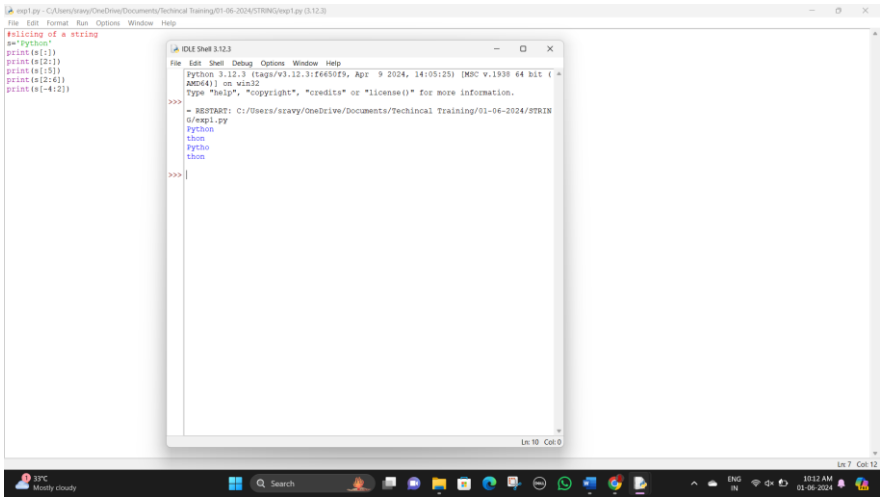
- **String.ascci_letters:** Concatenation of the ascii_lowercase and ascii_uppercase constants. [Syntax: print(string.ascii_letters)]
- **String.ascci_lowercase:** Concatenation of lowercase letters. [Syntax: print(string.ascii_lowercase)]
- **String.ascci_uppercase:** Concatenation of uppercase letters. [Syntax: print(string.ascii_uppercase)]
- **String.ascci_punctuation:** ASCII characters having punctuation character. [Syntax: print(string.ascii_punctuation)]
- **String.digits:** Digit in strings. [Syntax: print(string.digits)]
- **String.hexdigits:** Hexadigit in strings. [Syntax: print(string.hexdigits)]
- **String.octdigits:** Octadigit in strings. [Syntax: print(string.octdigits)]
- **String.endswith():** Returns True if a string ends with the given suffix otherwise return False. [Syntax: print('hello'.endswith('o'))]
- **String.startswith():** Returns True if a string ends with the given prefix otherwise return False. [Syntax: print('hello'.startswith('h'))]
- **replace():** Returns a copy of the string where all occurrences of a substring is replaced with another substring. [Syntax: print('hello'.replace('l','t'))]
- **String.isdigit():** Returns True if all characters in the string are digits, Otherwise, It return False. [Syntax: print('100'.isdigit())]
- **String.isalpha():** Returns True if all characters in the string are alphabets, Otherwise, It return False. [Syntax: print('abc'.isalpha())]
- **String.isdigit():** Returns True if all characters in the string are decimal. [Syntax: print('72'.isdecimal())]
- **String.isalnum:** Returns true if all the characters in a given string are alphanumeric. [Syntax: print('ab12'.isalnum())]
- **String.istitle:** Returns True if the string is a titlecased string. [Syntax: print('Hello World'.istitle())]
- **String.upper():** Returns the string with all uppercases. [Syntax: print('hello'.upper())]
- **String.lower():** Returns the string with all lowercases. [Syntax: print('hello'.lower())]

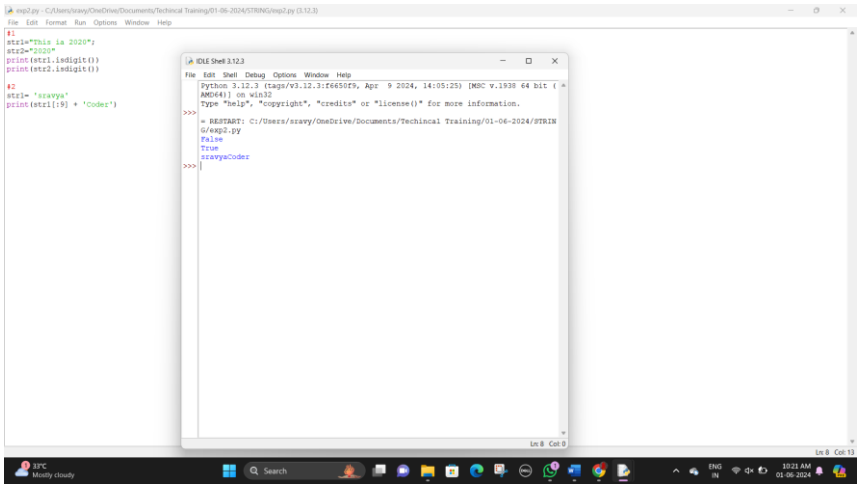
- **String.swapcase():** Returns the string with all uppercases to lowercases and vice versa. [Syntax: print('hello'.swapcase())]
- **String.partition():** Splits the string at the first occurrence of the separator and returns a tuple. [Syntax: print(str.partition())]
- **String.index():** Returns the position at the first occurrence of substring in a string. [Syntax: print(str.index())]
- **String.rindex():** Returns the highest index of the substring inside the string if substring is found. [Syntax: print(str.rindex())]
- **String.splitlines():** Returns a list of lines in the string. [Syntax: print(str.splitlines())]

Dt:01-06-2024

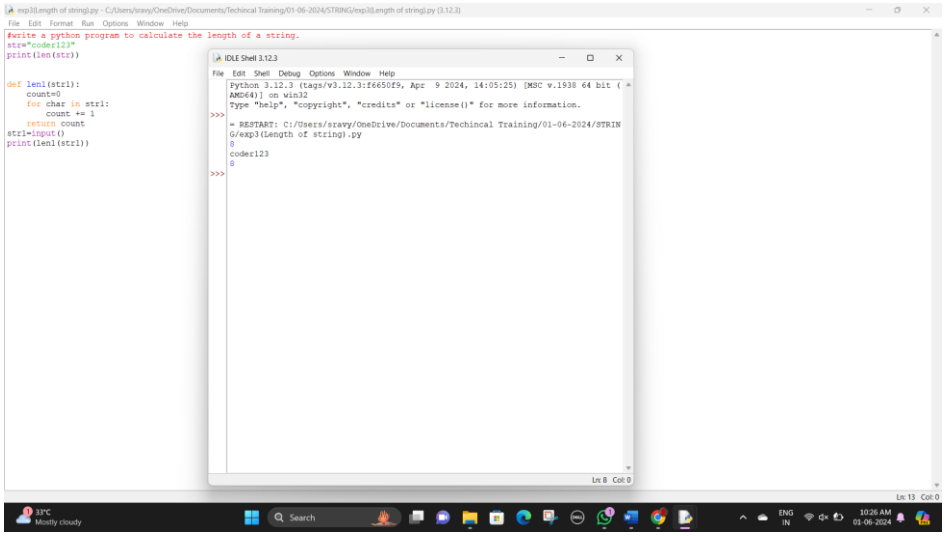
- **String.capitalize:** Return a word with its first character capitalized. [Syntax: print('python programming'.capitalize())]
- **String.find:** Return the lowest index in a sub string. [Syntax: print('python cython'.find('th'))]
- **String.rfind:** Find the highest index. [Syntax: print('python cython'.rfind('th'))]
- **String.count:** Return the number of (non-overlapping) occurrences of substring sub in string. [Syntax: print('python cython'.count('th'))]
- **Len():** Returns the length of the string. [Syntax: print(len())]
- **Max():** Returns the highest alphabetical character in a string. [Syntax: print(max())]
- **Min():** Returns the minimum alphabetical character in a string. [Syntax: print(min())]

SLICING A STRING:

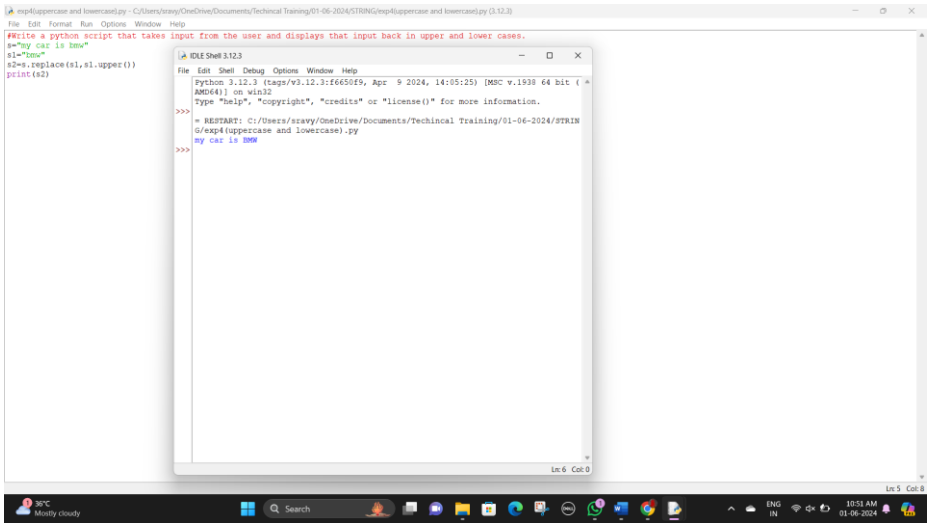




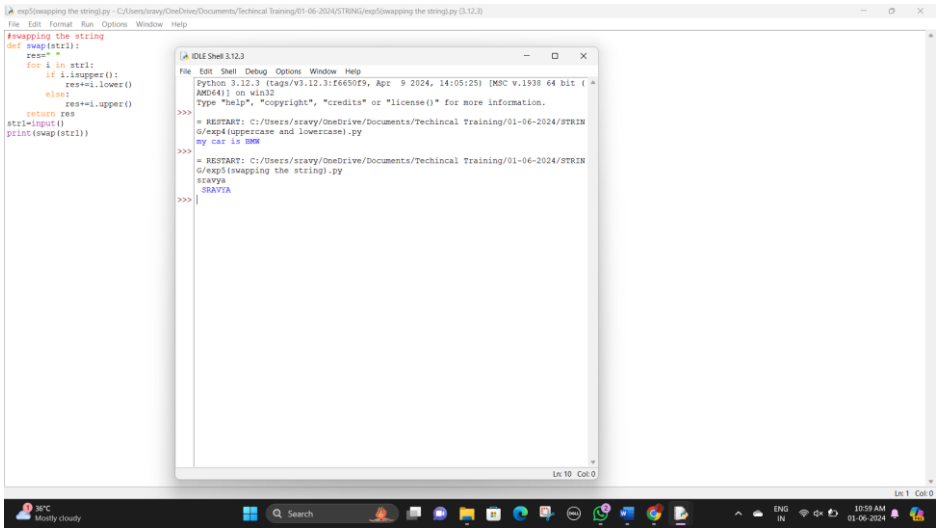
Write a python program to calculate the length of a string.



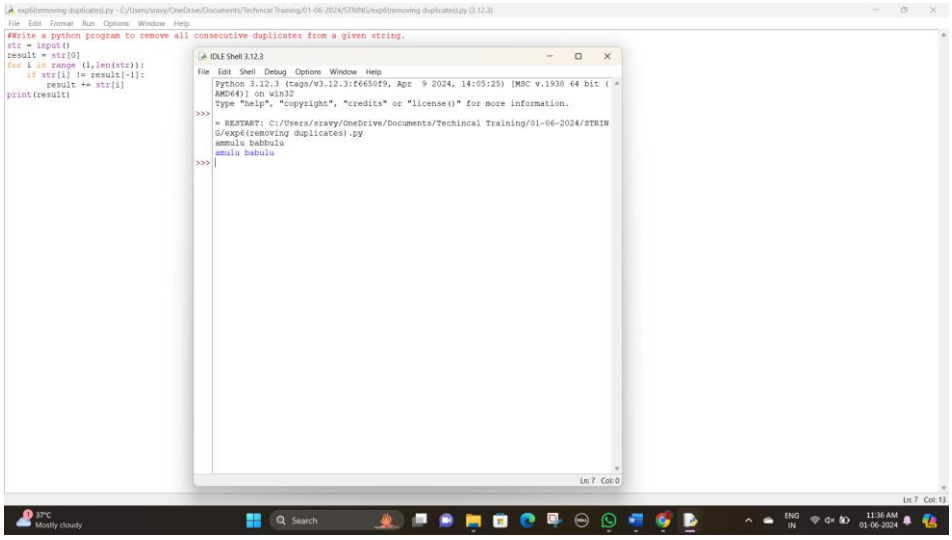
Write a python script that takes input from the user and displays that input back in upper and lower cases.



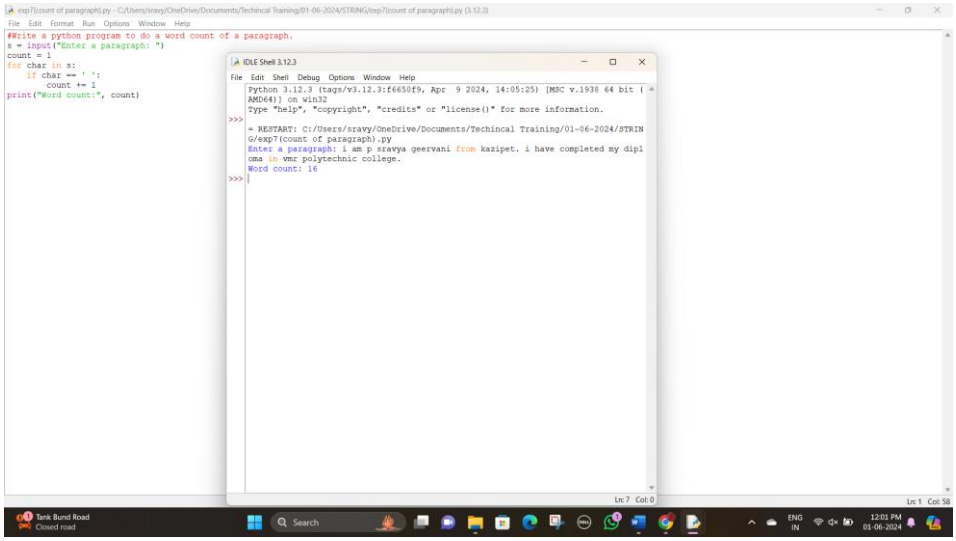
Write a python program to swap a string.



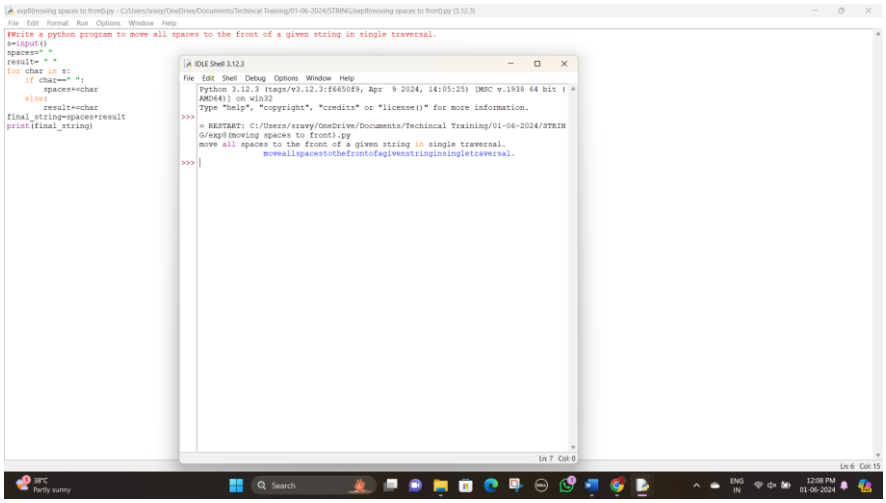
Write a python program to remove all consecutive duplicates from a given string.



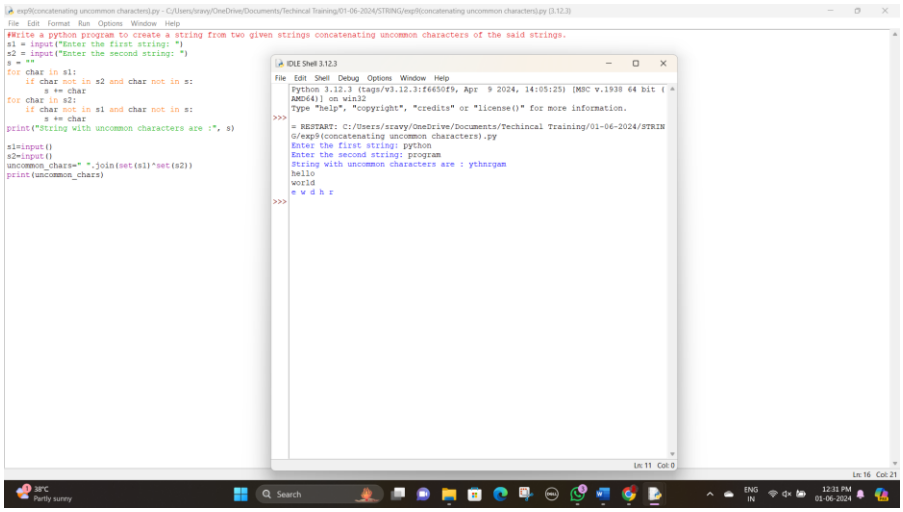
Write a python program to do a word count of a paragraph.



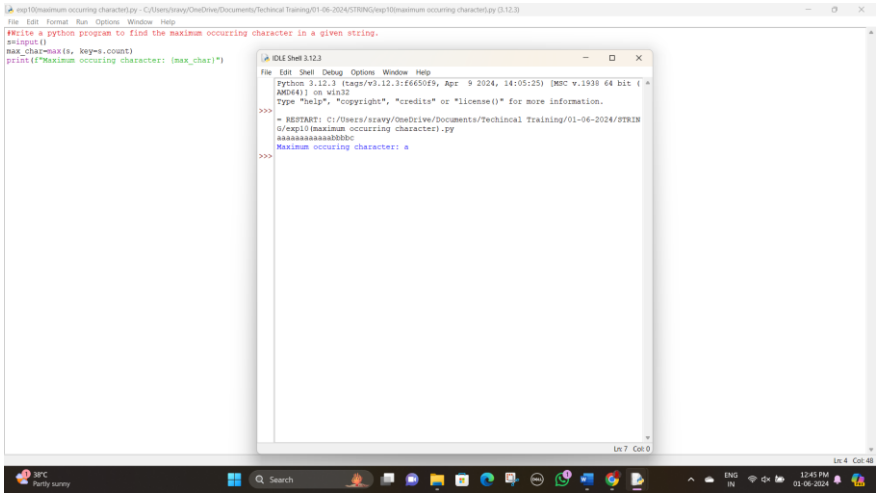
Write a python program to move all spaces to the front of a given string in single traversal.



Write a python program to create a string from two given strings concatenating uncommon characters of the said strings.



Write a python program to find the maximum occurring character in a given string.



FUNCTIONS

- The syntax to declare a function is: def function_name(parameters):
- Def: keyword, function_name: Function name and parameters: parameter

Def function_name(parameters):

#statement→ body of statement

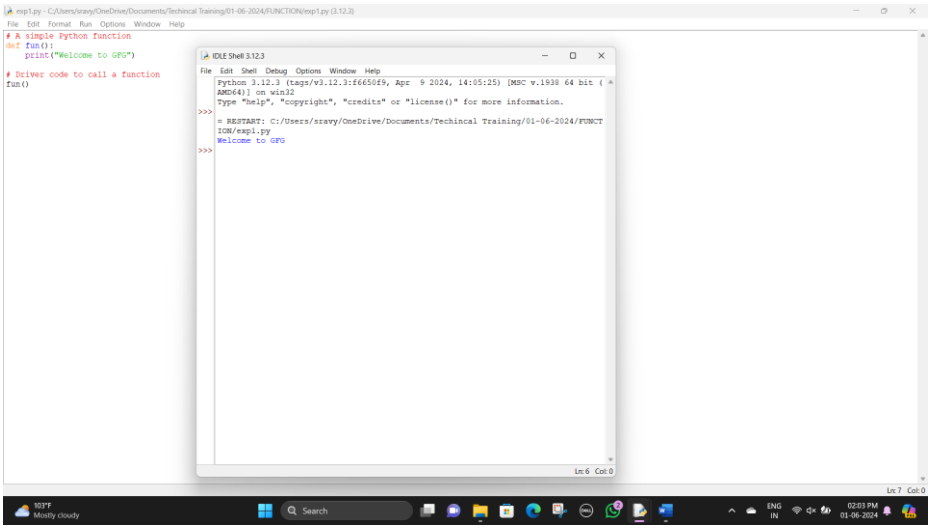
Function return<-----Return expression

- **Parameters:** A parameter is the variable defined within the parameters during function definition. Simply they are written when we declare a function.

TYPES OF FUNCTIONS:

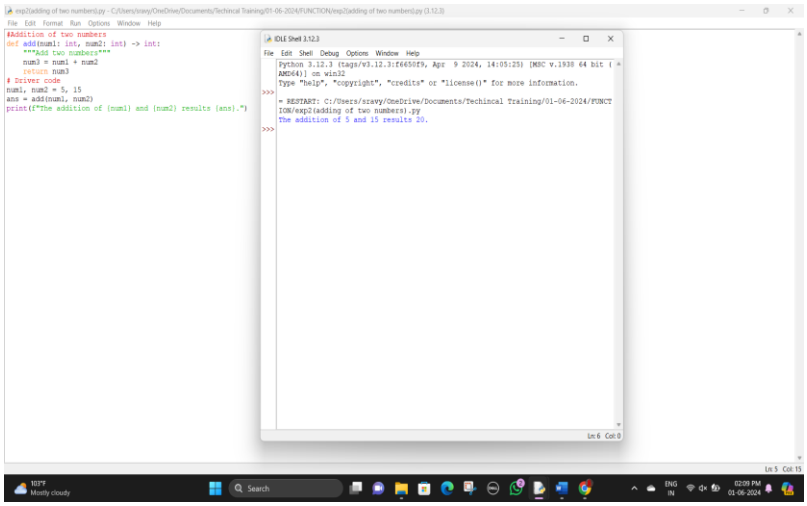
- **Built-in Function:** These are standard functions in Python that are available to use.
- **User-defined Function:** We can create our own functions based on our requirements.

CREATING AND CALL A FUNCTION:

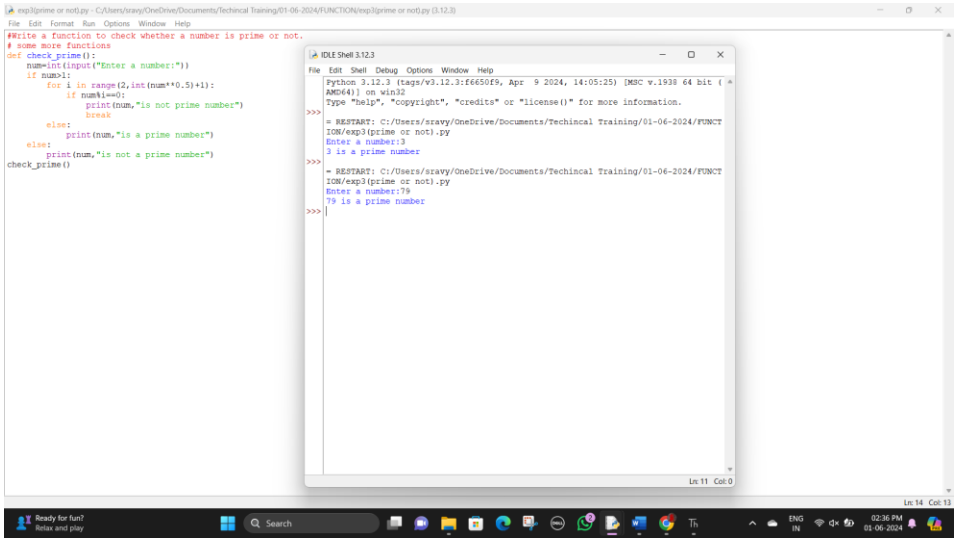


PYTHON FUNCTION WITH PARAMETERS

Addition of two numbers

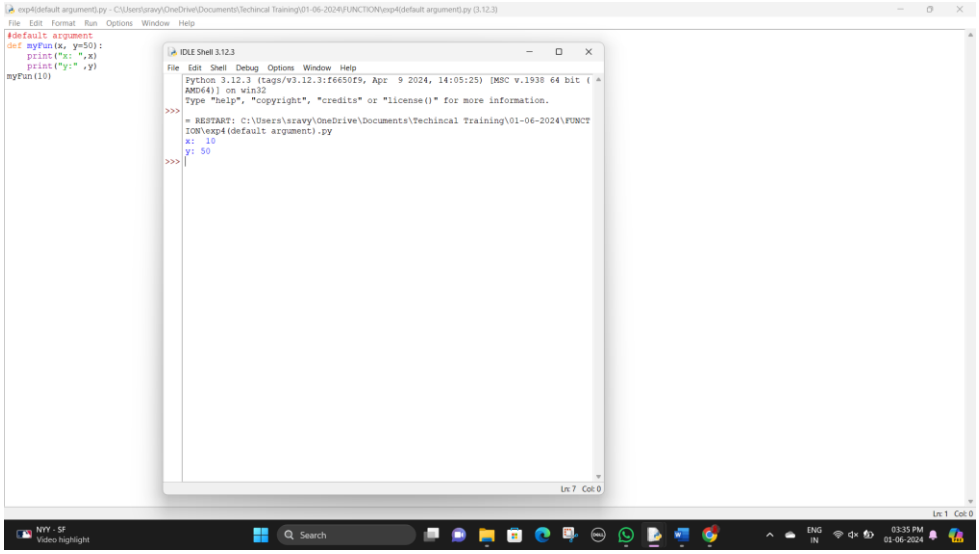


Write a function to check whether a number is prime or not.

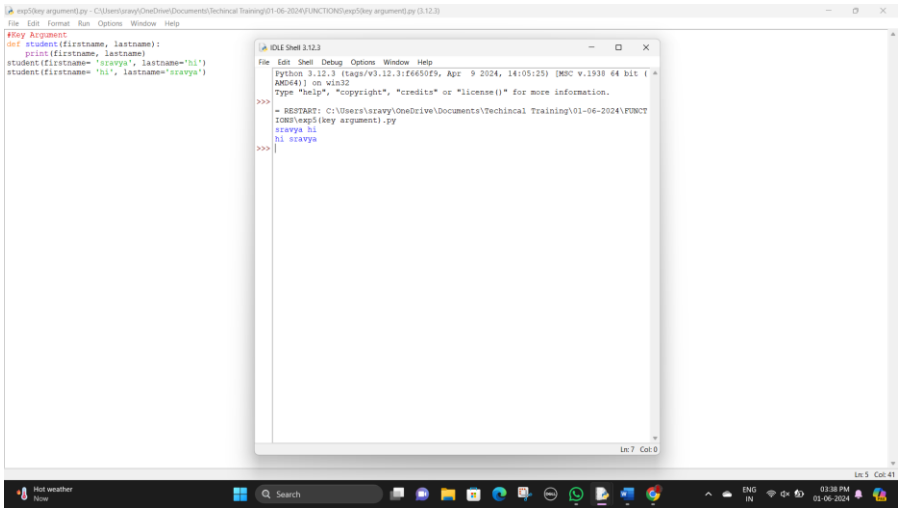


ARGUMENTS IN FUNCTIONS

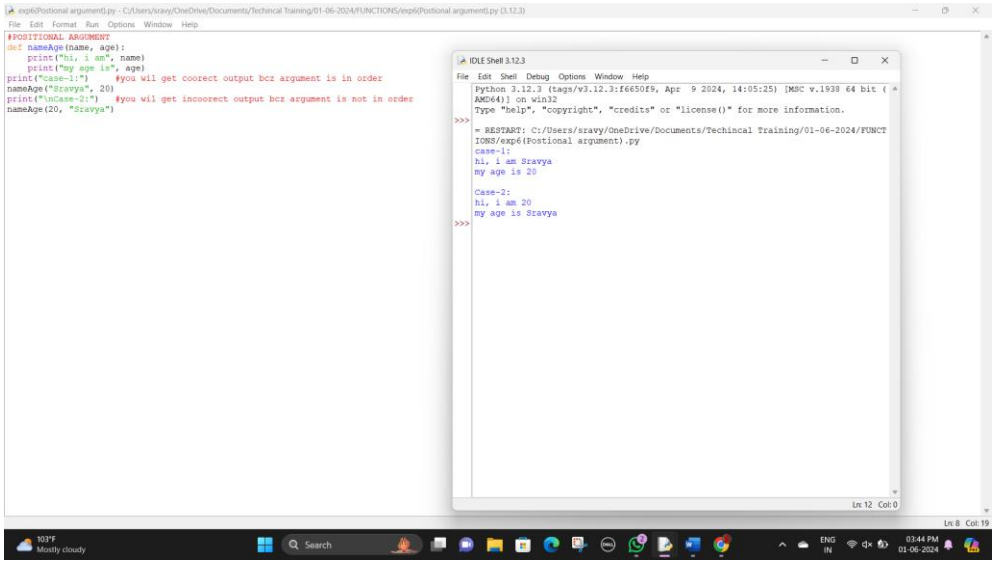
DEFAULT ARGUMENT



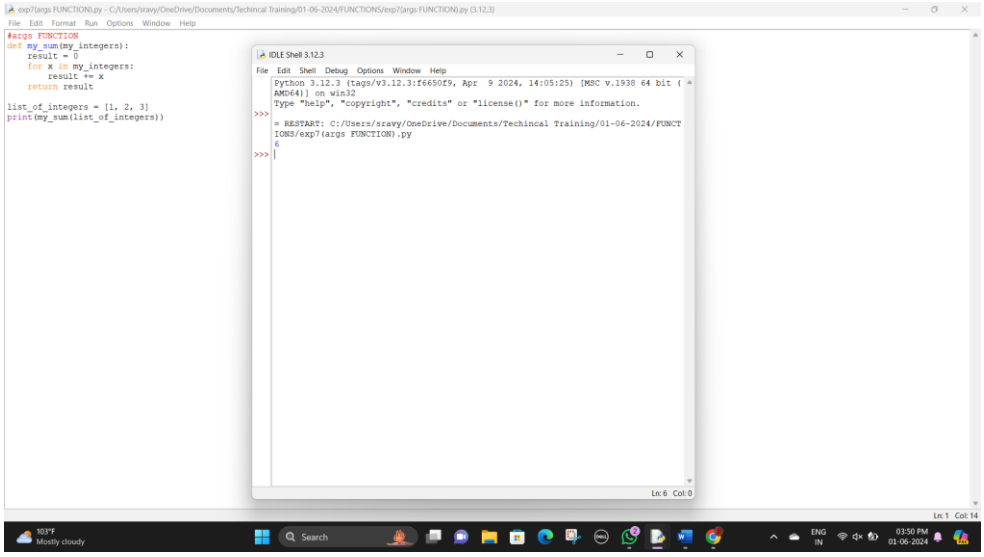
KEY ARGUMENT



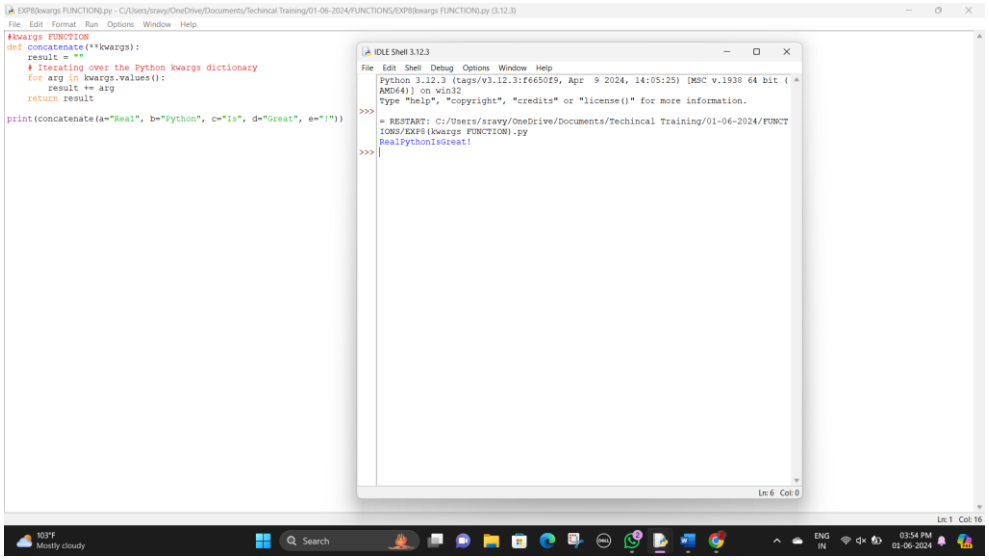
POSITIONAL ARGUMENT



USING THE PYTHON args VARIABLE IN FUNCTION



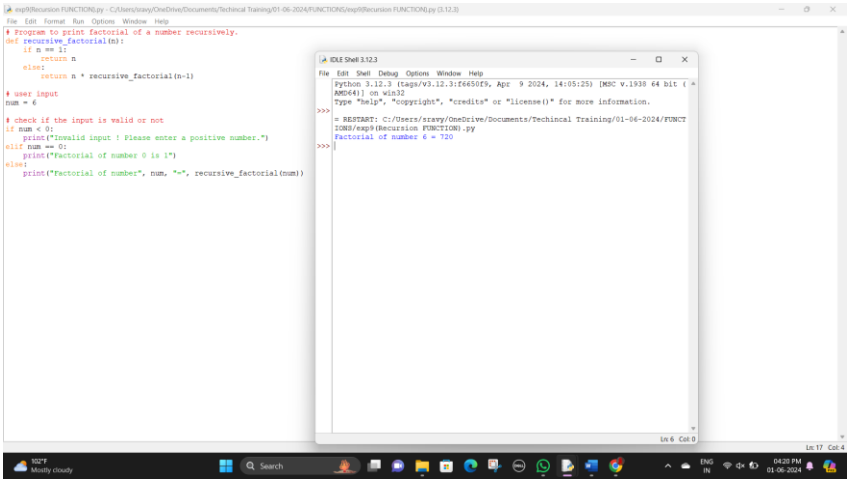
USING THE PYTHON kwargs VARIABLE IN FUNCTION



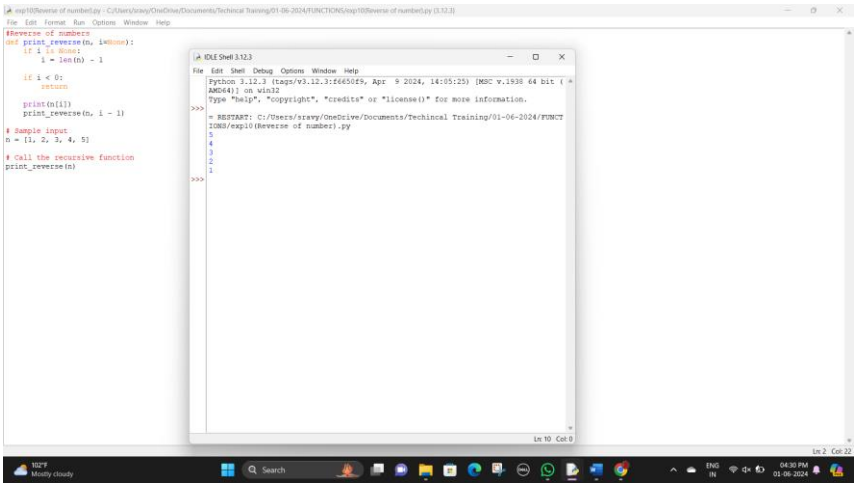
RECURSION

- The process of defining something in terms of itself. In simple words, it is a process in which a function calls itself directly or indirectly.

Factorial of a number recursively

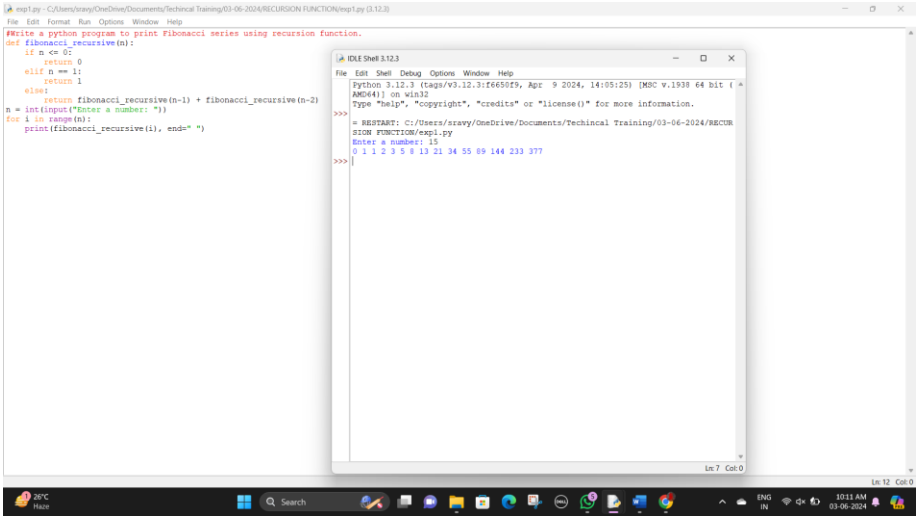


Reverse of numbers using recursion

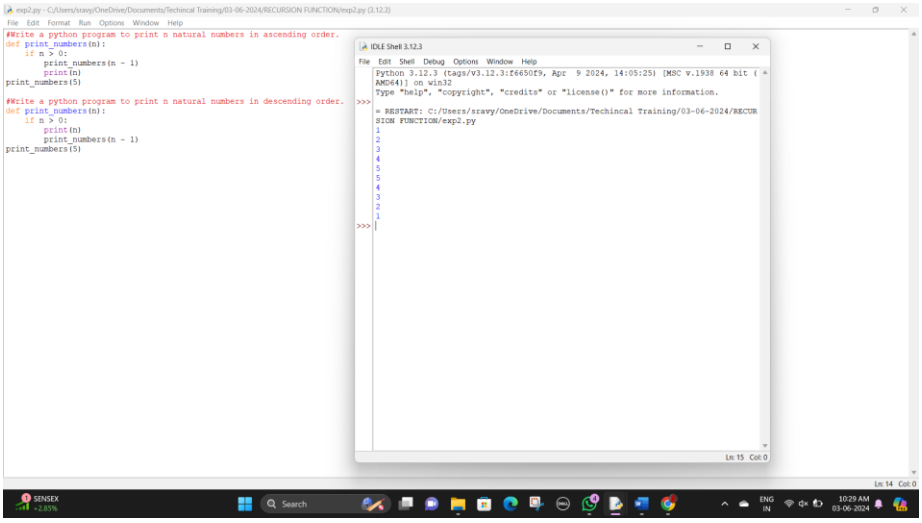


Dt:03-06-2024

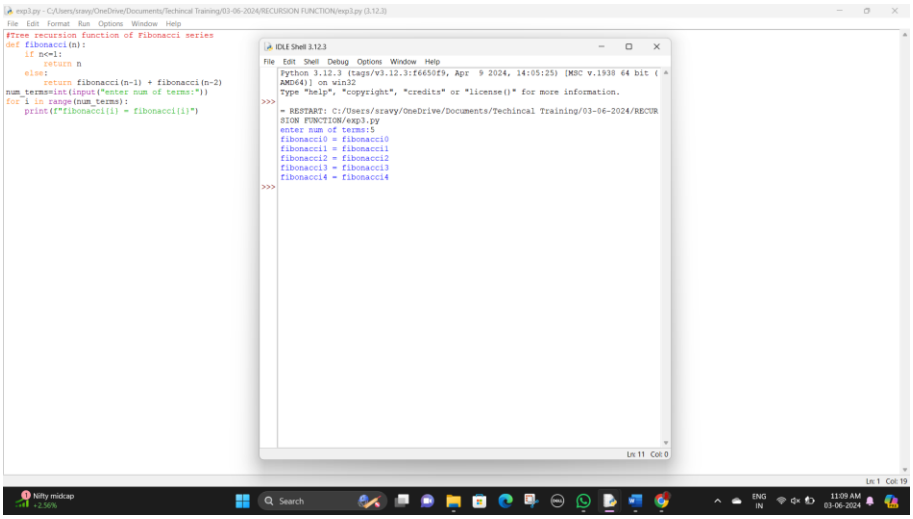
Write a python program to print Fibonacci series using recursion function.



Write a python program to print n natural numbers in ascending and descending order.



Tree recursion function of Fibonacci series.



PYTHON AND DATA ANALYSIS

→ Python is the leading language of choice for many data scientists.
→ Python supports Object-Oriented Programming.

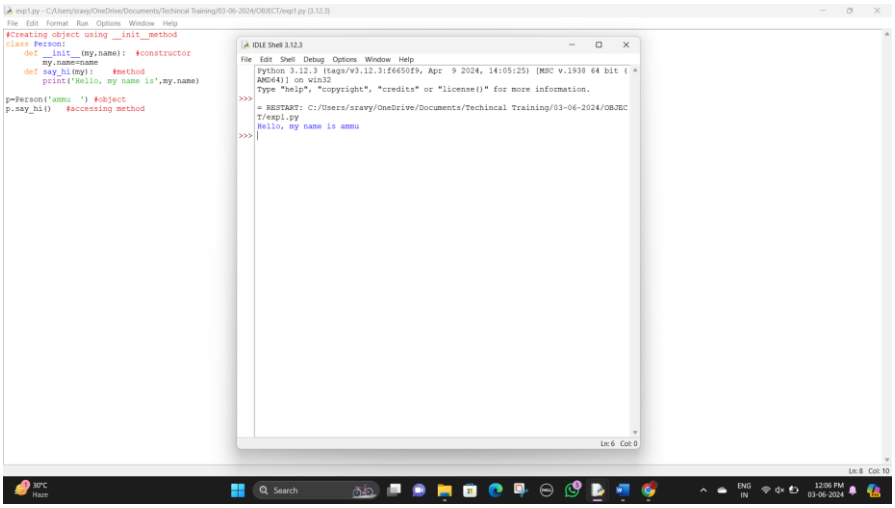
BASIC TERMINOLOGIES OF OOPS

→ Class
→ Objects
→ Methods
→ Constructors

1. **CLASS:**
   - Class is an abstract as well user defined data type,
   - The primary purpose of class is to store data and information.
   - The members of the class defines the class.
   - Class cannot be seen to real world but an object can be seen.
   - A class is basically an object constructor or a blueprint for an object.
2. **OBJECTS:**
   - Objects are an encapsulation of variables and functions into a single entity.
   - Objects get their variables and function from classes.
   - The implementation of class is an object.
   - The class is not visible to the outside world but object does.

OBJECT CREATION using__init__method

- The __init__method is similar to constructors in C++ and Java.
- It is run as soon as an object of a class is **instantiated.**
- The method is useful to do any **initialization** you want to do with your **object.**



OOPS CONCEPT

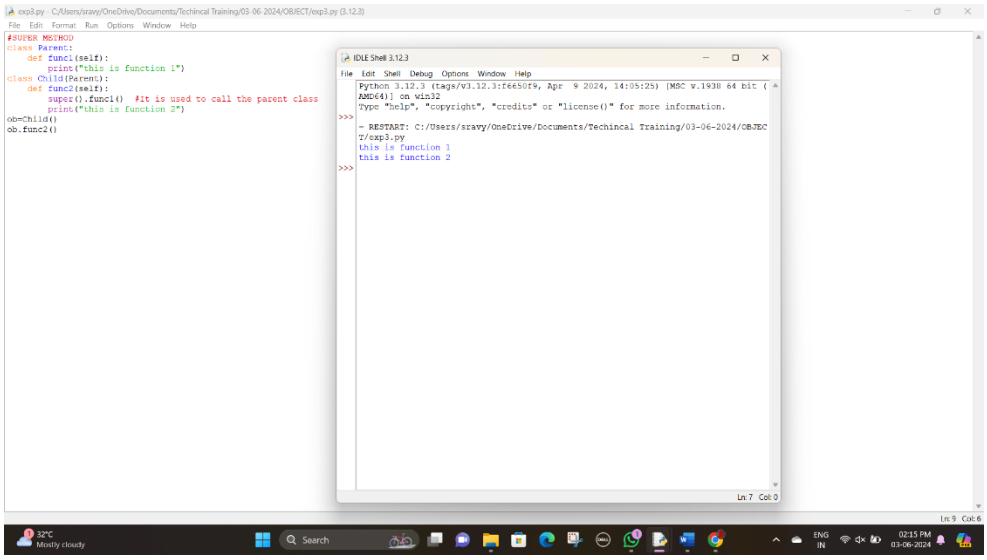
- → Inheritance
- → Polymorphism
- → Data Encapsulation
- → Data Abstraction
- → Except Handling

INHERITANCE

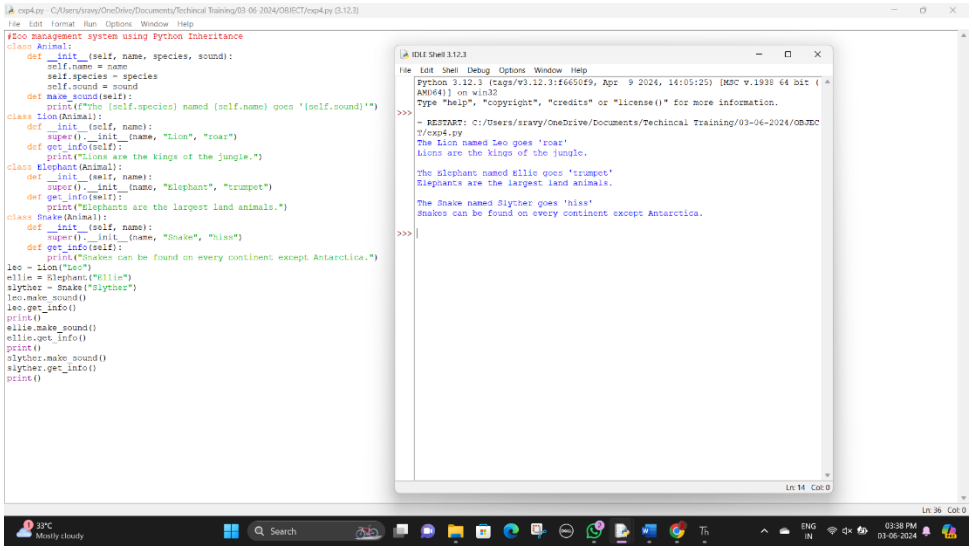
TYPES OF INHERITENCE:

- → Single Inheritance: When a child class inherits from one parent class, it is single inheritance.
- → Multiple Inheritance: When a child class inherits from more than one parent class, it is single inheritance.
- → Multilevel Inheritance: When a child class inherits from other class, it is multi-level inheritance.
- → Hierarchal Inheritance: Hierarchal inheritance involves multiple inheritance from the same base or parent class.
- → Hybrid Inheritance: Hybrid inheritance involves multiple inheritance taking place in single program.

SUPER METHOD



Zoo management system using Python Inheritance



Anton likes to play chess, and so does his friend Danik.

Once they have played n games in a row. For each game it's known who was the winner — Anton or Danik. None of the games ended with a tie.

Now Anton wonders, who won more games, he or Danik? Help him determine this.

Input

The first line of the input contains a single integer n (1 ≤ n ≤ 100 000) — the number of games played.

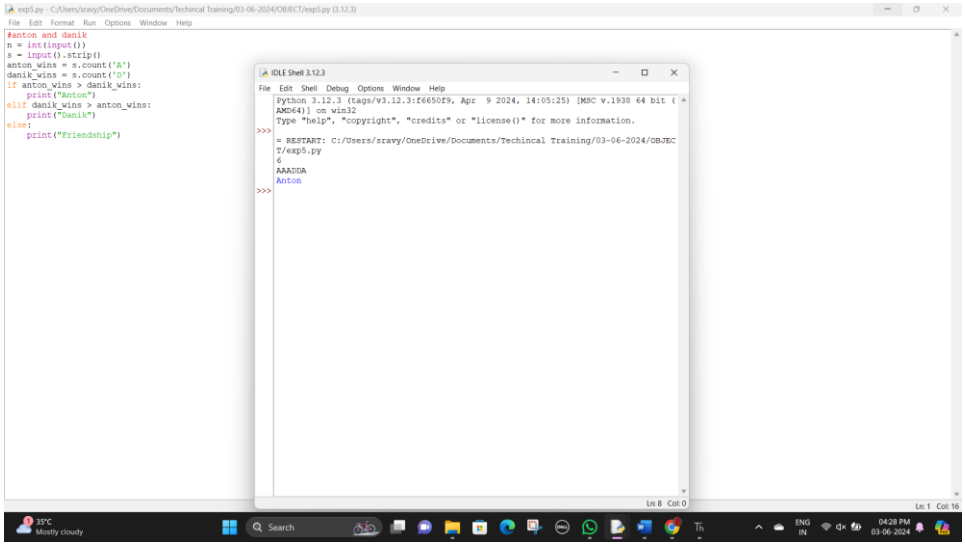
The second line contains a string s, consisting of n uppercase English letters 'A' and 'D' — the outcome of each of the games. The i-th character of the string is equal to 'A' if the Anton won the i-th game and 'D' if Danik won the i-th game.

Output

If Anton won more games than Danik, print "Anton" (without quotes) in the only line of the output.

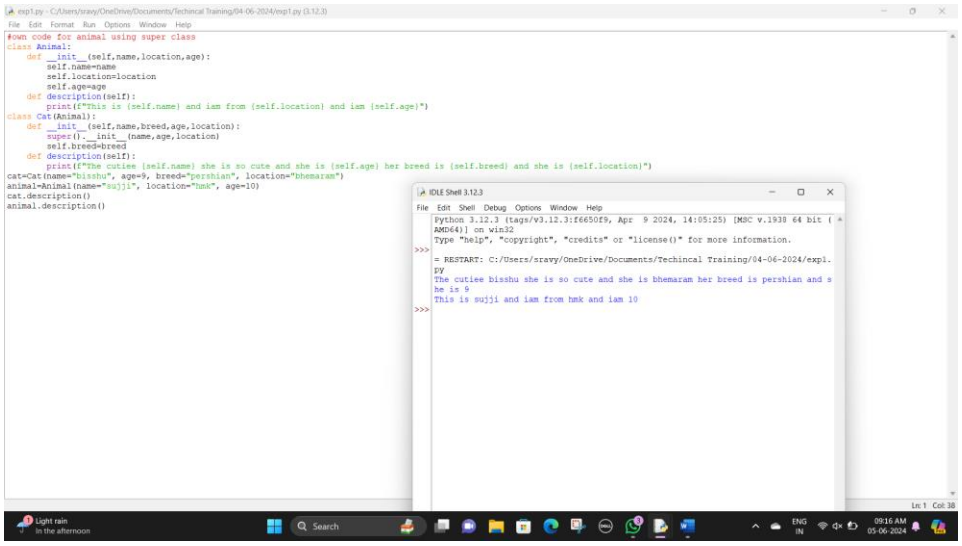
If Danik won more games than Anton, print "Danik" (without quotes) in the only line of the output.

If Anton and Danik won the same number of games, print "Friendship" (without quotes).

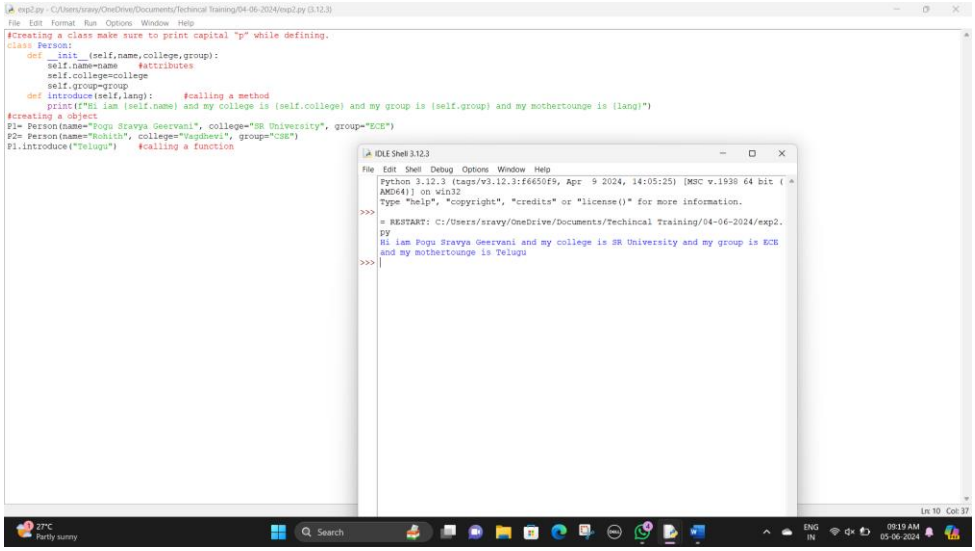


Dt:04-06-2024

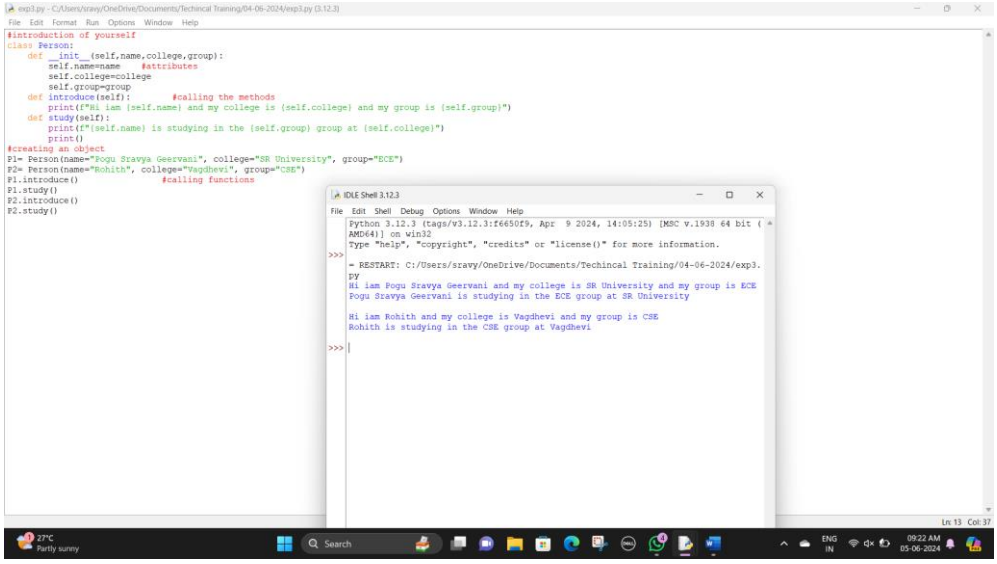
Own code for animal using super class



Creating a class make sure to print capital “p” while defining.

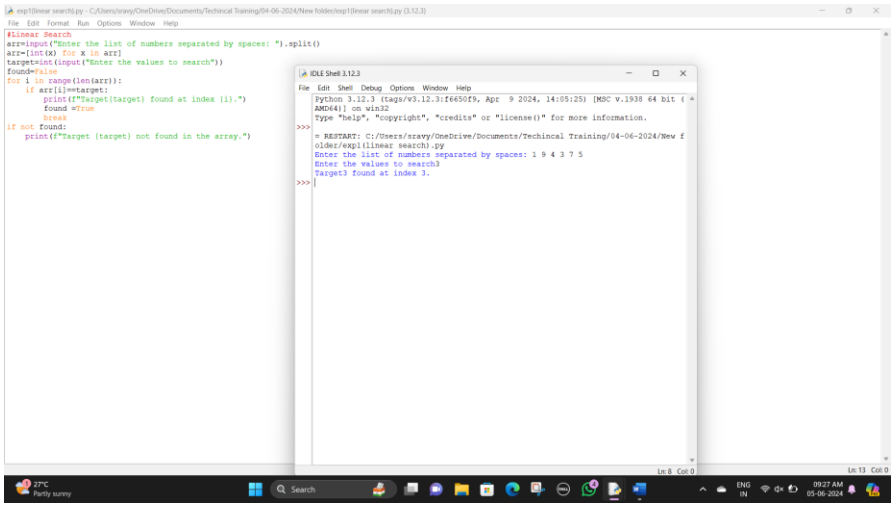


To print our own code for introduction.

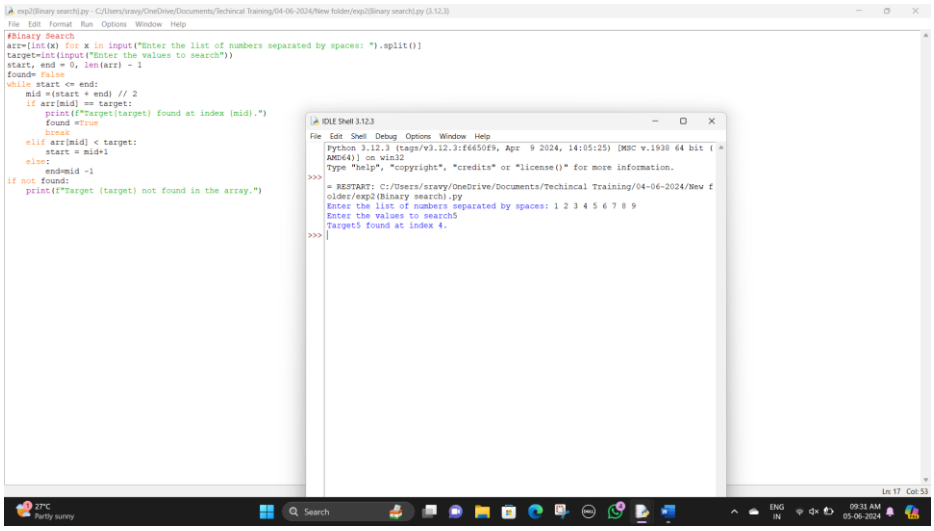


SEARCH METHODS USING PYTHON

1. **Linear Search:**
    - We can give the input as ascending or descending order.



2. **Binary Search:**
    - The input should be given in ascending order then the output will be appeared otherwise it shows as not found in array.



Dt:05-06-2024

VECHILE DESCRIPTION

Main program:

```python
from Vehicle import Bike, Car, Scooty

def main():
    Xpulse=Bike("Black",2,5)
    XUV700=Car("Navy Blue", 4, 6)
    Activa=Scooty("White",2, 0)

    XUV700.display_details()
    print("\n")
    Xpulse.display_details()
    print("\n")
    Activa.display_details()
    print("\n")

if __name__ == "__main__":
    main()
```

```
>>> %Run -c $EDITOR_CONTENT
Name(Public): Pogu Sravya Geervani
Roll Number(Protected): 2205A41L15
```

Vehicle program:

```python
from AbstractVehicle import AbstractVehicle

class Bike(AbstractVehicle):
    def display_details(self):
        print("Bike Details:")
        print(f"Colour:{self.colour}")
        print(f"Number of tyres:{self.num_tyre}")
        print(f"Gears:{self.gears}")

class Car(AbstractVehicle):
    def display_details(self):
        print("Car Details:")
        print(f"Colour:{self.colour}")
        print(f"Number of tyres:{self.num_tyre}")
        print(f"Gears:{self.gears}")

class Scooty(AbstractVehicle):
    def display_details(self):
        print("Scooty Details:")
        print(f"Colour:{self.colour}")
        print(f"Number of tyres:{self.num_tyre}")
        print(f"Gears:{self.gears}")
```

```
>>> %Run -c $EDITOR_CONTENT
Name(Public): Pogu Sravya Geervani
Roll Number(Protected): 2205A41L15
Name: Pogu Sravya Geervani
```

AbstractVechile:

```python
from abc import ABC, abstractmethod

class AbstractVehicle(ABC):
    def __init__(self,colour,num_tyre,gears):
        self.colour=colour
        self.num_tyre=num_tyre
        self.gears=gears

    @abstractmethod
    def display_details(self):
        pass
```

```
>>> %Run -c $EDITOR_CONTENT
Name(Public): Pogu Sravya Geervani
Roll Number(Protected): 2205A41L15
```

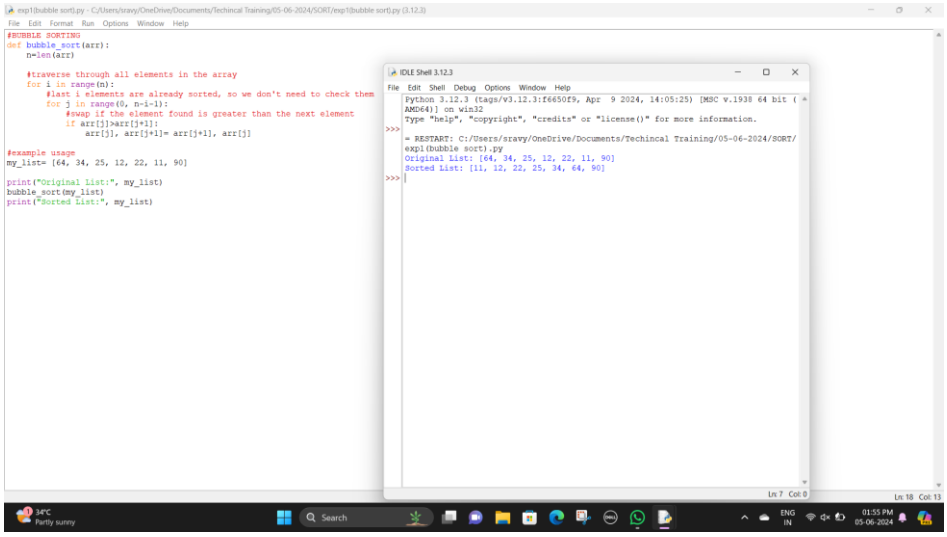
Output:

```
from Vehicle import Bike, Car, Scooty

def main():
    Xpulse=Bike("Black",2,5)
    XUV700=Car("Navy Blue", 4, 6)
```

```
Car Details:
Colour:Navy Blue
Number of tyres:4
Gears:6


Bike Details:
Colour:Black
Number of tyres:2
Gears:5


Scooty Details:
Colour:White
Number of tyres:2
Gears:0
```

SORT

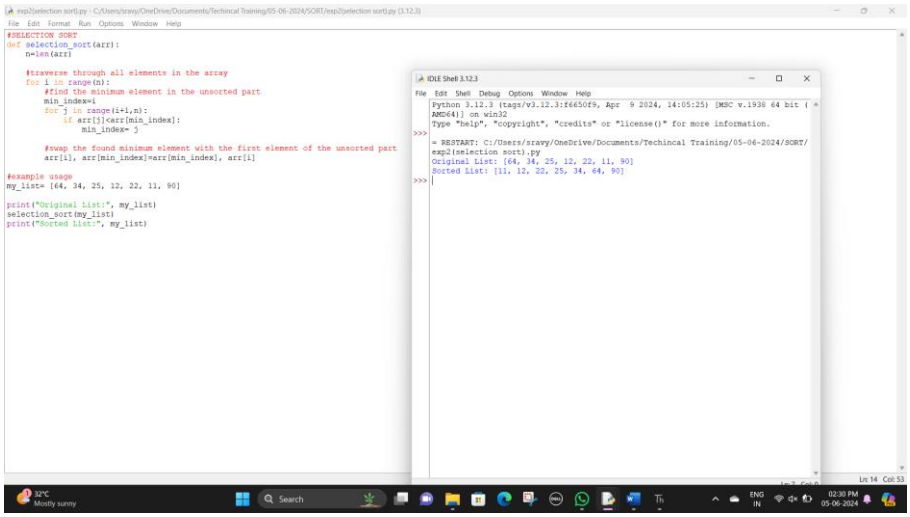
BUBBLE SORT



```
#BUBBLE SORTING
def bubble_sort(arr):
    n=len(arr)

    #traverse through all elements in the array
    for i in range(n):
        #last i elements are already sorted, so we don't need to check them
        for j in range(0, n-i-1):
            #swap if the element found is greater than the next element
            if arr[j]>arr[j+1]:
                arr[j], arr[j+1]= arr[j+1], arr[j]

#example usage
my_list= [64, 34, 25, 12, 22, 11, 90]

print("Original List:", my_list)
bubble_sort(my_list)
print("Sorted List:", my_list)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Techincal Training/05-06-2024/SORT/
exp1(bubble sort).py
Original List: [64, 34, 25, 12, 22, 11, 90]
Sorted List: [11, 12, 22, 25, 34, 64, 90]
>>>
```

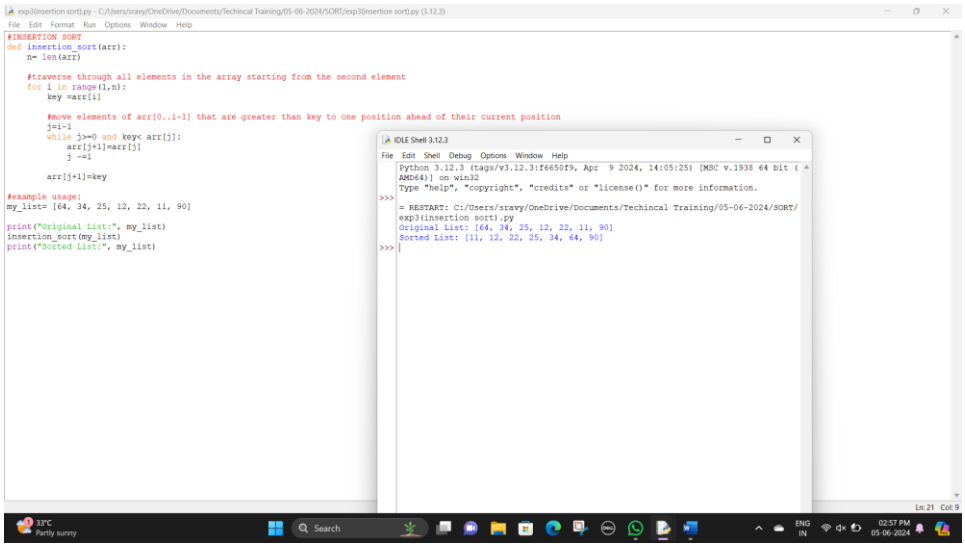
SELECTION SORT



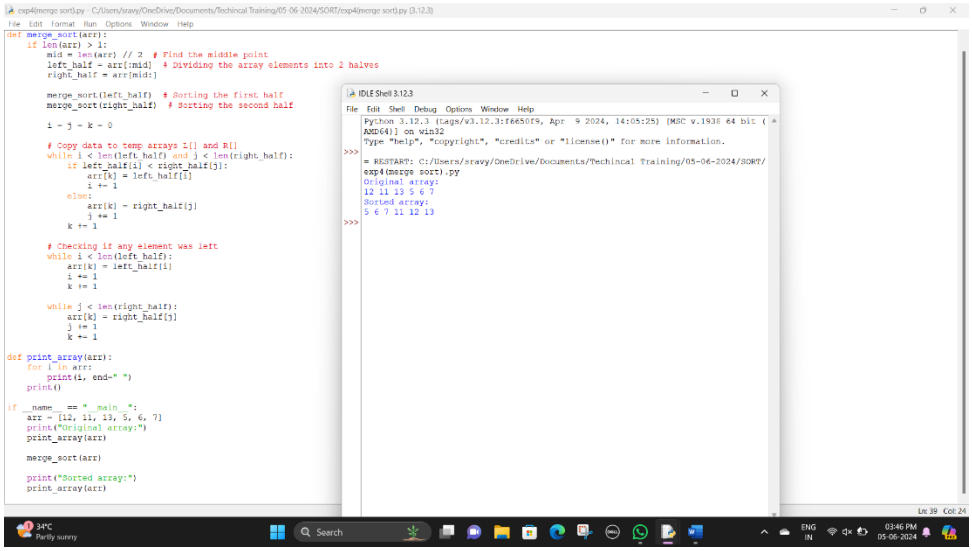
```
#SELECTION SORT
def selection_sort(arr):
    n=len(arr)

    #traverse through all elements in the array
    for i in range(n):
        #find the minimum element in the unsorted part
        min_index=i
        for j in range(i+1,n):
            if arr[j]<arr[min_index]:
                min_index= j

        #swap the found minimum element with the first element of the unsorted part
        arr[i], arr[min_index]=arr[min_index], arr[i]

#example usage
my_list= [64, 34, 25, 12, 22, 11, 90]

print("Original List:", my_list)
selection_sort(my_list)
print("Sorted List:", my_list)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Techincal Training/05-06-2024/SORT/
exp2(selection sort).py
Original List: [64, 34, 25, 12, 22, 11, 90]
Sorted List: [11, 12, 22, 25, 34, 64, 90]
>>>
```

INSERTION SORT



MERGE SORT



Dt:06-06-2024

METHOD OVERRIDING