**Module 7)**

1. **Python – Collections, functions and Modules Accessing List**

**Theory:**

1. Understanding how to create and access elements in a list.

List is a collection of similar and dissimilar types of data. Which stores in a single variable. It is denoted by []

* List is mutable
* List is orderable
* List allows duplicate values
* List allows slicing

**how to create elements in a list:**

List = [1, 2, 44, 55, “Python”, “Database”]

print(List)

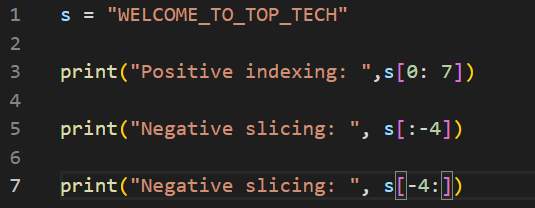
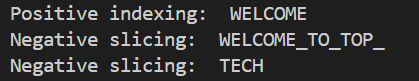
1. Indexing in lists (positive and negative indexing).

Indexing is a way to access individual elements of a list.

There are two types of list indexing:

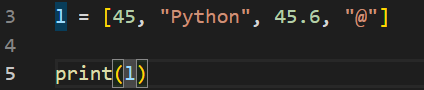
**Positive Indexing:** we can use it to access elements from start to end of the list.

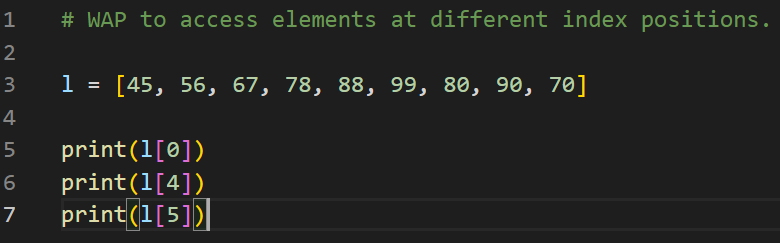
**Negative Indexing:** we can use negative indexing to access elements from the end of the list.

1. ****Slicing a list: accessing a range of elements.

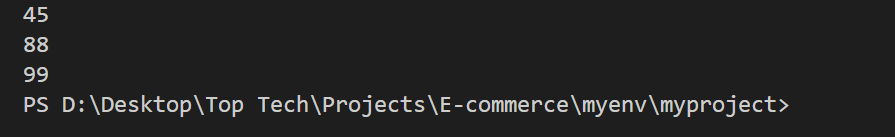
**Lab:**

1. Write a Python program to create a list with elements of multiple data types (integers, strings, floats, etc.).



1. Write a Python program to access elements at different index positions.

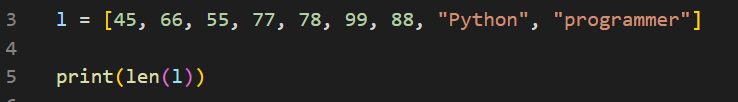
Output:



**Practical Examples:**

1. Write a Python program to create a list of multiple data type elements.

**Same as previous (1.4)**

1.  a Python program to find the length of a list using the len() function.

 Output:

**2. List Operations**

**Theory:**

1. Common list operations: concatenation, repetition, membership.

**Concatenation:** Concatenating list in python is a common operation that combines multiple list into one. **Here are several methods to achieve this.**

**L1= [45, 56]**

L2= [67, 78]

* **By using + operator**

Cancate\_list = **L1 + L2**

* **By using list extend method**

L1.extend(L2)

* **By using \* operator**

Cancate\_list = **L1 + L2**

1. Understanding list methods like append(), insert(), remove(), pop().

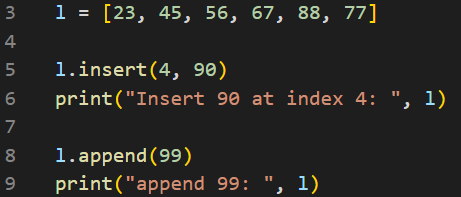
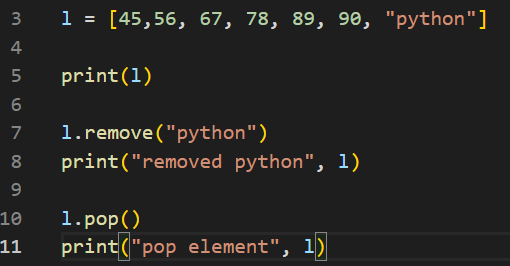
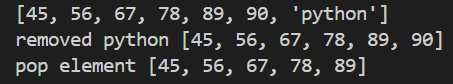
**append() : append method is used to add single element at the end of the list.**

**Insert() : To add element at the particular index we use insert method into the list.**

**remove() : To remove a value or element directly from the list.**

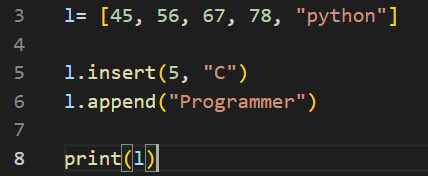
**pop(): Pop method delete last element from the given list.**

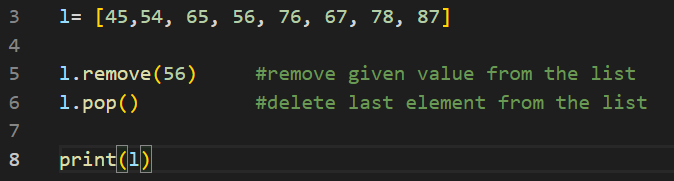
**Lab:**

1. Write a Python program to add elements to a list using insert() and append().
2.  Write a Python program to remove elements from a list using pop() and remove().

**Practical Examples:**

1. Write a Python program to update a list using insert() and append().



6. Write a Python program to remove elements from a list using pop() and remove().

**3. Working with Lists**

**Theory:**

1. Iterating over a list using loops.

Python provides several ways to iterate over a list. The simplest and the most common way to iterate over a list to use a for loop. This method allows us to access each element in the list directly. We can use range() method with for loop into python to traverse the list. This method allows us to access elements by their index, which is useful if we need to know the position of an element or modify the list in place.

Ex.

list = []

n= len(list)

for i in range(n):

print(a[i])

1. Sorting and reversing a list using sort(), sorted(), and reverse().

**sort():**

sort is a built-in method of list.

List.sort()

Sort is original list (modifies original list) - Returns none

List.sort() --- to sort into ascending order

List.sort(reverse = True) --- to sort into descending order

**sorted():**

sorted is a built-in function used for sorting iterables (list, set, tuple, dictionary, string etc.)

sorted() returns new modified sorted iterable – returns new object

By using sorted function list if string can be sorted according to their len or without lenth

**Ex.**

Name= []

Sort= sorted(name) --- sort list alphabetically

Sort= sorted(name, key = len) -- sort a list by the length

**reverse():**

reverse is a built-in method used into list.

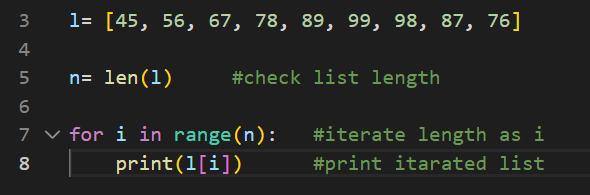
It is used to reverse the element of a list.

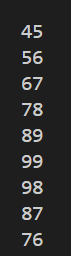
It modify list as reverse order without creating a new one.

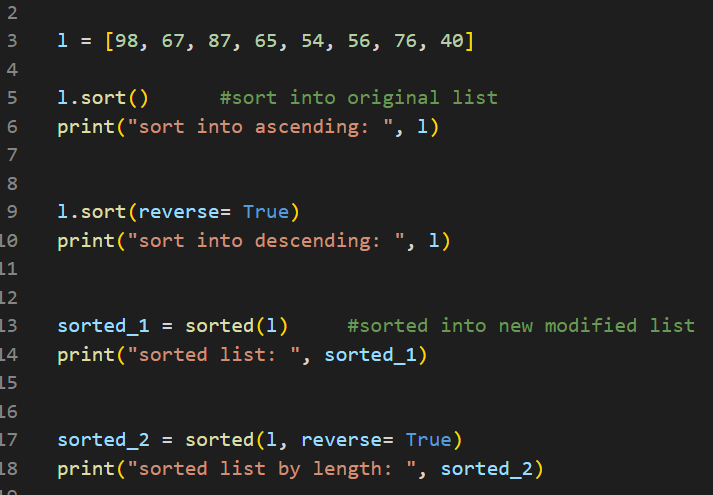
1. Basic list manipulations: addition, deletion, updating, and slicing.

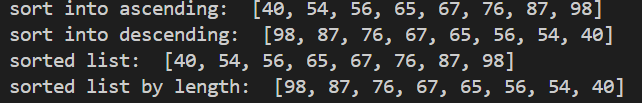
|  |  |
| --- | --- |
| **Addition** | To add element into list can be use some built-in method  Append() : add element at the end of the list.  Insert(): add element according to their index.  Extend(): can be add more than one value.  ‘+’ operator can be used to add two list. |
| **Deletion** | To delete element from list can be use list built-in method.  Pop() : pop method is used to remove last element by default from given list.  Remove() : Can be remove value from the list directly by using remove method.  Del() : del method is used to delete particular index value.  Clear() : It is used to empty the list |
| **Updating** | Modify an element by index  Modify a range of element |
| **Slicing** | Slicing refers to accessing the part of a string, divide a program into small parts.  There are two types of slicing  Positive slicing: Positive indexing starts from first to end.  Negative slicing: Negative indexing start from bottom to top.  Python slicing is an operation used to extract a portion of a string. |

**Lab:**

1.  Write a Python program to iterate over a list using a for loop.

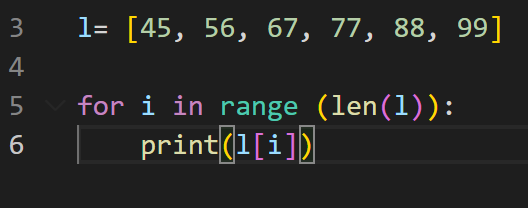
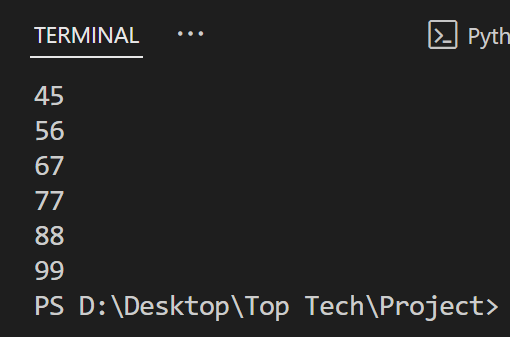


1. **** Write a Python program to sort a list using both sort() and sorted().

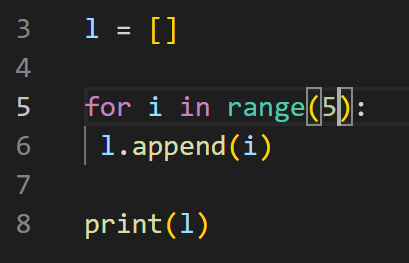
****

**Practical Examples:**

1. Write a Python program to iterate through a list and print each element.

Output:

7. Write a Python program to insert elements into an empty list using a for loop and append().

Output:

**4. Tuple**

**Theory:**

1. Introduction to tuples, immutability.

Tuple is a built-in data type in python that allows us to store a collection of similar or dissimilar types of data in single entity.

**Tuple is immutable:** it means there is not any changes directly possible into tuple because of its immutable property. If we want to do any changes into tuple list, it can be possible by convert tuple into list.

Tuple allow only two method:

**Count()**

**Index()**

1. Creating and accessing elements in a tuple.

**Creating tuple**

Tuples are created by enclosing elements within parenthesis () and separating them by commas. It can store different types of elements such as integer, float, string etc.

**Accessing elements in tuple**

Tuple allows accessing elements by using positive and negative indexing. Slicing and iteration.

* **By Index:**  it uses square [] bracket to access the elements. This types of indexing starts from zero.

**Ex.**

T= [56, 66, 77]

print(T[0])

* **Negative Indexing:** Access elements from the end of the tuple using negative index.

**Ex.**

T= [56, 66, 77]

print(T[-1])

1. Basic operations with tuples:

**Concatenation**: Concatenation operation can be possible by using ‘+’ operator between two or more tuple list.

T= ()

T1 = ()

Concate = T + T1

**Repetition:** Tuple supports repetition using \* operator. It creates a new tuple by repeating the elements of an existing tuple with a specified number/ times

T= (4, 5, 6)

Repetition = T\*4

**Membership:** We can check that member/element is present or not in tuple by using ‘in’ or ‘not in’ keyword.

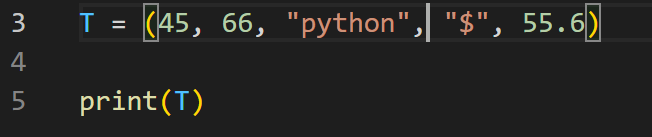
T= (45, 56, 66, 77, 88)

56 in T o/p = True

67 not in T o/p = False

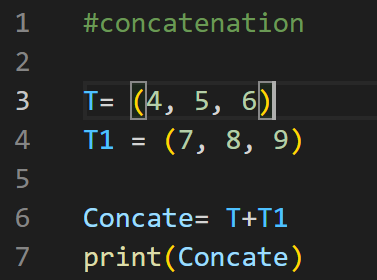
**Lab:**

1. Write a Python program to create a tuple with multiple data types.

Output:



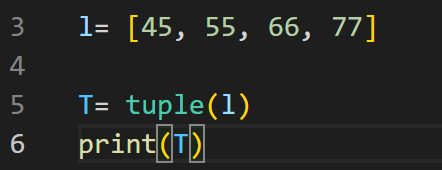
1. Write a Python program to concatenate two tuples.

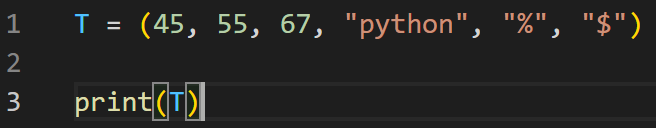
Output:



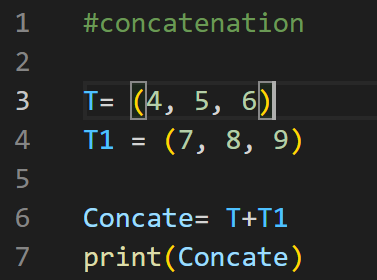
**Practical Examples:**

1. Write a Python program to convert a list into a tuple.

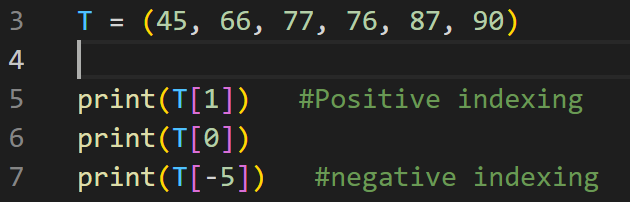
Output:

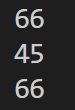
1. Write a Python program to create a tuple with multiple data types.

Output:

1. Write a Python program to concatenate two tuples into one.

Output:

1. Write a Python program to access the value of the first index in a tuple.

Output:

**5. Accessing Tuples**

**Theory:**

1. Accessing tuple elements using positive and negative indexing.

**Accessing the element using positive indexing:**

In positive indexing, first element of the tuple has an index of 0 and the second element has an index of 1.

**Accessing the element using negative indexing:**

In negative indexing, first element of the tuple has an index of -1 and the second element has an index of -2.

**Ex.**

T= (44, 55, 66, 67)

Positive indexing: T[0]

Negative indexing: T[-1]

1. Slicing a tuple to access ranges of elements.

Slicing in tuples is a technique to extract a range or a portion of elements from a tuple. It is done by specifying a start, stop and step value using the slicing syntax

tuple[start : Stop : step]

**Start:** It show the index of first element range

**Stop:** It show the index where slicing ends.

**Step:** It show the increment/decrement between the elements

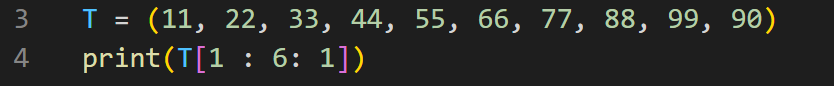
Slicing a Tuple to access the range of element: Slicing creates a new tuple containing the specified range of elements without modifying the original tuple.

Ex: T = (45, 56, 67, 78, 89)

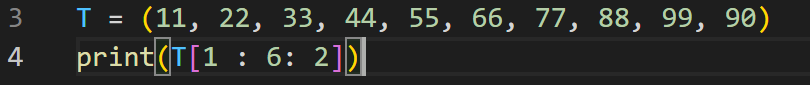
Positive indexing: T[0 : 4: 1]

Negative indexing: T[-4: -1 : 1]

**Lab:**

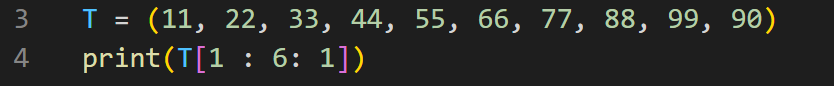
1.  Write a Python program to access values between index 1 and 5 in a tuple.

Output:

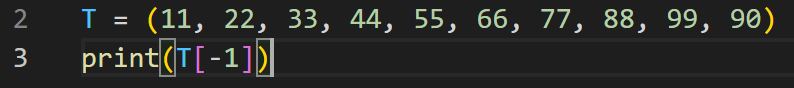
1. ****Write a Python program to access alternate values between index 1 and 5 in a tuple.

 Output:

**Practical Examples:**

1. Write a Python program to access values between index 1 and 5 in a tuple.

 Output:

6. Write a Python program to access the value from the last index in a tuple. 

Output:

**6. Dictionaries**

**Theory:**

1. Introduction to dictionaries: key-value pairs.

Dictionary is a collection data type in python. We can store multiple data value in key value pairs. Where the keys must be unique and the values can be any data type (including list, tuples etc.)

Key + Value = Item

* Items are mutable (changeable)
* Items are ordered
* Items don’t allow duplicate keys.
* Dictionary are shown as {} curley braces.
* Dictionary doesn’t allow duplicate key.
* It means same or different types of data can be stored into different keys.
* List can be stored into dictionary. {[ \_ \_ ]}
* Dictionary can also be stored into list. [{ \_ \_ }]

1. Accessing, adding, updating, and deleting dictionary elements.

We can access the values in a dictionary by using the corresponding key. There are a few different ways to access dictionary elements.

1. Using the key directly
2. By using built-in method

Add elements into dictionary

If the key does not exist, we can directly assign a value to the key using square bracket [].

D[‘city’] = “Ahemdabad”

Print(D)

Update elements into dictionary

The .update() method allows us to add multiple items from another dictionary.

d.update({‘country’ : ‘India’, ‘Name’ : ‘XYZ’)

print(d)

Delete elements into dictionary

The .delete() method allows us to add multiple items from another dictionary.

1. Dictionary methods like keys(), values(), and items().

**Key:** A key is unique identifier for a value in a dictionary. It must be immutable and unique within dictionary.

* Use **key()**  **to get all key in the dictionary.**

**Value:** Value is the data or information associated with a key. It can be of any types of data such as string, int, float, etc.

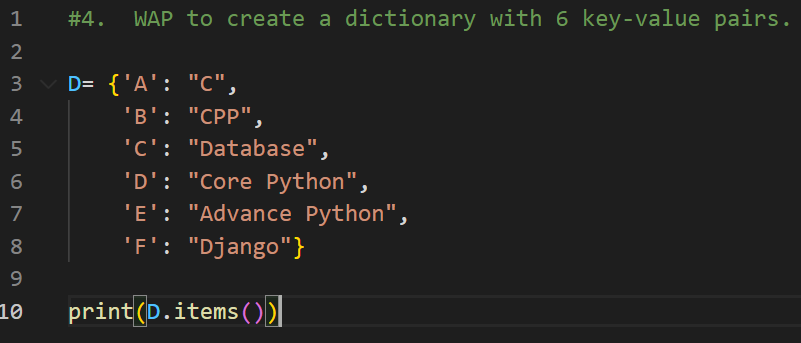
* Use **values() to get all values in the dictionary**

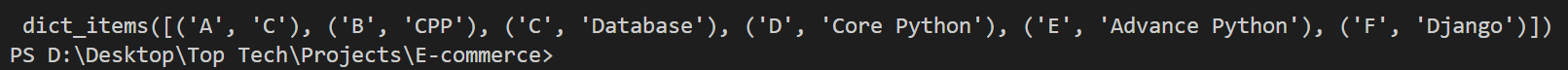
**Items:** Items are the key-value pairs as a whole.

* Use **items() to get all key-value pairs as tuple.**

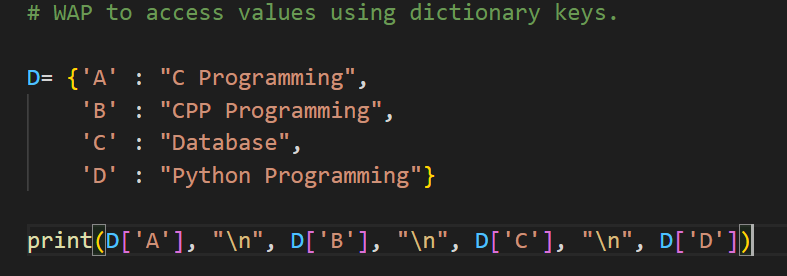
**Lab:**

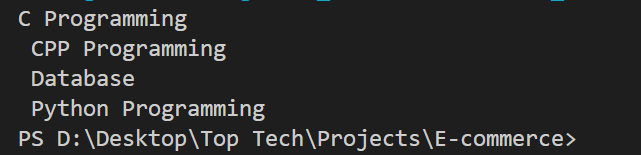
1. Write a Python program to create a dictionary with 6 key-value pairs.

Output:

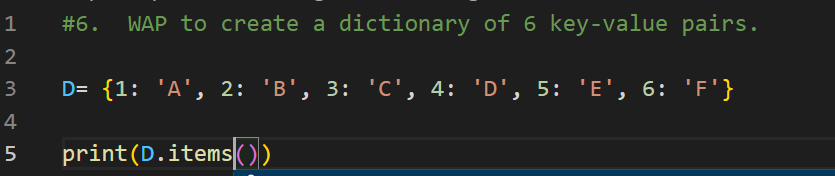


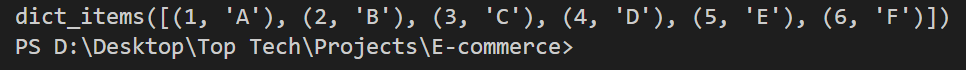
1. Write a Python program to access values using dictionary keys.

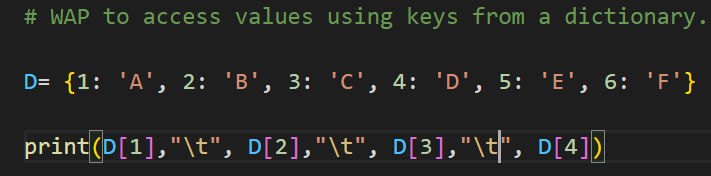
Output:

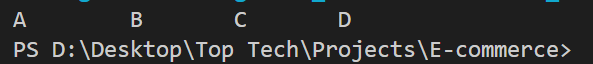


**Practical Examples:**

1. Write a Python program to create a dictionary of 6 key-value pairs.

Output:

1. Write a Python program to acces s values using keys from a dictionary.

 Output:

**7. Working with Dictionaries**

**Theory:**

1. Iterating over a dictionary using loops.

To iterate over a dictionary, we can iterate it through by using key, value or by using key-value pair that is known as item.

**Iterate through value:**

All values of python dictionary can be iterate by using the **.values()** method. This method allows us to display each value in the dictionary without explicitly referencing the corresponding keys.

**Iterate through key:**

All keys of python dictionary can be iterate by using the **.keys()** method.

**Iterate through both keys and values:**

All keys and values of python dictionary can be iterate by using the built-in **.items()** method. We can access both keys and items at the same time. Items() method returns the view object that contains the key-value pair as tuple.

1. Merging two lists into a dictionary using loops or zip().

In python, we can merge two lists into a dictionary, where one list becomes the keys and the other list becomes the corresponding values. This can be done using loops or the zip() function.

**By using loops:** We can iterate through both lists simultaneously and create a dictionary by assigning elements from one list as keys and the other list as their corresponding values.

**Ex:** Keys= [‘a’, ‘b’, ‘c’]

Values= [45, 55, 65]

merge\_dict= {}

for i in range(len(keys)):

merge\_dict[keys[i]]= values[i]

print(merge\_dict)

**By using zip:** The zip() function pairs elements from two lists together, creating an iterable of tuples. This can be directly converted into a dictionary.

**Ex.** Keys= [‘a’, ‘b’, ‘c’]

Values= [45, 55, 65]

Merge\_dict= dict(zip(keys, values))

Print(Merge\_dict)

1. Counting occurrences of characters in a string using dictionaries.

Counting the occurrences of character in a string using dictionaries means calculating how many times repeat the character appears in a given string and storing this information in a dictionary.

In this dictionary, each key represents a unique character from the string and the value associated with that key is the number of times that character appears.

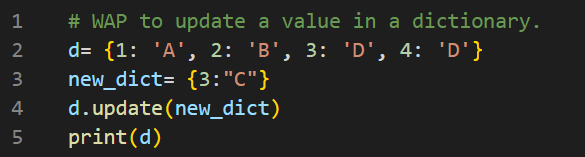
**How it works:**

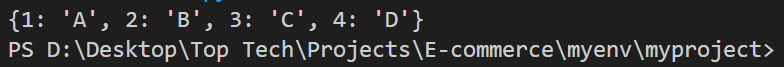
1. Start with an empty dictionary.
2. Loop to count each character in the string.
3. **For every character**

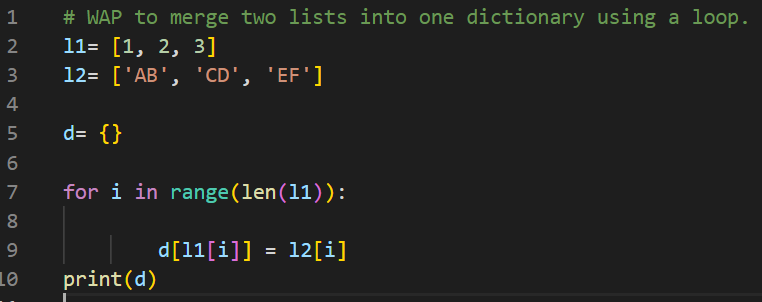
* If the character is already present in the dictionary, increament its count by 1.
* If the character not in the dictionary, add it as a key with the initial value of 1.

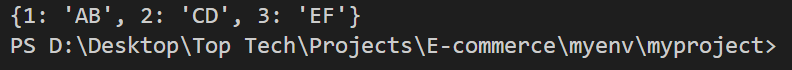
1. Once the loop complete, the dictionary contains all character from the string as keys and as their values.

**Lab:**

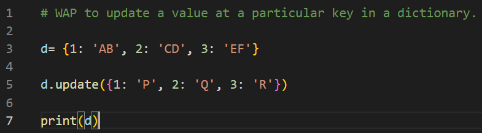
1.  Write a Python program to update a value in a dictionary.

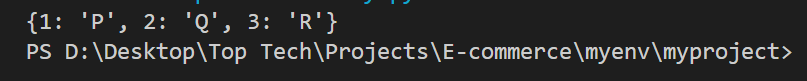
 Output:

1. **** Write a Python program to merge two lists into one dictionary using a loop.

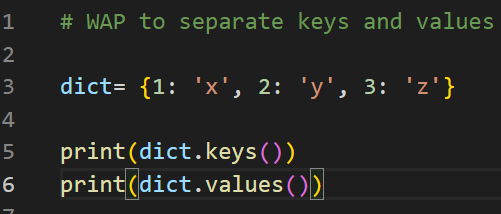
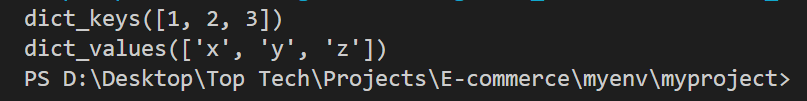
**Output:**

**Practical Examples:**

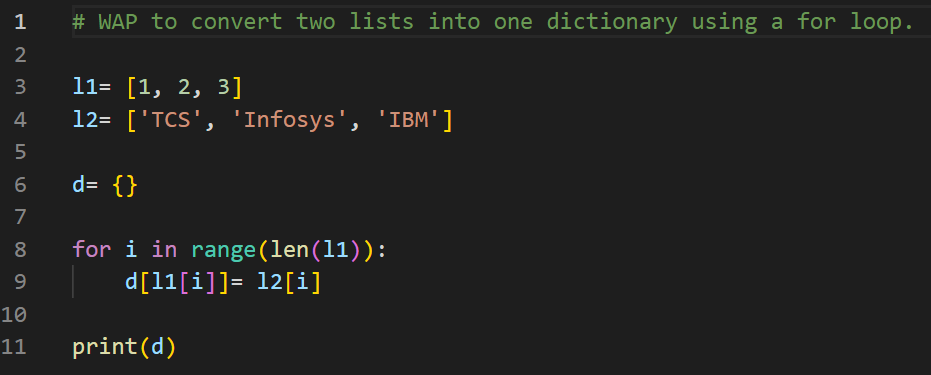
1. Write a Python program to update a value at a particular key in a dictionary.

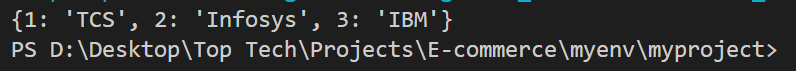
Output:

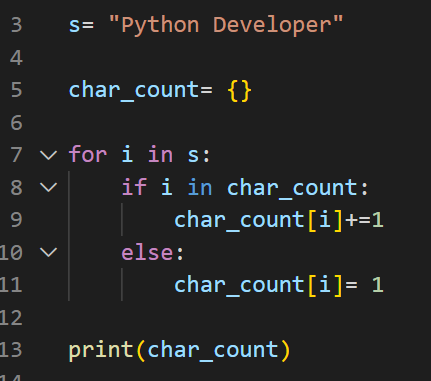
1. Write a Python program to separate keys and values from a dictionary using keys() and values() methods.

Output:

1. Write a Python program to convert two lists into one dictionary using a for loop.

Output:



9. Write a Python program to count how many times each character appears in a string.   
Output:  


**8. Functions**

**Theory:**

1. Defining functions in Python.

A Function in python is a block of reusable code that performs a specific task. This helps to organize a program into smaller, logical blocks, make the code more readable, maintainable and reusable.

**Key points :**

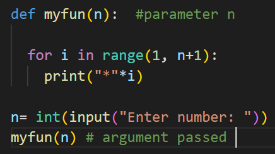
* A function consists of a group of related statements that is designed to carries out some well-defined task.
* It helps to organize code into logical blocks, simplifying complex programs.
* Functions act as subprogram or part of a program that helps us divide a large statement into smaller parts.
* It improves code efficiency, readability and reduce redundancy by enabling code reuse.

1. Different types of functions: with/without parameters, with/without return values.

Functions in python can be categorized based on two key factors:

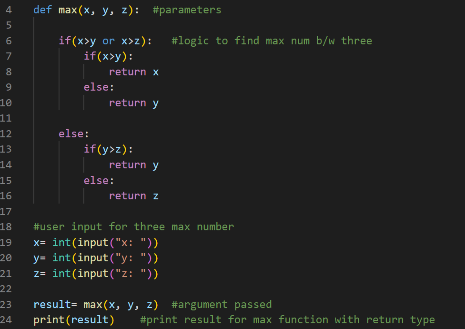
* Functions that accept parameter
* Functions that return a value
* **Functions with parameter without return:**

These functions accept parameter values (argument) to perform their tasks. However do not send any value back to the caller. It is commonly used for displaying output or modifying external data.

 **Ex:**

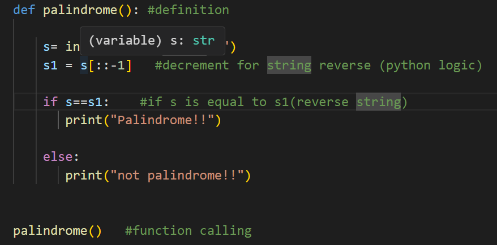
* **Functions with parameter with return:**

These functions accept parameter values (argument) to compute a result and sent it back to the caller by using return statement. It allows functions to return value for further processing.

**Ex:**

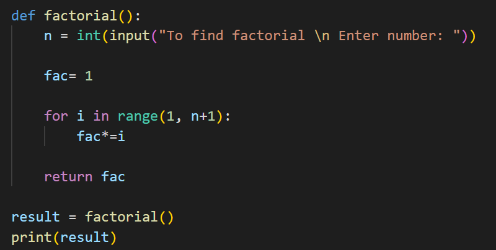
* **Functions without parameter without return:**

These functions do not accept any parameter value as argument. But do not send any value back to the caller or not to process the return statement.

**Ex:**

* **Functions without parameter with return**

These functions do not accept any parameter value as argument. these is used to perform their task and return the value back to the caller using return the statement.

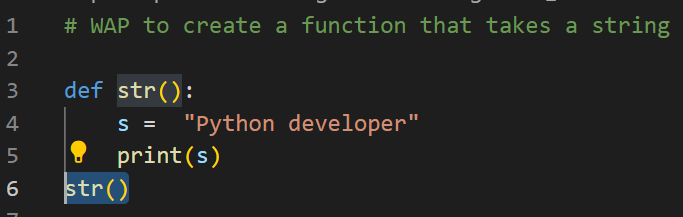
**Ex:**

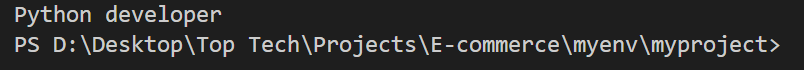
1. Anonymous functions (lambda functions).

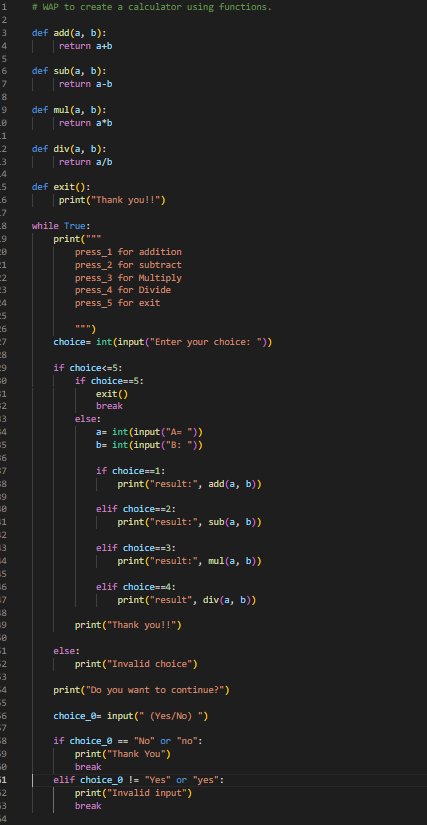
In python normal functions are defined by using def keyword, similarly the functions that are defined as by without a name and defined using lambda keyword are known as anonymous functions. This anonymous functions are also referred to as lambda functions.

**Lab:**

1. Write a Python program to create a function that takes a string as input and prints it.

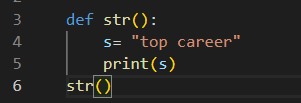
Output:

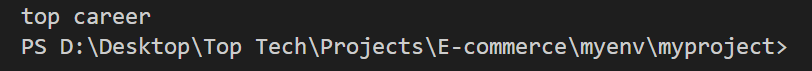


1. Write a Python program to create a calculator using functions.

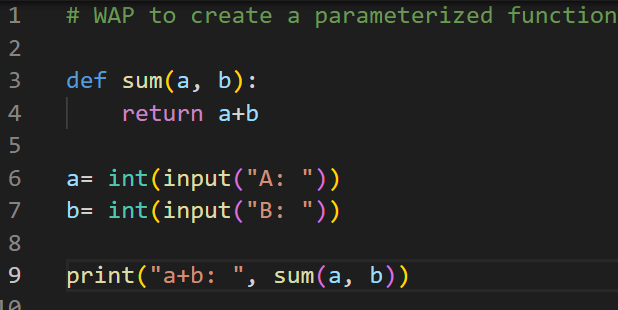
**Practical Examples:**

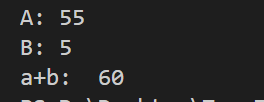
1. Write a Python program to print a string using a function.

Output:

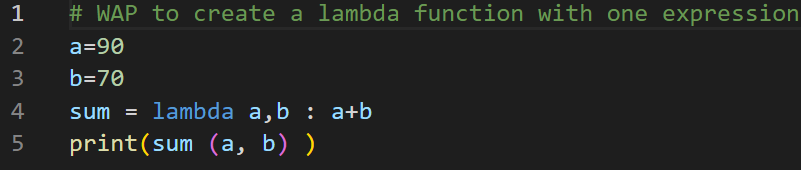


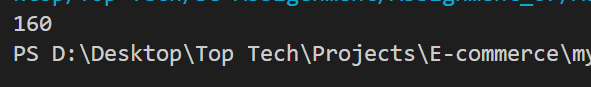
1. Write a Python program to create a parameterized function that takes two arguments and prints their sum.

Output:

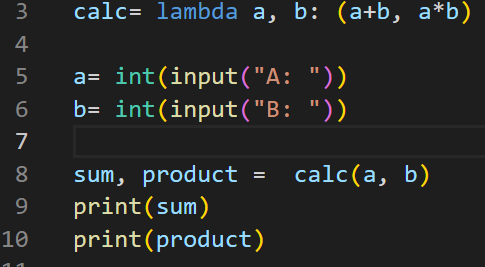


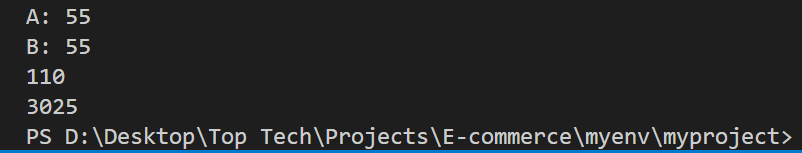
1. Write a Python program to create a lambda function with one expression.

Output:



1. Write a Python program to create a lambda function with two expressions.

Output:



**9. Modules**

**Theory:**

1. Introduction to Python modules and importing modules.

A module in python is essentially a file that contains python code. This could include functions, classes, variables and executable statements. Developers use modules to organize code into reusable and manageable components, reducing redundancy and promoting better organization.

**Key benefits of using modules:**

* **Code reusability:** Modules allow **developers** to reuse code in multiple projects or files.
* **Code organization:** by diving a large program into smaller, logical parts. It becomes easier to read and maintain.
* **Standardized functionality:** Python standard library provides a collection of built-in modules to perform various task.
* **Namespace management:** Modules provide their own namespace, which help avoid naming conflicts and encapsulate functionality.

**Types of module:**

**Standard library modules:**math(provide mathematical functions and constant)  
os (interact with the operating system)   
random (Generates random numbers)  
Custom modules

**Custom Modules:**Developers can create their own modules to encapsulate functionality specific to their applications.  
(A module is simply .py file containing into python code)

**Third party modules:**These are community-developed modules that can be installed using pip.  
**Ex.**

|  |  |
| --- | --- |
| **pip install requests** | Simplifies making http requests in python(get, post, put, delete). It is widely used for interacting with apis & fetching web data. |
| **pip install django** | A high-level web framework for building robust, scalable web applications. |
| **pip install numpy** | Numpy is a library for numerical computations, providing supports for arrays, matrices and mathematical operation. |
| **pip install panda** | Pandas is a library for data manipulation and analysis, offering data structure like data frames and series. |
| **pip install sqlalchemy** | Sql toolkit and orm for working with relational database. |
| **pip install pymongo** | Interface for working with mongodb database. |
| **pip install scrapy** | A powerfull web scraping framework |
| **pip install pillow** | For image processing (successor to the pil library). |
| **pip install flask** | A lightweight web framework for building web applications & apis. |
| **pip install fastapi** | A modern web framework for building apis with python, known as its performance. |
| **pip install tornado** | A web server and framework for asynchronous network applications. |
| **pip install opencv-python** | A library for computer vision and image processing. |
| **pip install moviepy** | For video editing and processing |
| **pip install passlib** | For password hashing |
| **pip install pytest** | A framework for testing python application. |
| **pip install boto3** | For interacting with aws services. |
| **pip install pypdf2** | For manipulating pdf files. |
| **pip install faker** | For generating fake data. |

**Importing Modules:**Python provides flexible ways to import & use modules.

**Basic Import:**  import the entire module & access its component using the module name.  
**import math**  
print(math.pi)

**Import with Alias:** Assign an alias to a module for convenience or brevity.  
**import numpy as np**array= np.array([4, 5, 6])

**Import Specific Components:** Import only required function, classes or variables.

**From math import sqrt, pi**

print(sqrt(55))  
print(pi)

Import all Components: Import all components from a module into the global namespace

**From math import\***print(sin(0))

1. Standard library modules: math, random.

**Math:** The math module provides functions for mathematical operations as trigonometry, log and many more.

* **Provides constants like pi and e.**
* Supports advance maths function like trigonometry, log and factorial.

Ex: pi, sqrt, e, factorial, cos, sin, log, etc.

**Random:** The random module is used to generate random numbers and perform random operations.

* Generate randome integer, float and sequences.
* Shuffle lists and choose random items.
* Simulate random events with distributions

Ex: randint, random, choice, shuffle, sample

1. Creating custom modules.

A custom module is a python file created by a developer that contains reusable code such as function, classes and variables. Custom modules help to organize code, improve reusability and make complex programs more maintainable.

**Steps to create custom module**

* **Create a Python File and save as .py extension** (Basic.py)
* **Import the module**

Import my\_module (using custom module)

import basic.py

print(basic.greet(“hello”)

print(basic.add(5, 60)

print(basic.pi)

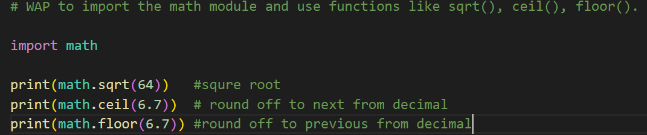
* **Using specific function or variables**

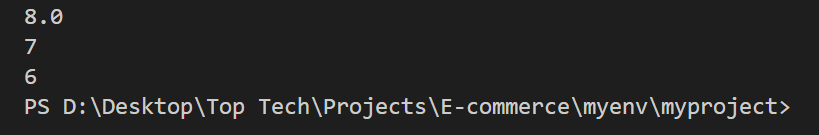
From basic import greet, PI

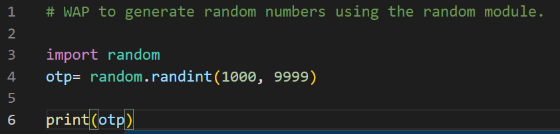
print(basic.greet(“hello”)

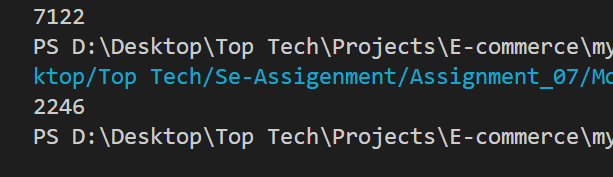
print(basic.pi)

**Lab:**

1. Write a Python program to import the math module and use functions like sqrt(), ceil(), floor().

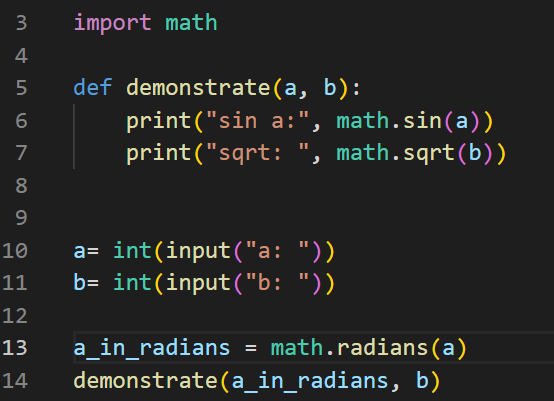
Output:  


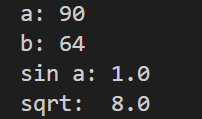
1. Write a Python program to generate random numbers using the random module.

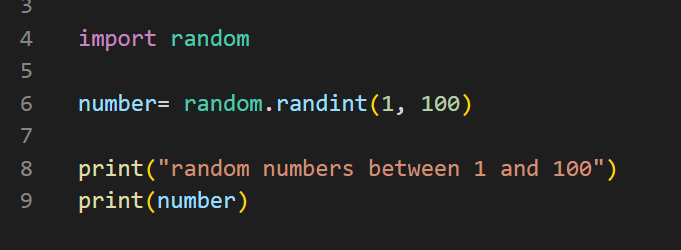
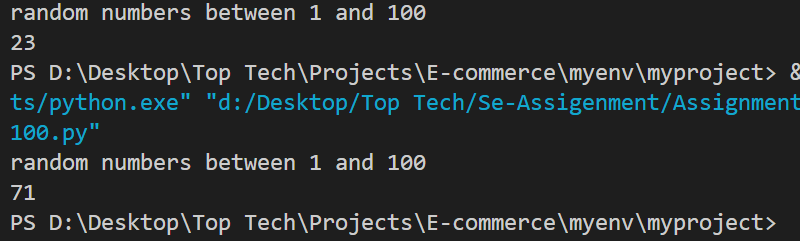
 Output:

**Practical Examples:**

1. Write a Python program to demonstrate the use of functions from the math module.

Output:



1. Write a Python program to generate random numbers between 1 and 100 using the random module.  
   Output: