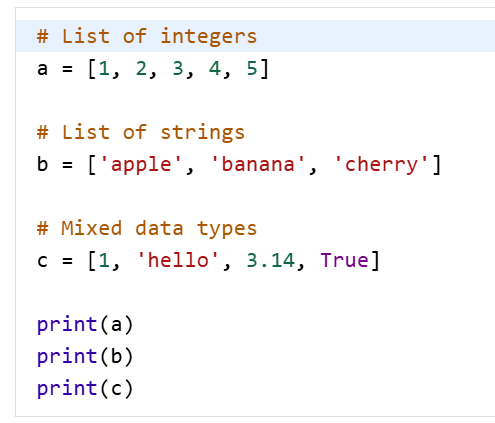
**Introduction to Python Theory (Module – 2)**

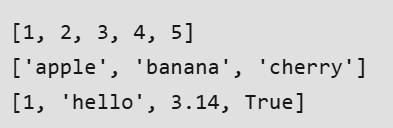
**Q1) Understanding how to create and access elements in a list.**

**ANS)** In Python, a list is a built-in dynamic sized array (automatically grows and shrinks). We can store all types of items (including another list) in a list.

**Here are some common methods to create a list:**

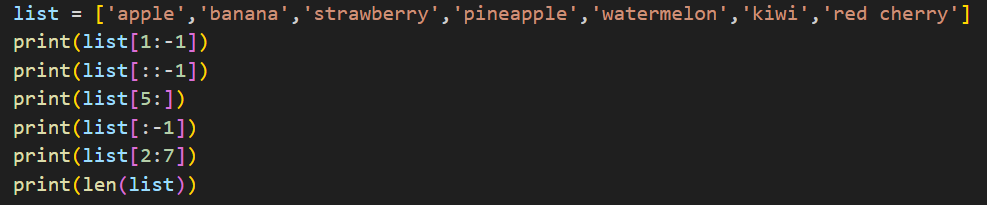
****

**Output:**

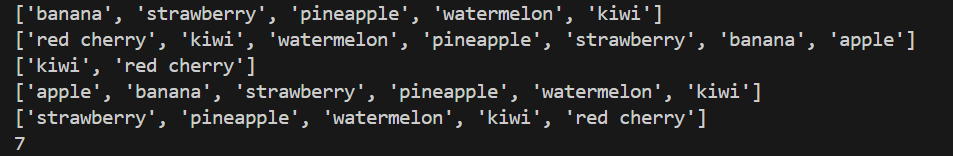
****

**Accessing the list:**

Elements in a list can be accessed using indexing. Python indexes start at 0, so a[0] will access the first element, while negative indexing allows us to access elements from the end of the list. Like index -1 represents the last elements of list.

****

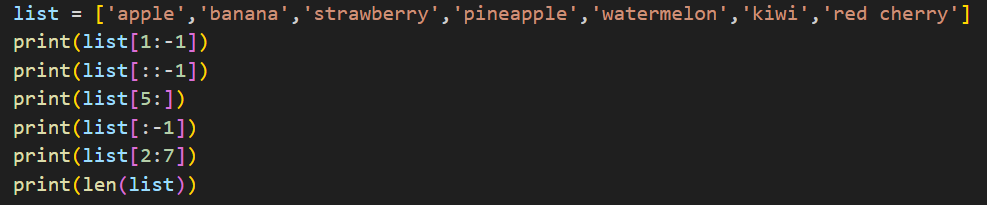
**Output:**

****

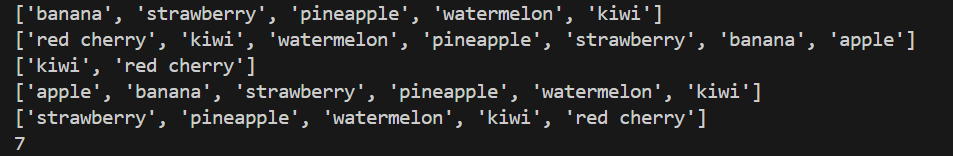
**Q2) Indexing in lists (positive and negative indexing).**

**ANS)** In Python, sequences have both positive and negative indices:

* **Positive Indexing**: Starts from 0 and goes up to n-1 (where n is the length of the sequence).
* **Negative Indexing**: Starts from -1 for the last element and goes up to -n for the first element.

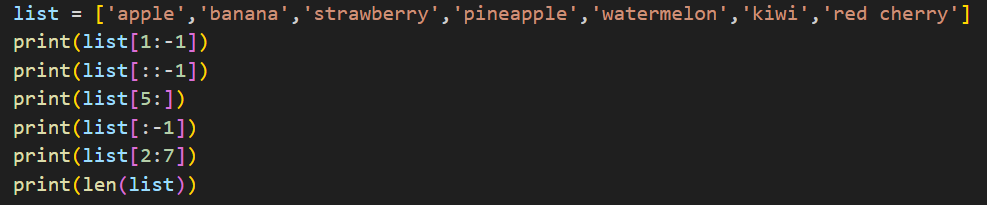
****

**Output:**

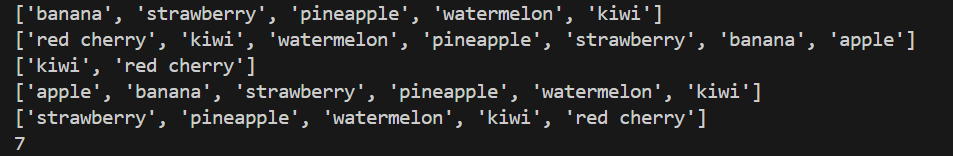
****

**Q3) Slicing a list: accessing a range of elements.**

**ANS)** Python list slicing is fundamental concept that let us easily access specific elements in a list.

****

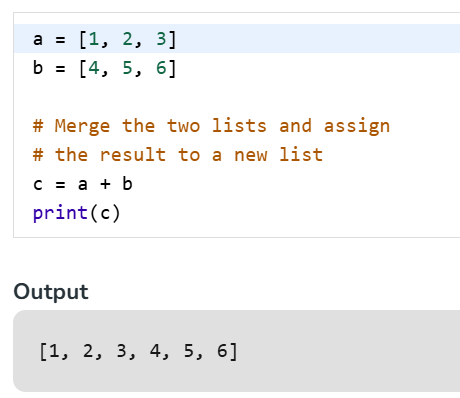
**Output:**

****

**Q4) Common list operations: concatenation, repetition, membership.**

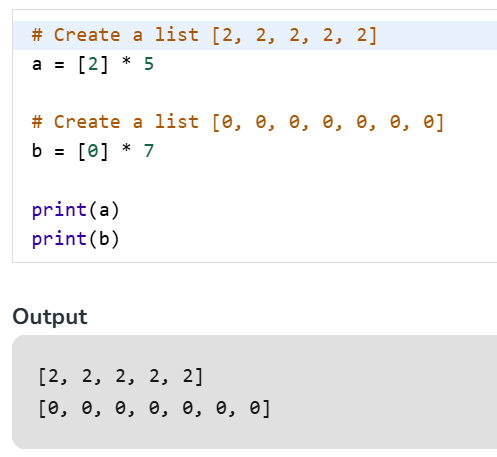
**ANS) Concatenation:**

* Python provides several approaches to merge two lists**.**
* The simplest way to merge two lists is by using the + operator.



**Repetition:**

* We can create a list with repeated elements using the multiplication operator.

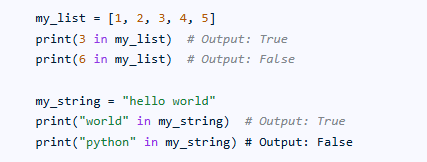


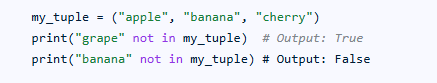
**Membership:**

* Membership operators in Python are used to test whether a value or variable is present within a sequence. These operators return a boolean value (True or False) based on the outcome of the test.

Python has two primary membership operators:

* IN operator: This operator returns True if the specified value is found within the sequence, and False otherwise.





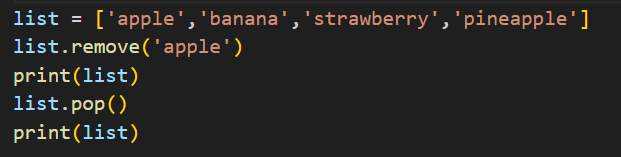
**Q5) Understanding list methods like append(), insert(), remove(), pop().**

**ANS) pop():**

* This method deletes the element at the position mentioned in its argument.

**Remove():**

* This function is used to delete the first occurrence of number mentioned in its arguments.

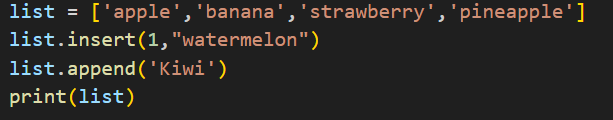
****

**Insert(a,x):**

* This function inserts an element at the position mentioned in its arguments. It takes 2 arguments, position and element to be added at respective position.

**Append():**

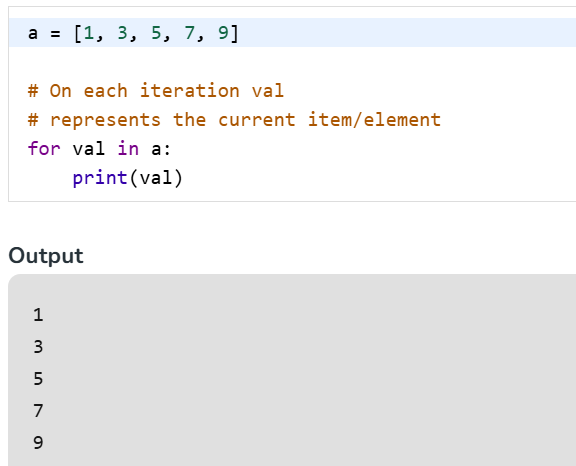
* The list.append() method is a built-in function that adds a single element to the end of an existing list.



**Q6) Iterating over a list using loops.**

**ANS) FOR LOOP:**

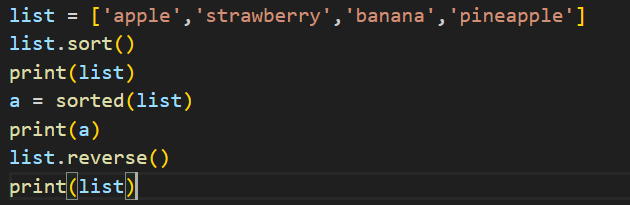
* Python provides several ways to iterate over list. The simplest and the most common way to iterate over a list is to use a for loop.
* This method allows us to access each element in the list directly.



**Q7) Sorting and reversing a list using sort(), sorted(), and reverse().**

**ANS**) The sort() method in Python is a built-in function that allows us to sort the elements of a list in ascending or descending order and it modifies the list in place which means there is no new list created.

* To sort in descending order, use the reverse=True argument.

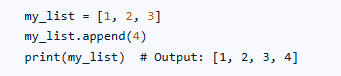


**Q8) Basic list manipulations: addition, deletion, updating, and slicing.**

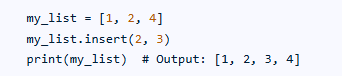
**ANS**) Lists are fundamental data structures in many programming languages, allowing for dynamic collections of items. They support various manipulations, including adding, deleting, updating, and extracting portions (slicing) of the list.

**Adding Elements:**

* append(element): Adds an element to the end of the list.

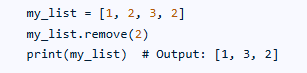


* insert(index, element): Inserts an element at a specific index.

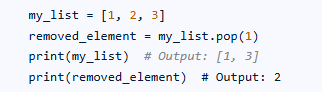


**Deleting Elements:**

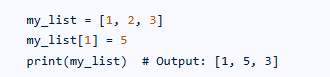
* remove(element): Removes the first occurrence of a specified element.

****

* pop([index]): Removes and returns the element at the specified index (or the last element if no index is provided).

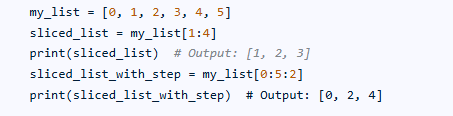
****

**Updating Elements:**

* ****

**Slicing Elements:**

* list[start:end:step]: Creates a new list containing a portion of the original list, starting at start, ending before end, and taking elements with the specified step (default is 1).



**Q9) Introduction to tuples, immutability.**

**ANS)** Python Tuple is a collection of objects separated by commas. A tuple is similar to a Python list in terms of indexing, nested objects, and repetition but the main difference between both is Python tuple is immutable, unlike the Python list which is mutable.

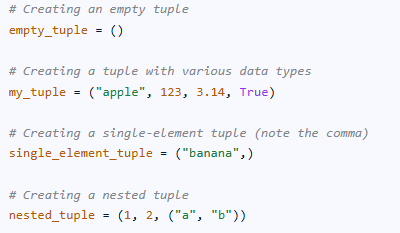
Some Characteristics of Tuples in Python.

1. Like Lists, tuples are ordered and we can access their elements using their index values
2. We cannot update items to a tuple once it is created.
3. Tuples cannot be appended or extended.
4. We cannot remove items from a tuple once it is created.

**Q10) Creating and accessing elements in a tuple**.

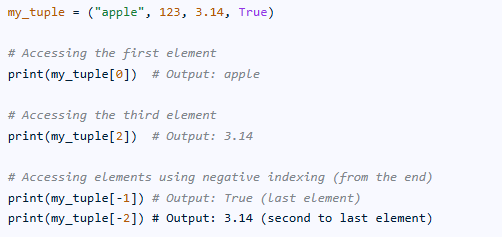
**ANS)** Tuples in Python are ordered, immutable collections of elements. They are defined by enclosing elements within parentheses (), separated by commas.

**Creating a Tuple:**

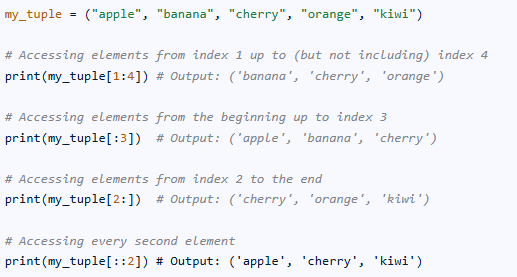


Accessing a tuple:

1. Indexing : Individual elements are accessed using their index within square brackets [].



1. Slicing : A subset of elements can be accessed by specifying a range of indices using slicing [start:end:step]. The end index is exclusive.

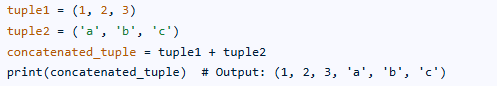


**Q11) Basic operations with tuples: concatenation, repetition, membership.**

**ANS**) Tuples in Python support several operations, including concatenation, repetition, and membership testing.

**Concatenation:**

* Concatenation involves combining two or more tuples to create a new, larger tuple. This operation is performed using the + operator. The elements of the first tuple are followed by the elements of the second tuple in the resulting concatenated tuple.



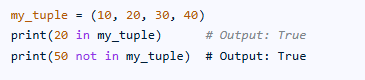
**Repetition:**

* Repetition involves creating a new tuple by repeating the elements of an existing tuple a specified number of times. This operation is performed using the \* operator, where the tuple is multiplied by an integer representing the number of repetitions.



Membership :

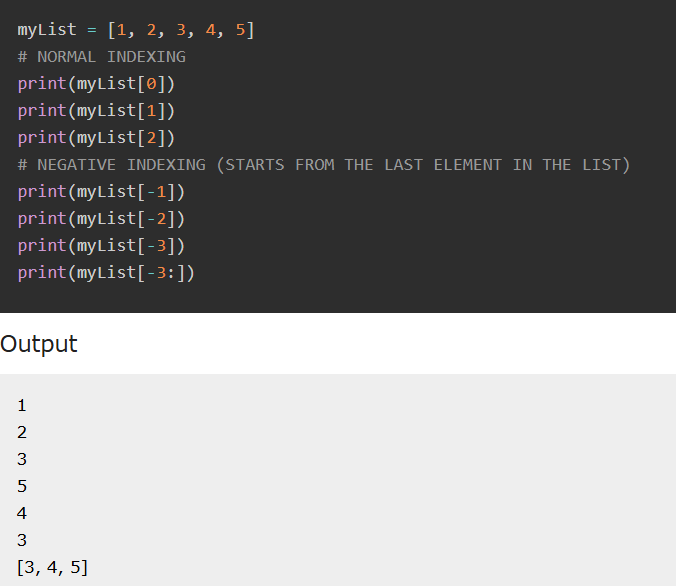
* Membership testing determines whether a specific element exists within a tuple. This is achieved using the in and not in operators. The in operator returns True if the element is found in the tuple and False otherwise. The not in operator returns True if the element is not found and False otherwise.



**Q12) Accessing tuple elements using positive and negative indexing.**

**ANS)** Similar to how we access elements in lists and strings, we can access elements in tuples. So, indexing and slicing are the only methods we need to access items**.**

In Python, positive zero-based indexing is the fundamental method for accessing iterable items.



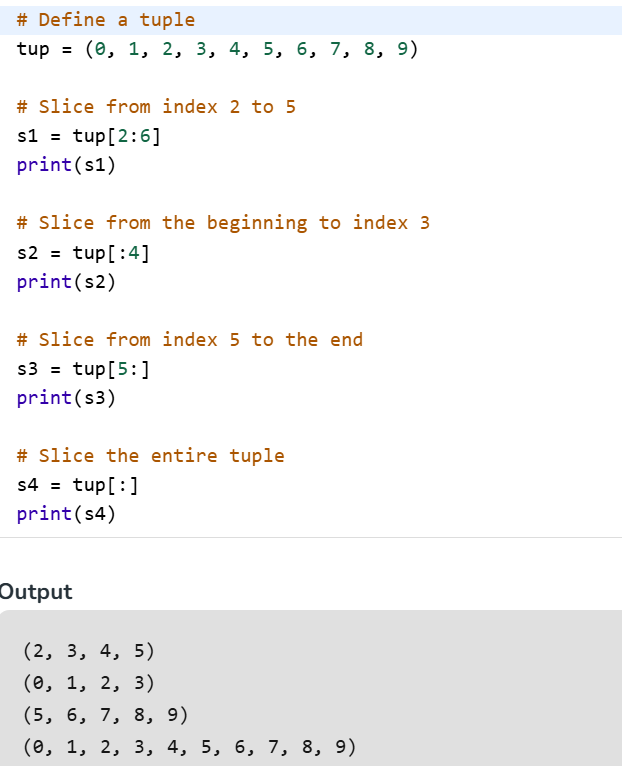
**Q13) Slicing a tuple to access ranges of elements.**

**ANS)** Tuple slicing is a technique to extract a sub-part of a tuple. It uses a range of indices to create a new tuple from the original tuple.

The syntax for slicing is straightforward:

tuple[start:stop:step]

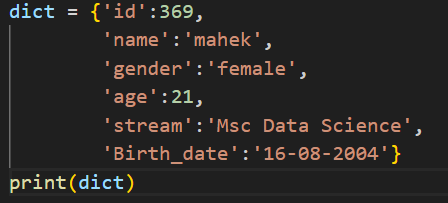
* **start:** The starting index from where the slice begins (inclusive). Default is 0.
* **stop:**The ending index where the slice ends (exclusive).
* **step:**The step size or stride. Default is 1

****

**Q14) Introduction to dictionaries: key-value pairs.**

**ANS)** Python dictionary is a data structure that stores the value in key: value pairs. Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be immutable.

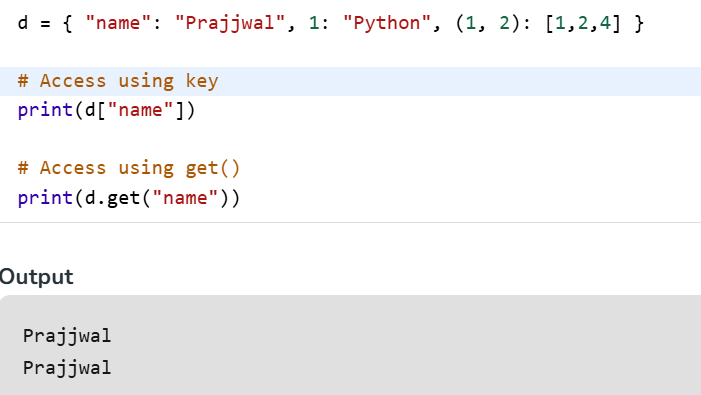
* Dictionary keys are case sensitive: the same name but different cases of Key will be treated distinctly.
* **Keys must be immutable:** This means keys can be strings, numbers or tuples but not lists.
* **Keys must be unique:** Duplicate keys are not allowed and any duplicate key will overwrite the previous value.



**Q15) Accessing, adding, updating, and deleting dictionary elements.**

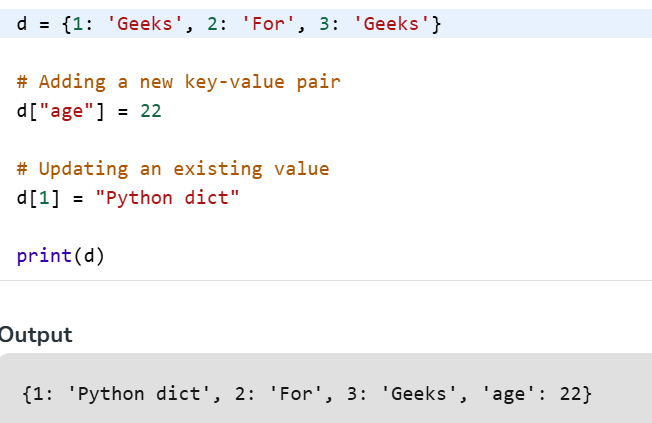
**ANS**) **Accessing Elements:**

* We can access a value from a dictionary by using the key within square brackets or get() method.



**Adding and Updating** **Elements:**

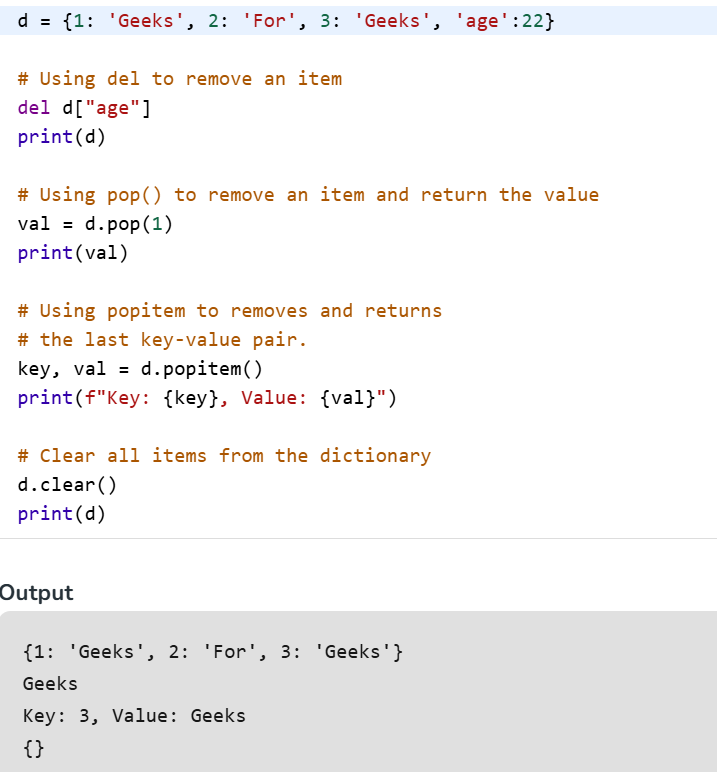
* We can add new key-value pairs or update existing keys by using assignment.



**Deleting Elements:**

**We can remove items from dictionary using the following methods:**

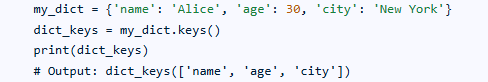
1. [**del**](https://www.geeksforgeeks.org/python/python-del-to-delete-objects/): Removes an item by key.
2. [**pop()**](https://www.geeksforgeeks.org/python/python-dictionary-pop-method/)**:** Removes an item by key and returns its value.
3. [**clear()**](https://www.geeksforgeeks.org/python/python-dictionary-clear/)**:** Empties the dictionary.



**Q16) Dictionary methods like keys(), values(), and items().**

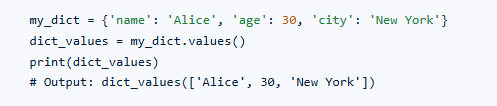
**ANS) Keys():**

* This method returns a view object that displays a list of all the keys in the dictionary. The view object is dynamic, meaning any changes made to the dictionary (like adding or removing a key) will be reflected in the view object.



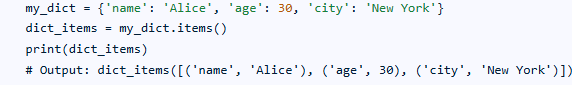
**Values():**

* Similar to keys(), this method returns a view object that displays a list of all the values in the dictionary. This view also reflects dynamic changes to the dictionary's values.



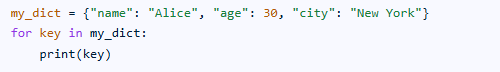
**Items():**

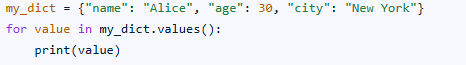
* This method returns a view object that displays a list of all the key-value pairs in the dictionary as tuples. Each tuple contains a key and its corresponding value. Like keys() and values(), this view object is dynamic.

****

**Q17) Iterating over a dictionary using loops.**

**ANS)** Iterating over a dictionary using loops in Python allows access to its keys, values, or both key-value pairs.

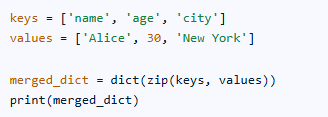




**Q18) Merging two lists into a dictionary using loops or zip().**

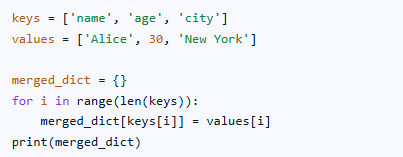
**ANS) Using dict() and zip():**

* This is the most concise and Pythonic way to merge two lists into a dictionary when one list contains the keys and the other contains the corresponding values.



**Using Loop :**

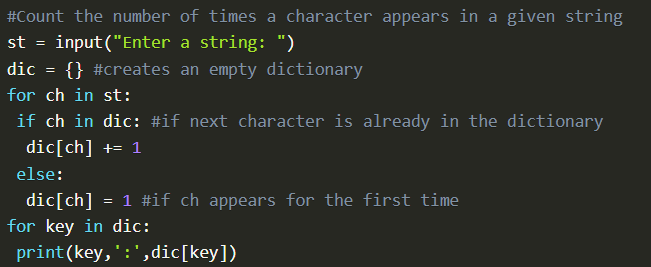
* A traditional for loop can also be used to iterate through the lists and populate a new dictionary.



**Q19) Counting occurrences of characters in a string using dictionaries.**

**ANS) Logic:**

1. Prompt the user to enter a string.
2. Create an empty dictionary called “dic”.
3. Iterate through each character in the given string using a for loop.
4. Check if the character already exists in the “dic” dictionary. If yes, increase its corresponding value by 1.
5. If the character is not yet in the dictionary, add it as a new key with a value of 1.
6. Iterate through each key in the dictionary.
7. Print the key-value pair, separating them with a colon.

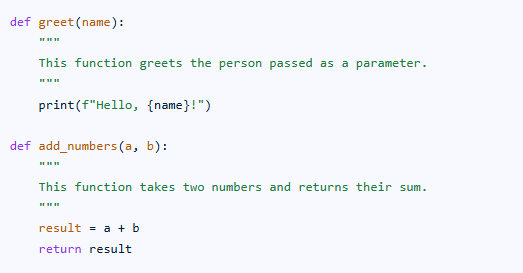


**Q20) Defining functions in Python.**

**ANS)** Defining a function in Python involves using the def keyword. This keyword signals the start of a function definition.

Steps to Define a Function:

* Use the def keyword: Begin the function definition with def, followed by a space.
* Function Name: Choose a descriptive name for your function. Function names should generally be lowercase and use underscores to separate words (e.g., my\_function, calculate\_sum).
* Parentheses and Parameters (Optional): After the function name, include a set of parentheses (). If the function needs to receive data, define parameters within these parentheses, separated by commas (e.g., def greet(name):). These parameters act as placeholders for the arguments that will be passed when the function is called.
* Colon: End the function signature with a colon :.
* Indented Body: The code block that constitutes the function's body must be indented. Indentation in Python is crucial as it defines the scope of the code belonging to the function.
* Return Statement (Optional): If the function needs to produce an output, use the return statement followed by the value or expression to be returned. If no return statement is present, the function implicitly returns None.

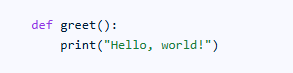


**Q21) Different types of functions: with/without parameters, with/without return values.**

**ANS)** Functions in programming can be categorized based on whether they accept input values (parameters/arguments) and whether they produce an output value (return value). This leads to four main types:

* Functions with no parameters and no return value:

These functions perform a specific task without needing any external input and do not produce a result that needs to be used elsewhere in the program.



* Functions with parameters and no return value:

These functions accept input values, which they use to perform their task, but they do not return a specific value.



* Functions with no parameters and a return value:

These functions do not require any input but calculate and return a specific value.



* Functions with parameters and a return value:

These functions accept input values, use them in their operations, and then return a calculated result.

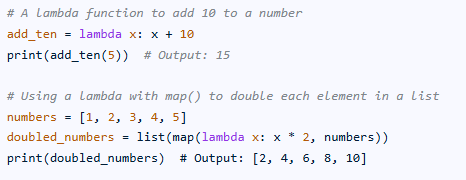


**Q22) Anonymous functions (lambda functions).**

**ANS)** Anonymous functions, also known as lambda functions, are small, single-expression functions in Python that are not bound to a name. They are defined using the lambda keyword and are often used for short, concise operations that don't require a full function definition.

**Characteristics:**

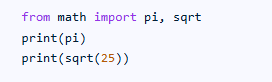
1. No name: They are called anonymous because they don't have a standard function name like those defined with def.
2. Single Expression: They can only contain a single expression, which is evaluated and returned when the function is called.
3. Concise Syntax : The syntax is lambda arguments: expression.
4. Higher – Order Functions : They are frequently used as arguments to higher-order functions like map(), filter(), and sorted().

****

**Q23) Introduction to Python modules and importing modules.**

**ANS)** In Python, a module is a file containing Python code, including definitions of functions, classes, variables, and runnable statements. Modules serve as a fundamental way to organize code into reusable and manageable units, promoting modularity and code reusability across different projects. Any Python file with a .py extension can be considered a module.

Importing modules is the process of making the definitions and functionalities within a module available for use in another Python script or interactive session. This is achieved using the import statement.



**Q24) Standard library modules: math, random.**

**ANS)** The math and random modules are part of Python's standard library. The math module provides mathematical functions and constants, while the random module is used for generating random numbers.

**math Module:**

* Purpose: Offers a wide range of mathematical operations and constants, including trigonometric functions, logarithmic functions, and the value of pi.
* Example: math.sqrt(16) calculates the square root of 16, resulting in 4.0.
* Constants: Includes math.pi (approximately 3.14159) and math.e (Euler's number, approximately 2.71828).

**random Module:**

* Purpose:

Provides functions for generating random numbers, including integers, floats, and selecting random elements from sequences.

* Example:

random.randint(1, 10) generates a random integer between 1 and 10 (inclusive).

* Functions:

Includes random.random() for a random float between 0 and 1, random.choice() for selecting a random element from a sequence, and random.shuffle() for shuffling a sequence in place.

**Q25) Creating custom modules.**

**ANS)** Creating a custom module involves defining a new set of functionalities, data structures, or a distinct namespace within a larger system. This allows for code organization, reusability, and better management of related components. The process typically involves creating a new file, defining the module's contents (functions, classes, variables), and then importing and using it in other parts of the application.

**Importing and Using the Module:**

* Import Statement:

Use the import or from...import statement to make the module's contents accessible in other parts of your application.

* Accessing Contents:

Call functions, instantiate classes, or access variables from the imported module using the module's name or specific members.

* Example (Python):

