Python Tuples

A comma-separated group of items is called a Python triple.

Tuples are an immutable data type, meaning their elements cannot be changed after they are generated.

Each element in a tuple has a specific order that will never change because tuples are ordered sequences.

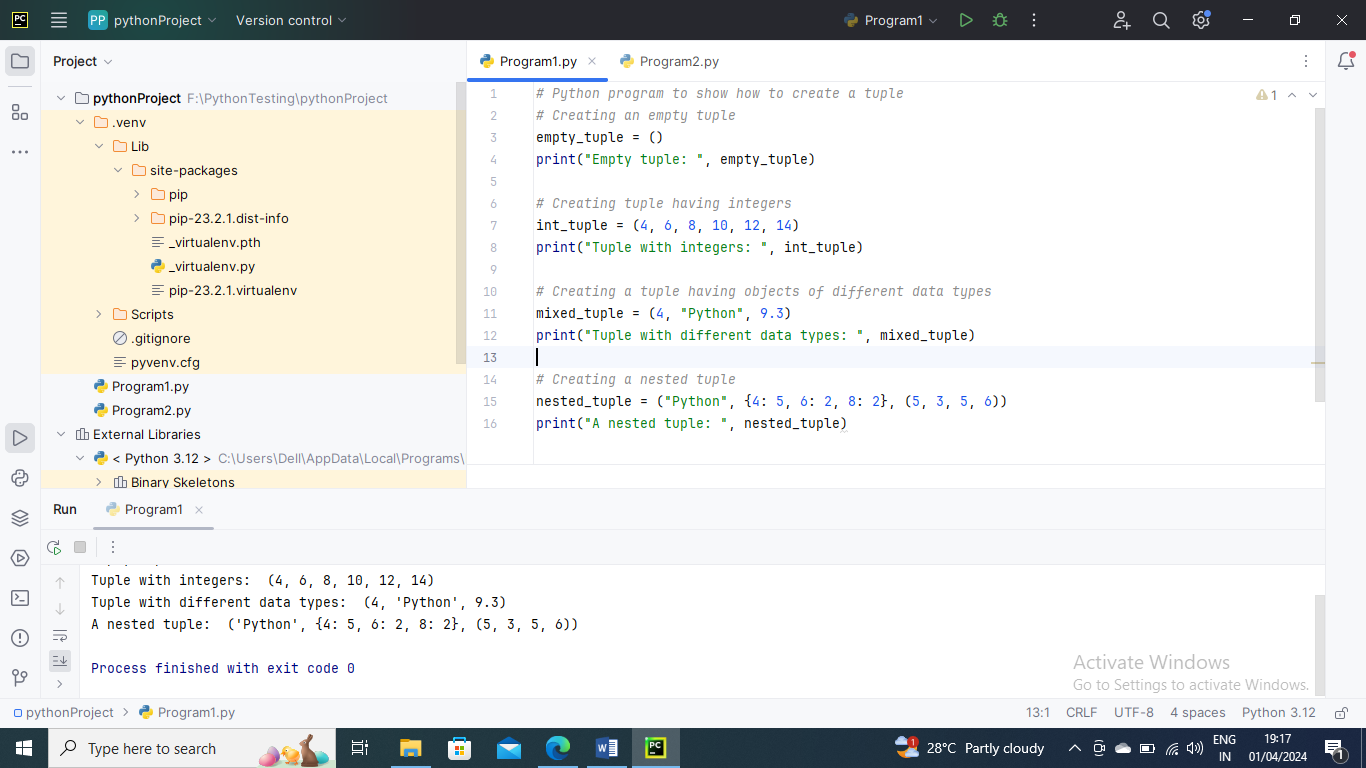
Forming a Tuple –

All the objects-also known as "elements"-must be separated by a comma, enclosed in parenthesis (). Although parentheses are not required, they are recommended.

Any number of items, including those with various data types (dictionary, string, float, list, etc.), can be contained in a tuple.

Below is the example

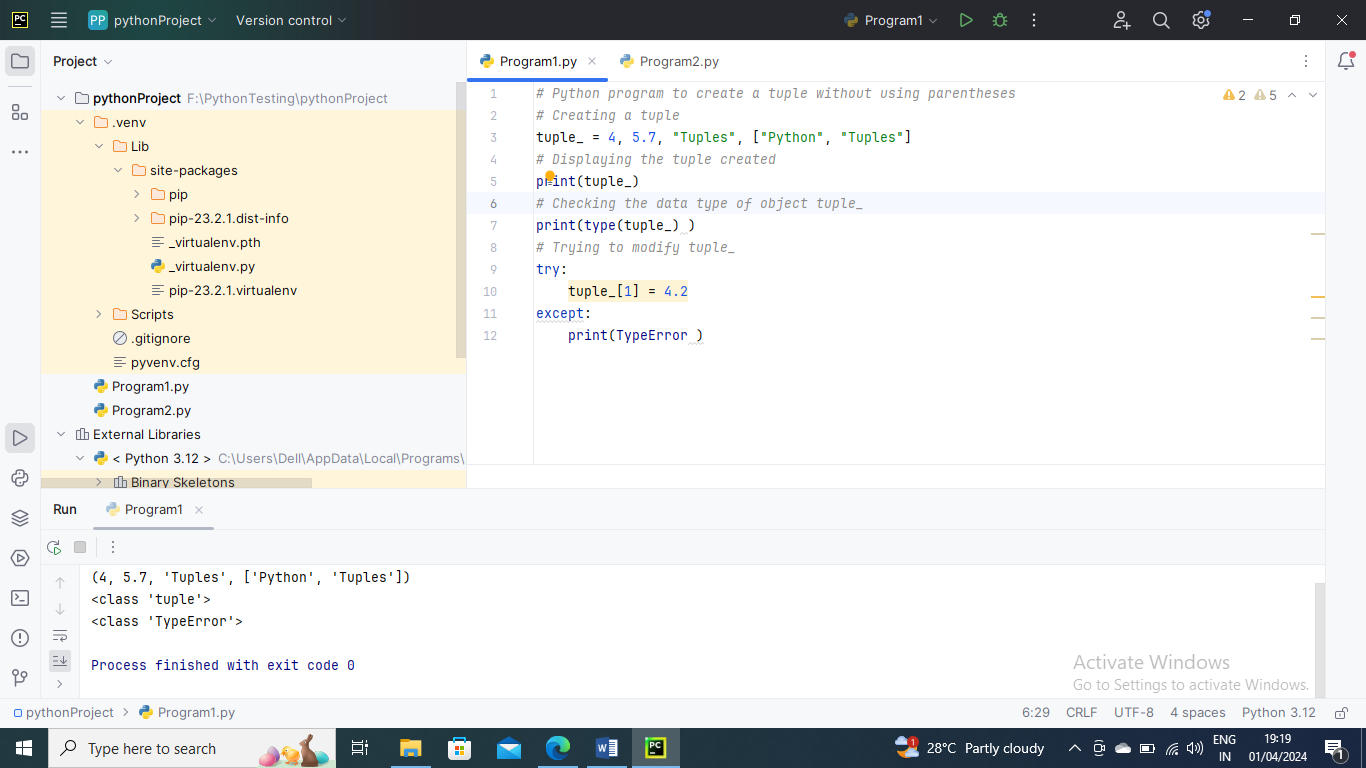
*# Python program to show how to create a tuple  
# Creating an empty tuple*empty\_tuple = ()  
print("Empty tuple: ", empty\_tuple)  
  
*# Creating tuple having integers*int\_tuple = (4, 6, 8, 10, 12, 14)  
print("Tuple with integers: ", int\_tuple)  
  
*# Creating a tuple having objects of different data types*mixed\_tuple = (4, "Python", 9.3)  
print("Tuple with different data types: ", mixed\_tuple)  
  
*# Creating a nested tuple*nested\_tuple = ("Python", {4: 5, 6: 2, 8: 2}, (5, 3, 5, 6))  
print("A nested tuple: ", nested\_tuple)



Parentheses are not necessary for the construction of multiples. This is known as triple pressing.

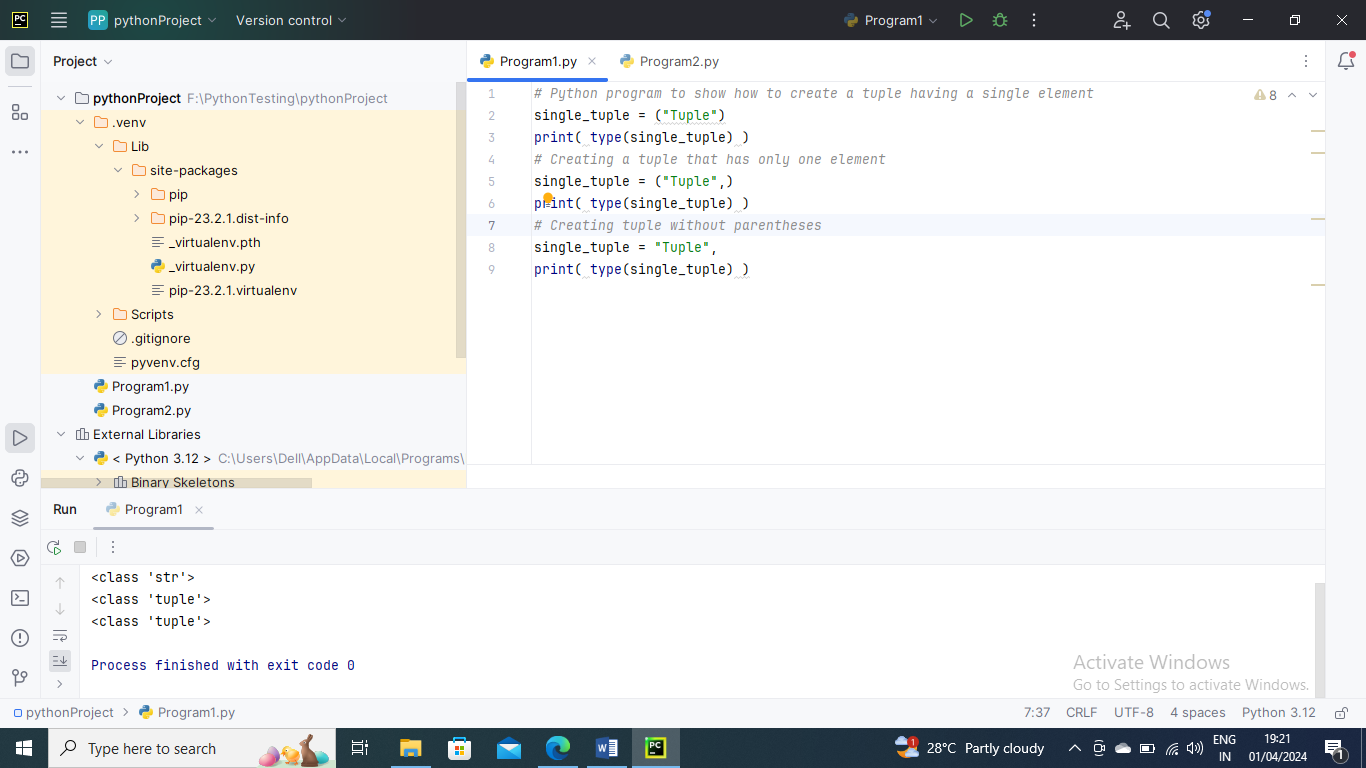
Below is the example

*# Python program to create a tuple without using parentheses  
# Creating a tuple*tuple\_ = 4, 5.7, "Tuples", ["Python", "Tuples"]  
*# Displaying the tuple created*print(tuple\_)  
*# Checking the data type of object tuple\_*print(type(tuple\_) )  
*# Trying to modify tuple\_*try:  
 tuple\_[1] = 4.2  
except:  
 print(TypeError )



Below is an example to show how to create a tuple having a single element

*# Python program to show how to create a tuple having a single element*single\_tuple = ("Tuple")  
print( type(single\_tuple) )  
*# Creating a tuple that has only one element*single\_tuple = ("Tuple",)   
print( type(single\_tuple) )  
*# Creating tuple without parentheses*single\_tuple = "Tuple",  
print( type(single\_tuple) )



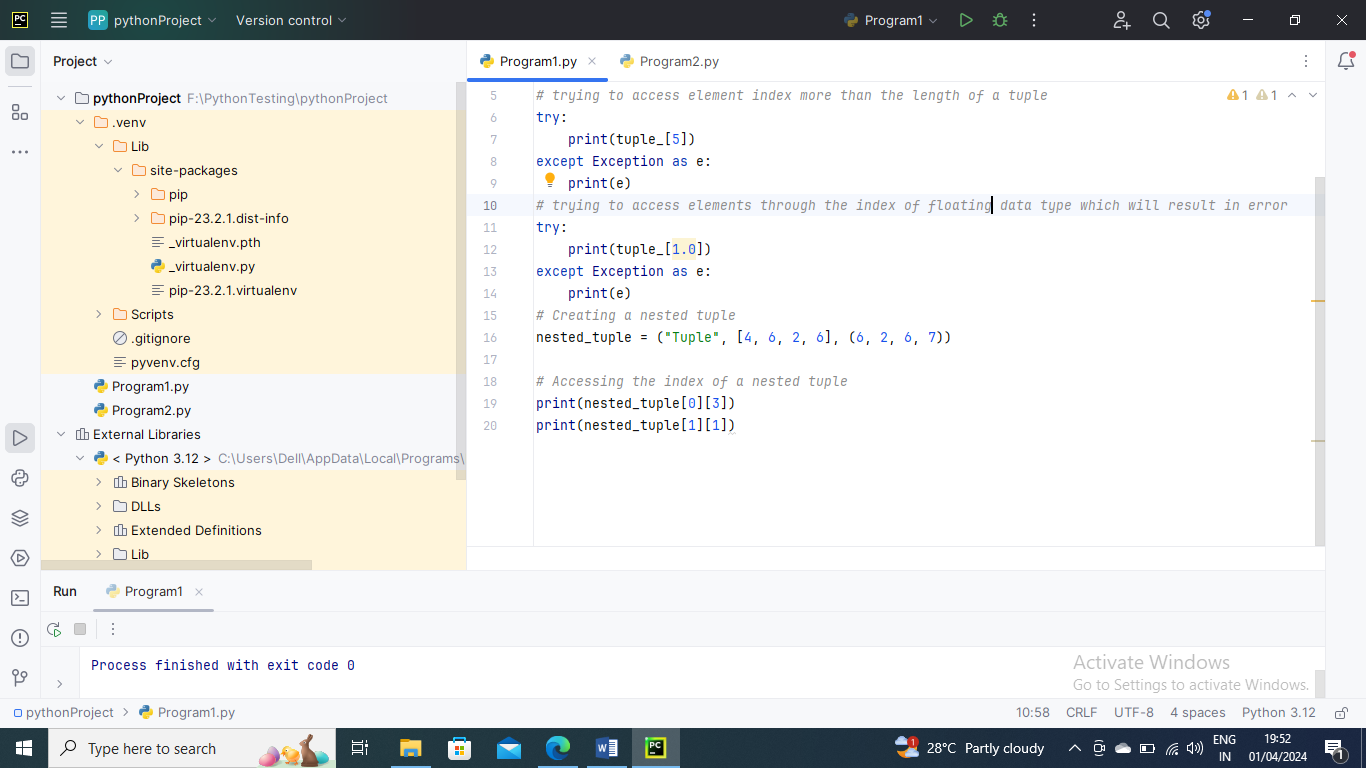
Accessing Tuple Elements –

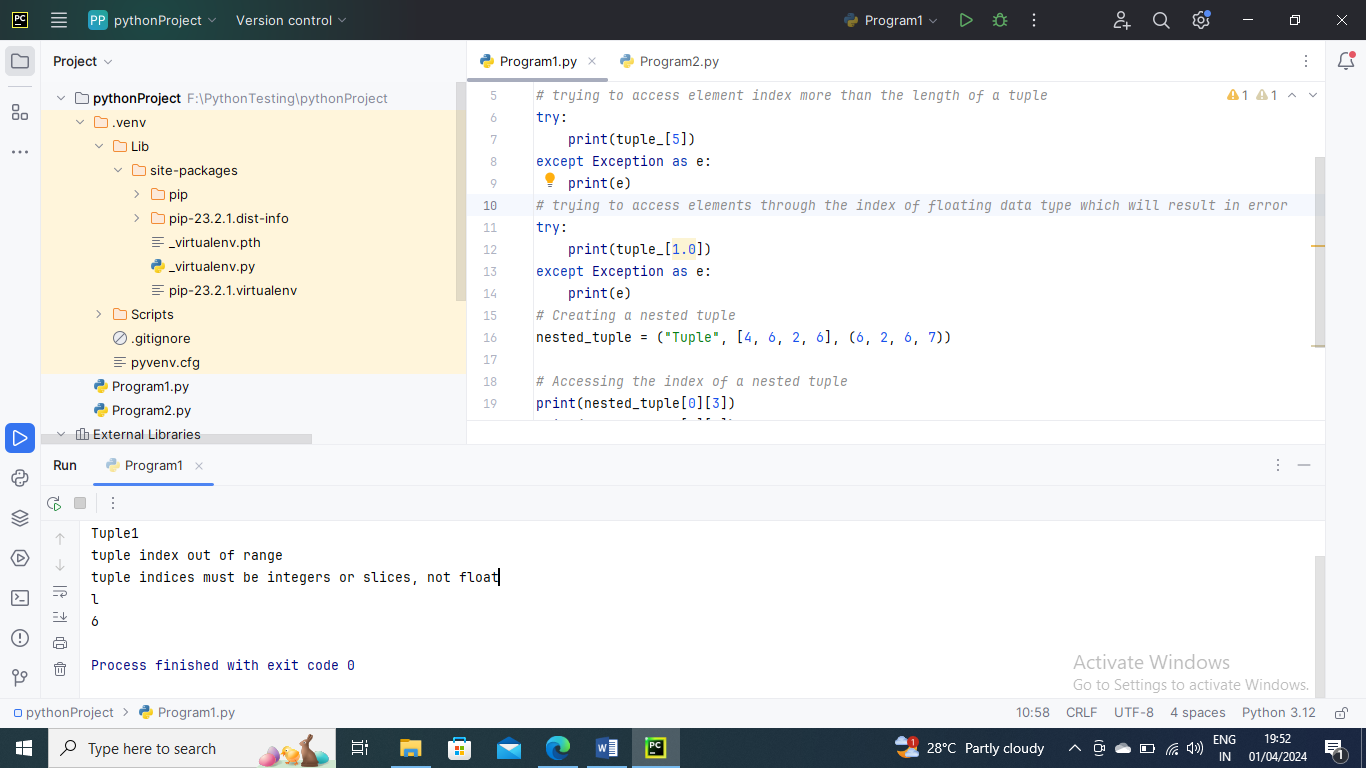
A tuple's objects can be accessed in a variety of ways.

Indexing –

We can use the index operator [] to access an object in a tuple, where the index starts at 0.

*# Creating a tuple*tuple\_ = ("Python1", "Tuple1", "Ordered", "Collection")  
print(tuple\_[0])  
print(tuple\_[1])  
*# trying to access element index more than the length of a tuple*try:  
 print(tuple\_[5])  
except Exception as e:  
 print(e)  
*# trying to access elements through the index of floating data type which will result in error*try:  
 print(tuple\_[1.0])  
except Exception as e:  
 print(e)  
*# Creating a nested tuple*nested\_tuple = ("Tuple", [4, 6, 2, 6], (6, 2, 6, 7))  
  
*# Accessing the index of a nested tuple*print(nested\_tuple[0][3])  
print(nested\_tuple[1][1])



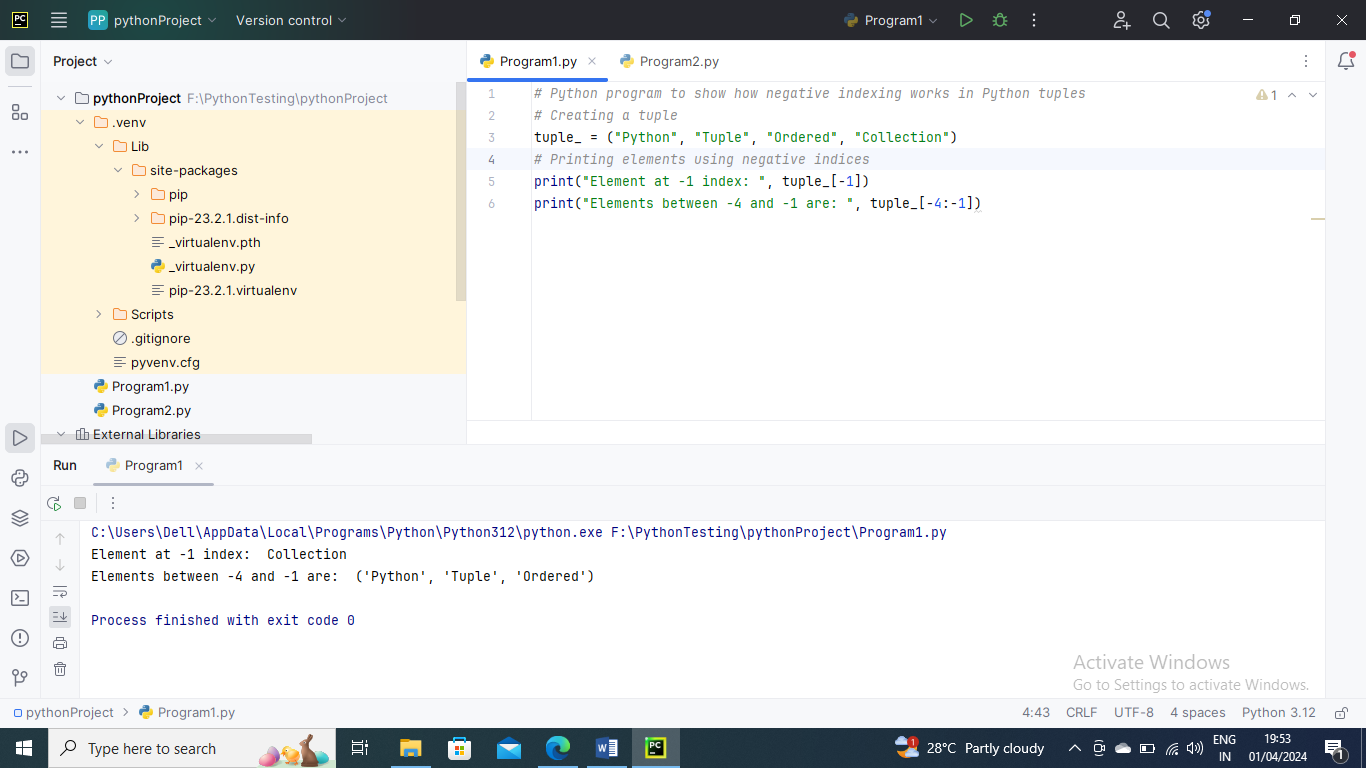


Negative Indexing –

Python's sequence objects support negative indexing.

The last thing of the assortment is addressed by - 1, the second last thing by - 2, etc.

*# Python program to show how negative indexing works in Python tuples  
# Creating a tuple*tuple\_ = ("Python", "Tuple", "Ordered", "Collection")  
*# Printing elements using negative indices*print("Element at -1 index: ", tuple\_[-1])  
print("Elements between -4 and -1 are: ", tuple\_[-4:-1])

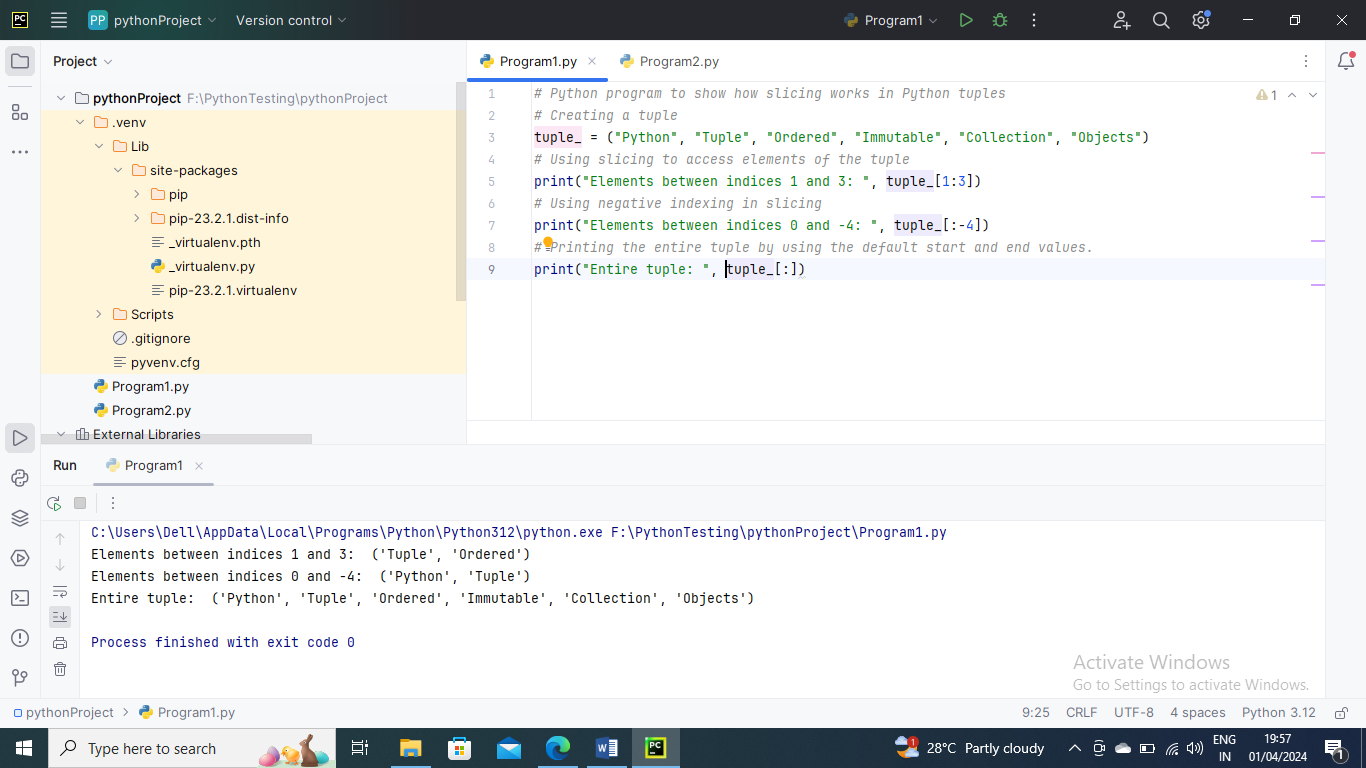


Slicing –

Tuple slicing is a common practice in Python and the most common way for programmers to deal with practical issues. Look at a tuple in Python. Slice a tuple to access a variety of its elements. Using the colon as a straightforward slicing operator (:) is one strategy.

To gain access to various tuple elements, we can use the slicing operator colon (:).

*# Python program to show how slicing works in Python tuples  
# Creating a tuple*tuple\_ = ("Python", "Tuple", "Ordered", "Immutable", "Collection", "Objects")  
*# Using slicing to access elements of the tuple*print("Elements between indices 1 and 3: ", tuple\_[1:3])  
*# Using negative indexing in slicing*print("Elements between indices 0 and -4: ", tuple\_[:-4])  
*# Printing the entire tuple by using the default start and end values.*print("Entire tuple: ", tuple\_[:])

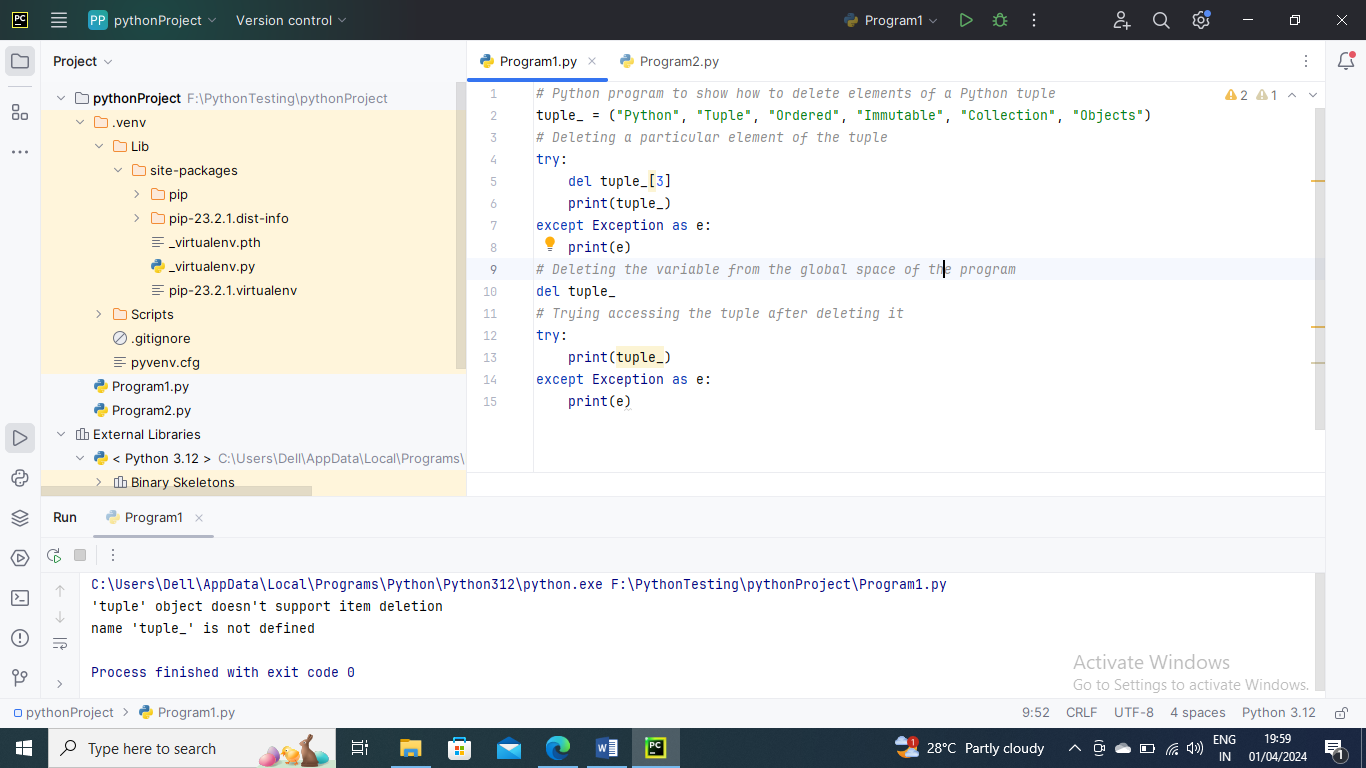


Deleting a Tuple

A tuple's parts can't be modified, as was recently said. We are unable to eliminate or remove tuple components as a result.

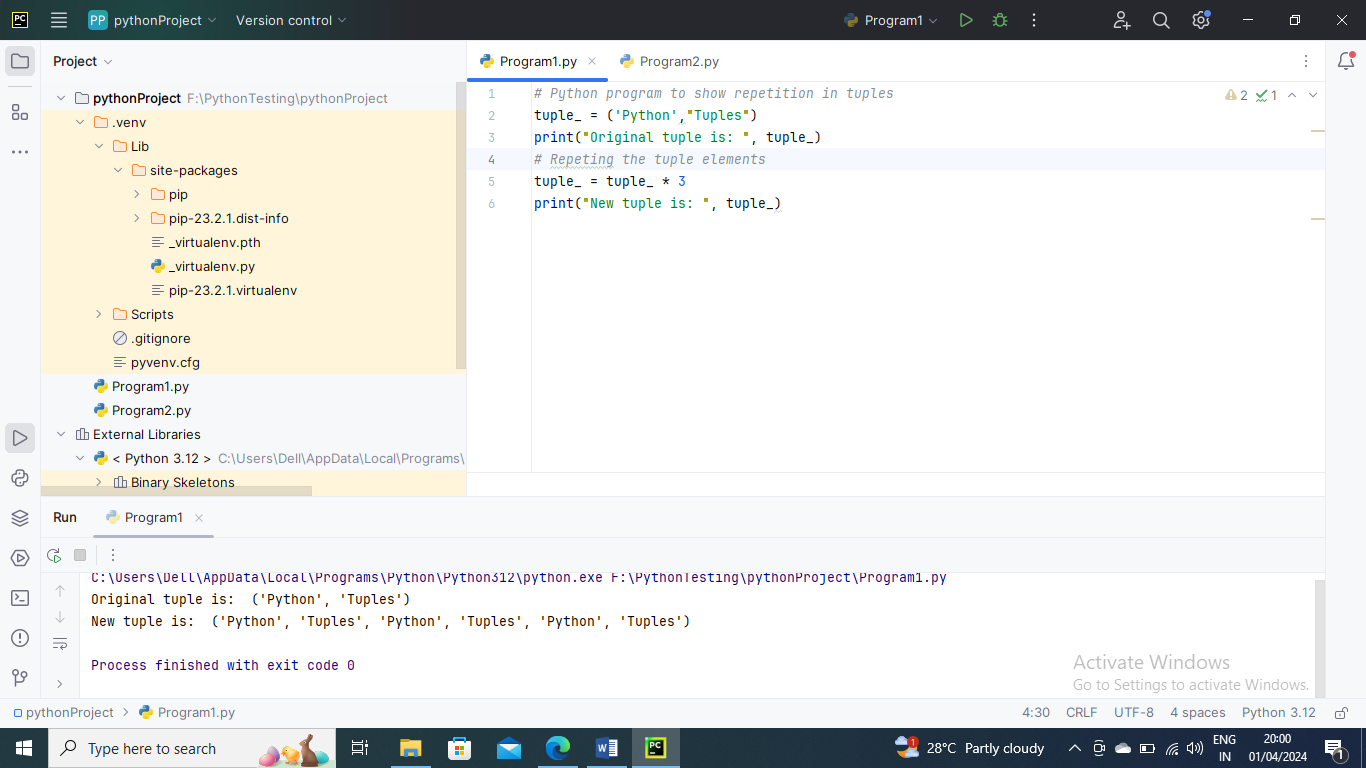
However, the keyword del can completely delete a tuple.

*# Python program to show how to delete elements of a Python tuple*tuple\_ = ("Python", "Tuple", "Ordered", "Immutable", "Collection", "Objects")  
*# Deleting a particular element of the tuple*try:  
 del tuple\_[3]  
 print(tuple\_)  
except Exception as e:  
 print(e)  
*# Deleting the variable from the global space of the program*del tuple\_  
*# Trying accessing the tuple after deleting it*try:  
 print(tuple\_)  
except Exception as e:  
 print(e)



Below is an example of repetition Typles

*# Python program to show repetition in tuples*tuple\_ = ('Python',"Tuples")  
print("Original tuple is: ", tuple\_)  
*# Repeting the tuple elements*tuple\_ = tuple\_ \* 3  
print("New tuple is: ", tuple\_)

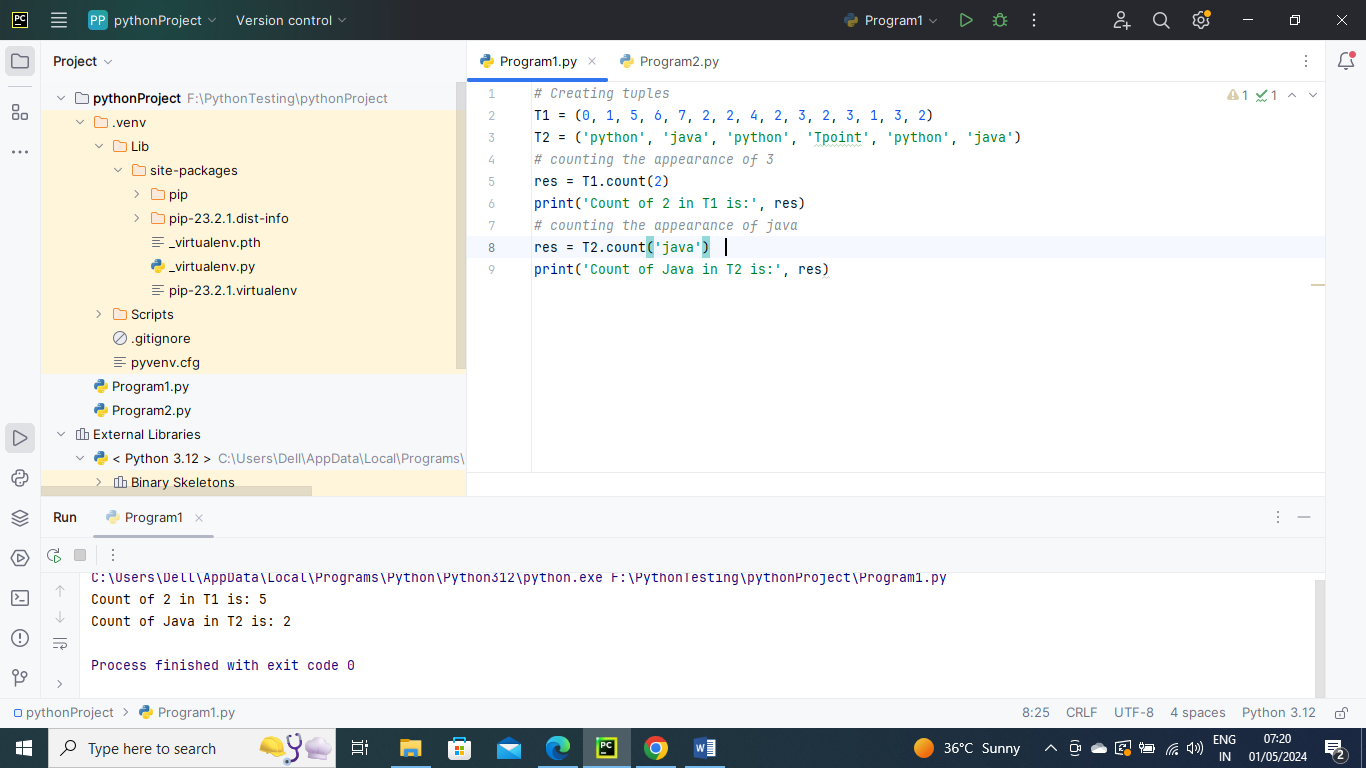


Tuple Methods –

Like the list, Python Tuples is a collection of immutable objects. There are a few ways to work with tuples in Python. With some examples, this essay will go over these two approaches in detail.

Count() Method :

The times the predetermined component happens in the Tuple is returned by the count () capability of the Tuple.

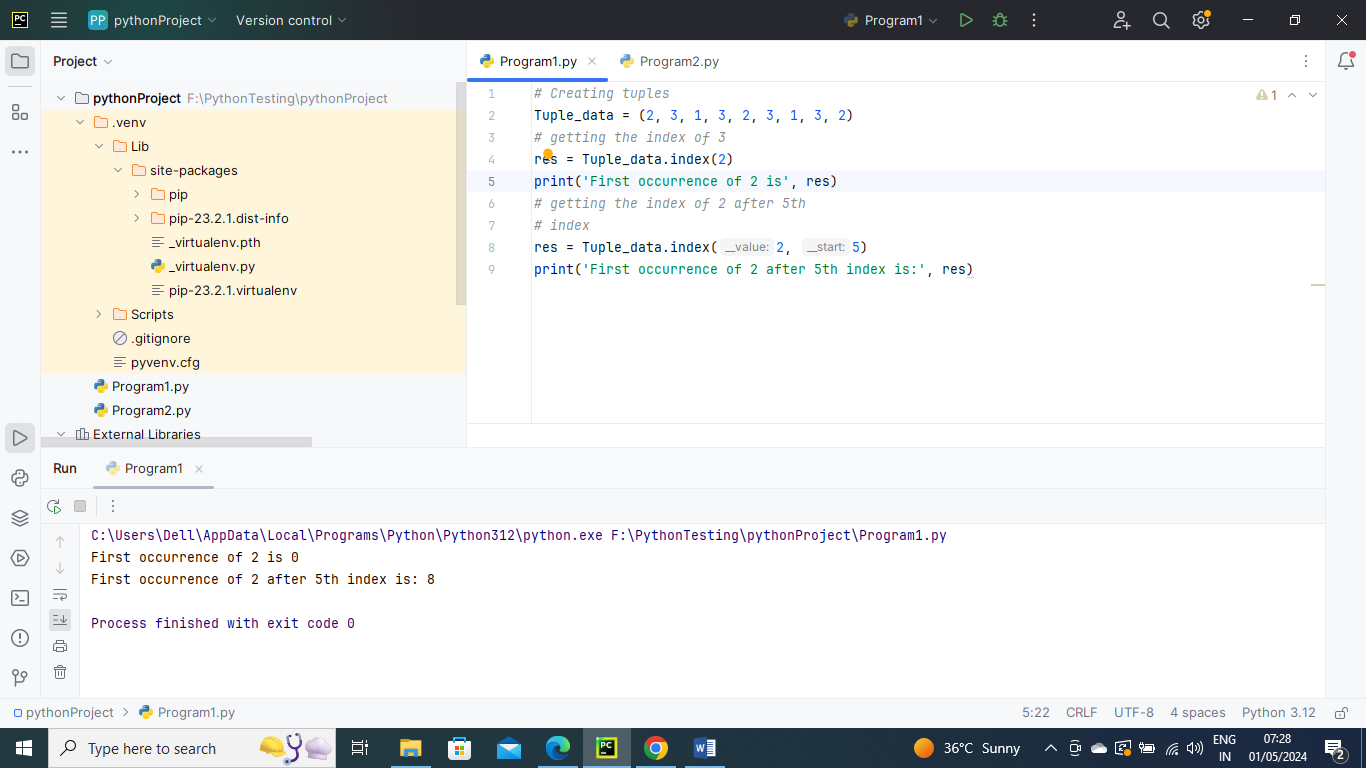


Index() Method :

The Index() function returns the first instance of the requested element from the Tuple.

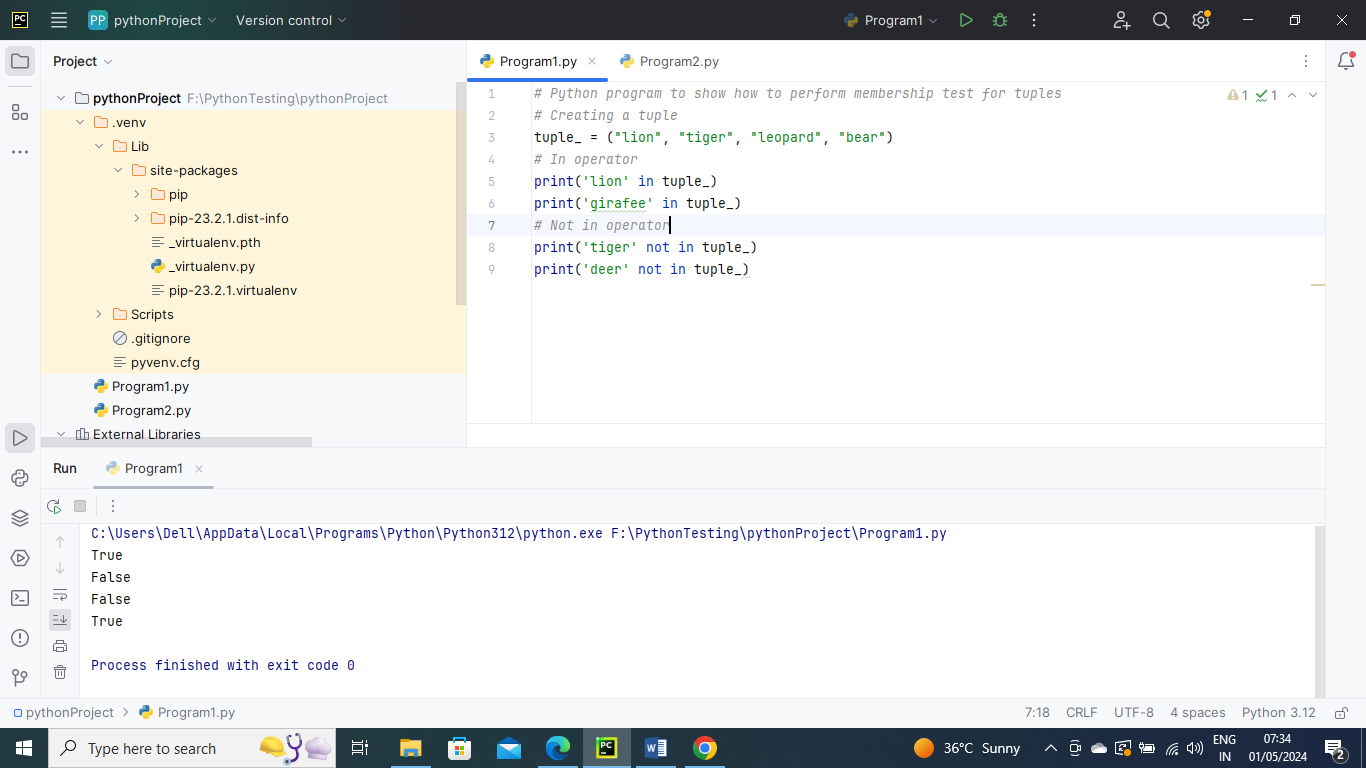
**Parameters:**

* The thing that must be looked for.
* Start: (Optional) the index that is used to begin the final (optional) search: The most recent index from which the search is carried out
* Index Method



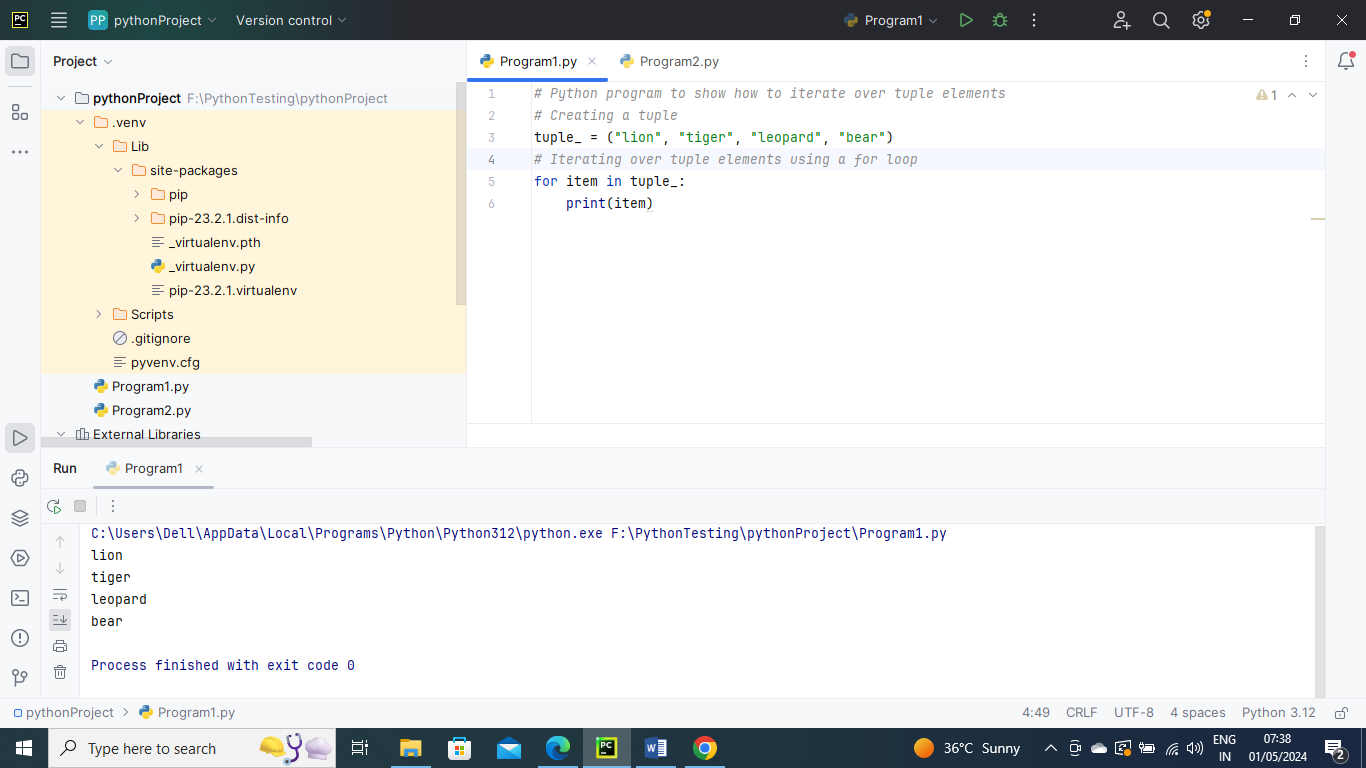
Tuple Membership Test:

Utilizing the watchword, we can decide whether a thing is available in the given Tuple.



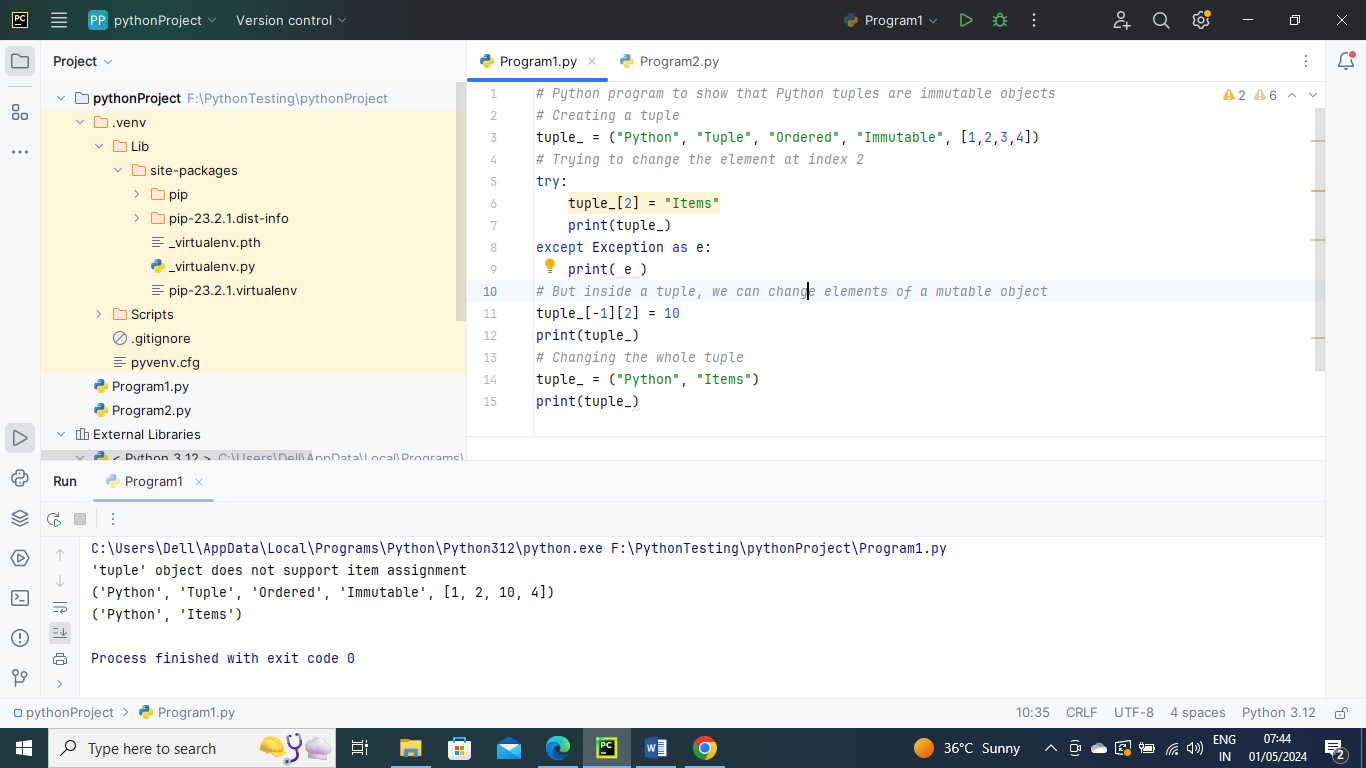
Iterating through a Tuple:

A for loop can be used to iterate through each tuple element.



Changing Tuples:

Tuples are immutable means we cannot change the values in Tuples. But we can re-assign complete tuple again.



Concatenation in Tuples:

