## Assignment Part-1

Q1. Why do we call Python as a general purpose and high-level programming language?

Ans:- Python is also considered a high-level programming language because it provides abstractions that allow programmers to focus on the problem at hand rather than low-level implementation details. This makes it easier to write and understand code, and also helps to improve productivity. Additionally, Python has a simple syntax and a large standard library that makes it easier to write complex programs with fewer lines of code.

Q2. Why is Python called a dynamically typed language?

Ans:- Python is called a dynamically typed language because the data types of variables are determined at runtime, rather than at compile-time. This means that a variable can be assigned a value of any data type during the execution of the program.

Q3. List some pros and cons of Python programming language?

Pros:

* Easy to learn and read: Python has a simple syntax and a large standard library that makes it easier to write and understand code, especially for beginners.
* Versatile and powerful: Python can be used for a wide range of applications, from web development to data analysis, scientific computing, machine learning, and more.
* Large and active community: Python has a huge community of developers who contribute to open source libraries and frameworks, which can be used to solve various problems.
* Cross-platform support: Python can run on multiple platforms, including Windows, macOS, Linux, and other operating systems.
* Dynamic typing: Python's dynamic typing feature allows for greater flexibility and ease of use when writing code.

Cons:

* Performance: Python is an interpreted language, which can make it slower than compiled languages like C or C++. However, this gap has been significantly reduced by using libraries like NumPy, Pandas, etc.
* Global Interpreter Lock (GIL): The GIL is a mechanism used by the Python interpreter to synchronize access to Python objects. This can lead to performance issues when running CPU-bound tasks on multi-core machines.
* Memory management: Python's automatic memory management can be a disadvantage when dealing with large amounts of data or when trying to optimize performance.
* Not suitable for mobile development: Python is not the best choice for developing mobile applications due to its performance and memory management issues.
* Less suited for low-level system programming: Python is not well-suited for low-level system programming like device drivers, embedded systems, or operating systems development due to its high-level nature.

Q4. In what all domains can we use Python?

Ans:-

* Web Development: Python is widely used in web development frameworks like Django, Flask, Pyramid, etc. to build web applications and APIs.
* Data Science and Analytics: Python has become the go-to language for data science and analytics, with libraries like NumPy, Pandas, SciPy, and Scikit-learn providing powerful tools for data analysis, machine learning, and scientific computing.
* Machine Learning and Artificial Intelligence: Python has many libraries and frameworks that are used in machine learning and artificial intelligence, such as TensorFlow, Keras, PyTorch, and OpenCV.
* Scientific Computing: Python has libraries like NumPy, SciPy, and Matplotlib that provide powerful tools for scientific computing, including numerical computations, statistical analysis, and data visualization.
* Game Development: Python has game development libraries such as Pygame, PyOpenGL, and PyOgre that can be used to create 2D and 3D games.
* Desktop Application Development: Python can be used to create cross-platform desktop applications using frameworks such as PyQt, wxPython, and PyGTK.
* Network Programming: Python can be used for network programming, including building network applications and automating network tasks using libraries like Paramiko and Netmiko.

Q5. What are variable and how can we declare them?

Ans:- In Python, a variable is a name that refers to a value stored in the computer's memory. Variables are used to store data that can be accessed and manipulated in the program. In Python, variables are created when a value is assigned to them using the assignment operator "=".

Here's an example of declaring and initializing a variable in Python:

my\_var = 10

Q6. How can we take an input from the user in Python?

Ans:- In Python, you can take input from the user using the input() function. The input() function reads a line of text from the user and returns it as a string.

Here's an example of using the input() function to take input from the user:

name = input("What is your name? ")

age = input("What is your age? ")

print("Your name is " + name)

print("Your age is " + age)

Q7. What is the default datatype of the value that has been taken as an input using input() function?

Ans:- In Python, the input() function always returns a string, regardless of what the user enters. This means that the default data type of the value returned by the input() function is a string.

Q8. What is type casting?

Ans:- Type casting refers to the process of converting a value from one data type to another. Python provides several built-in functions for type casting, such as int(), float(), str(), and bool(), among others.

Q9. Can we take more than one input from the user using single input() function? If yes, how? If no, why?

Ans:- Yes, we can take more than one input from the user using a single input() function call in Python. One way to do this is to use the split() method on the input string to split it into a list of strings, and then use tuple unpacking to assign the values to separate variables.

Q10. What are keywords?

Ans:- Keywords in Python are reserved words that have a special meaning and purpose in the language. They are used to define the syntax and structure of Python code and cannot be used as variable names, function names, or any other identifiers.

Q11. Can we use keywords as a variable? Support your answer with reason.

Ans:- No, we cannot use keywords as variable names in Python because they are reserved for special use by the language. Attempting to use a keyword as a variable name will result in a SyntaxError.

For example, the keyword if is used to define conditional statements in Python. If we try to use it as a variable name, we'll get an error:

To avoid this error, we should always use valid and meaningful names for our variables that are not already used by the language.

Q12. What is indentation? What's the use of indentaion in Python?

Ans:- In Python, indentation refers to the whitespace characters (spaces or tabs) at the beginning of a line of code that indicate the level of nesting or block structure in the code. Indentation is used to group statements together in a logical block, such as within a loop, conditional statement, or function definition.The use of indentation in Python is significant because it determines the structure and execution of the code. Unlike other programming languages that use curly braces or keywords to indicate block structure, Python uses indentation to delimit blocks of code. This can help make the code more readable and easier to understand, as the indentation visually shows the hierarchy and flow of the code .

Q13. How can we throw some output in Python?

Ans:- In Python, we can print output to the console using the print() function. The print() function takes one or more arguments as input, separated by commas, and outputs them to the console

Q14. What are operators in Python?

Ans:- Operators in Python are symbols or keywords that perform various operations on one or more operands, such as variables, values, or expressions. There are several types of operators in Python, including: Arithmetic operators, Comparison operators, Logical operators, Assignment operators, Bitwise operators, Identity operators, Membership operators.

Q15. What is difference between / and // operators?

Ans:- The / operator performs floating-point division, which means that it returns a floating-point value even if the operands are integers

The // operator, on the other hand, performs integer division, which means that it returns the integer part of the quotient, discarding any fractional part

Q16. Write a code that gives following as an output.

```

iNeuroniNeuroniNeuroniNeuron

```

Ans:-

string = "iNeuron"

output = string \* 4 + string[1:]

print(output)

Q17. Write a code to take a number as an input from the user and check if the number is odd or even.

Ans:-

num = int(input("Enter a number: "))

if num % 2 == 0:

print(num, "is even")

else:

print(num, "is odd")

Q18. What are boolean operator?

Ans:- In Python, boolean operators are used to perform logical operations on boolean values, which can be either True or False. There are three boolean operators in Python:

and: The and operator returns True if both of its operands are True, and False otherwise. For example, True and False evaluates to False, while True and True evaluates to True.

or: The or operator returns True if at least one of its operands is True, and False otherwise. For example, True or False evaluates to True, while False or False evaluates to False.

not: The not operator returns the opposite of its operand. If the operand is True, then not returns False, and vice versa. For example, not True evaluates to False, while not False evaluates to True.

These operators can be used to combine boolean values and create more complex logical expressions that can be used in control statements and other parts of the code.

Q19. What will the output of the following?

```

1 or 0

0 and 0

True and False and True

1 or 0 or 0

```

Ans:-

1 or 0 evaluates to 1 because 1 is considered as True in boolean context and the or operator returns the first True operand.

0 and 0 evaluates to 0 because 0 is considered as False in boolean context and the and operator returns the first False operand.

True and False and True evaluates to False because the and operator returns the first False operand, and in this case, it's the second operand (False).

1 or 0 or 0 evaluates to 1 because 1 is considered as True in boolean context and the or operator returns the first True operand. In this case, the first operand (1) is True, so the expression short-circuits and returns 1 without evaluating the other operands.

Q20. What are conditional statements in Python?

Ans:- Conditional statements in Python are used to perform different actions based on different conditions. There are two types of conditional statements in Python: if statements and if-else statements.

* if statement: The if statement is used to check whether a condition is true or false, and perform a certain action based on the result.
* if-else statement: The if-else statement is used to perform one action if a condition is true, and a different action if the condition is false

Q21. What is use of 'if', 'elif' and 'else' keywords?

Ans:- The if, elif, and else keywords are used to construct conditional statements in Python

* if statement: The if statement is used to check whether a condition is true or false, and perform a certain action based on the result. If the condition is true, the code inside the if block is executed, otherwise it is skipped.
* elif statement: The elif statement is used to check for an additional condition if the previous condition(s) have not been met. If the if condition is false, the code inside the elif block is executed, provided the condition in the elif statement is true.
* else statement: The else statement is used to specify what to do if none of the previous conditions are met. If all the conditions specified in the if and elif statements are false, the code inside the else block is executed.

Q22. Write a code to take the age of person as an input and if age >= 18 display "I can vote". If age is < 18 display "I can't vote".

Ans:-

age = int(input("Enter your age: "))

if age >= 18:

print("I can vote")

else:

print("I can't vote")

Q23. Write a code that displays the sum of all the even numbers from the given list.

```

numbers = [12, 75, 150, 180, 145, 525, 50]

```

Ans:-

numbers = [12, 75, 150, 180, 145, 525, 50]

even\_sum = 0

for num in numbers:

if num % 2 == 0:

even\_sum += num

print("The sum of all even numbers in the list is:", even\_sum)

Q24. Write a code to take 3 numbers as an input from the user and display the greatest no as output.

Ans:-

num1 = int(input("Enter first number: "))

num2 = int(input("Enter second number: "))

num3 = int(input("Enter third number: "))

if num1 > num2 and num1 > num3:

print("The greatest number is:", num1)

elif num2 > num1 and num2 > num3:

print("The greatest number is:", num2)

else:

print("The greatest number is:", num3)

Q25. Write a program to display only those numbers from a list that satisfy the following conditions

- The number must be divisible by five

- If the number is greater than 150, then skip it and move to the next number

- If the number is greater than 500, then stop the loop

```

numbers = [12, 75, 150, 180, 145, 525, 50]

```

Ans:-

numbers = [12, 75, 150, 180, 145, 525, 50]

for num in numbers:

if num > 500:

break

if num > 150:

continue

if num % 5 == 0:

print(num)

Q26. What is a string? How can we declare string in Python?

Ans:- In Python, a string is a sequence of characters. It is used to represent text and is enclosed in quotation marks (either single or double quotes).

To declare a string in Python, simply assign a sequence of characters to a variable using quotes. For example: my\_string = "Hello, world!"

Q27. How can we access the string using its index?

Ans:- To access a character at a specific index, you can use square brackets [] and specify the index of the character you want to access. For example:

my\_string = "Hello, world!"

print(my\_string[0]) # Output: "H"

print(my\_string[7]) # Output: "w"

Q28. Write a code to get the desired output of the following

string = "Big Data iNeuron"

desired\_output = "iNeuron"

Ans:-

string = "Big Data iNeuron"

desired\_output = string[9:]

print(desired\_output)

Q29. Write a code to get the desired output of the following

string = "Big Data iNeuron"

desired\_output = "norueNi"

Ans:-

string = "Big Data iNeuron"

desired\_output = string[12:6:-1]

print(desired\_output)

Q30. Resverse the string given in the above question.

Ans:-

string = "Big Data iNeuron"

reversed\_string = string[::-1]

print(reversed\_string)

Q31. How can you delete entire string at once?

Ans:-

my\_string = "Hello, world!"

del my\_string

Q32. What is escape sequence?

Ans:- In Python, an escape sequence is a combination of characters that represents a special character or sequence of characters in a string. An escape sequence begins with a backslash \ character, which is followed by one or more special characters.

Escape sequences are used to represent characters that cannot be entered directly in a string

Q33. How can you print the below string?

'iNeuron's Big Data Course'

Ans:- print("iNeuron's Big Data Course")

Q34. What is a list in Python?

Ans:- In Python, a list is a collection of items that are ordered and changeable. Lists are one of the built-in data types in Python, and they are commonly used to store collections of related data.

Q35. How can you create a list in Python?

Ans:- my\_list = [1, 2, 3, 4, 5]

Q36. How can we access the elements in a list?

Ans:- You can access the elements in a list by their index. The index of the first element in a list is 0, and the index of the last element is len(my\_list) - 1, where my\_list is the name of the list.

To access an element of a list, you can use the square bracket [] notation and provide the index of the element you want to access

Q37. Write a code to access the word "iNeuron" from the given list.

lst = [1,2,3,"Hi",[45,54, "iNeuron"], "Big Data"]

Ans:-

lst = [1, 2, 3, "Hi", [45, 54, "iNeuron"], "Big Data"]

word = lst[4][2]

print(word)

Q38. Take a list as an input from the user and find the length of the list.

Ans:-

user\_input = input("Enter a list of values, separated by spaces: ")

my\_list = user\_input.split()

print("Length of the list:", len(my\_list))

Q39. Add the word "Big" in the 3rd index of the given list.

lst = ["Welcome", "to", "Data", "course"]

Ans:-

lst = ["Welcome", "to", "Data", "course"]

lst.insert(3, "Big")

print(lst)

Q40. What is a tuple? How is it different from list?

Ans:- In Python, a tuple is a collection of ordered, immutable elements, similar to a list. The main difference between a tuple and a list is that a tuple is immutable, meaning that once it is created, its contents cannot be changed.

Here are some other key differences between tuples and lists:

* Syntax: A tuple is created using parentheses (), while a list is created using square brackets [].
* Mutability: Tuples are immutable, meaning that once a tuple is created, its contents cannot be changed. Lists are mutable, meaning that their contents can be changed after creation.
* Performance: Tuples are generally faster than lists for certain operations, such as indexing and iteration, because they are simpler data structures.
* Use cases: Tuples are often used to represent fixed collections of related data, such as coordinates or RGB color values, whereas lists are often used to represent collections of data that can change over time.

Q41. How can you create a tuple in Python?

Ans:- In Python, you can create a tuple by enclosing a sequence of elements in parentheses () and separating them with commas. Here's an example: my\_tuple = (1, 2, 3, "hello", 4.5)

Q42. Create a tuple and try to add your name in the tuple. Are you able to do it? Support your answer with reason.

Ans:- No, it's not possible to add a new element to a tuple in Python because tuples are immutable, meaning that their contents cannot be changed after creation.

Q43. Can two tuple be appended. If yes, write a code for it. If not, why?

Ans:- No, it's not possible to append two tuples in Python. This is because tuples are immutable, meaning that their contents cannot be changed after creation. Therefore, there is no way to add elements to an existing tuple.

However, you can create a new tuple by concatenating two existing tuples using the + operator

Q44. Take a tuple as an input and print the count of elements in it.

Ans:-

my\_tuple = tuple(input("Enter a tuple of elements separated by commas: ").split(","))

print("The tuple you entered is:", my\_tuple)

print("The number of elements in the tuple is:", len(my\_tuple))

Q45. What are sets in Python?

Ans:- In Python, a set is an unordered collection of unique elements. A set can be created by enclosing a comma-separated list of values in curly braces {}, or by calling the built-in set() function with an iterable object as an argument.

Q46. How can you create a set?

Ans:- You can create a set in Python by enclosing a comma-separated list of values in curly braces {} or by calling the built-in set() function with an iterable object as an argument.

Q47. Create a set and add "iNeuron" in your set.

Ans:-

my\_set = {1, 2, 3, 4}

print("Original set:", my\_set)

my\_set.add("iNeuron")

print("Set after adding 'iNeuron':", my\_set)

Q48. Try to add multiple values using add() function.

Ans:-

my\_set = {1, 2, 3, 4}

print("Original set:", my\_set)

my\_set.update([5, 6, 7])

print("Set after adding multiple values:", my\_set)

Q49. How is update() different from add()?

Ans:- In Python, the add() and update() functions are used to add elements to a set, but they have different behaviors:

* The add() function is used to add a single element to a set. If the element is already in the set, the set remains unchanged.
* The update() function is used to add multiple elements to a set. It takes an iterable object (like a list, tuple, or another set) as an argument, and adds each element of the iterable to the set. If any of the elements in the iterable are already in the set, they are ignored

Q50. What is clear() in sets?

Ans:- In Python, the clear() method is a built-in function of a set, which is used to remove all the elements from a set. After the clear() method is applied to a set, the set becomes empty

Q51. What is frozen set?

Ans:- In Python, a frozen set is an immutable set, which means its elements cannot be changed once it is created. A frozen set is a type of set that is created by calling the built-in frozenset() function.

The elements of a frozen set must be immutable objects like numbers, strings, or tuples that contain only immutable objects. Like sets, frozen sets also support set operations such as union, intersection, and difference.

Q52. How is frozen set different from set?

Ans:- The main difference between a frozen set and a regular set in Python is that a frozen set is immutable and cannot be modified after it is created, whereas a regular set is mutable and can be changed by adding, removing, or updating its elements.

Once a frozen set is created, its elements cannot be added, removed, or modified. This means that methods like add(), remove(), and update() are not available for frozen sets, as they would modify the set. However, frozen sets still support set operations like union, intersection, and difference, just like regular sets.

Another difference is that frozen sets use less memory than regular sets, but they are less flexible because they cannot be modified.

Q53. What is union() in sets? Explain via code.

Ans:- In Python, the union() method is used to return a new set containing all the distinct elements from two or more sets. The union() method can be called on a set object and takes one or more iterable objects (such as sets or lists) as arguments.

Here's an example code to illustrate the union() method in sets:

# Creating two sets

set1 = {1, 2, 3, 4, 5}

set2 = {4, 5, 6, 7, 8}

# Performing union of the two sets using the union() method

set3 = set1.union(set2)

# Printing the original sets and the union set

print("Set 1:", set1)

print("Set 2:", set2)

print("Union Set:", set3)

Q54. What is intersection() in sets? Explain via code.

Ans:- In Python, the intersection() method is used to return a new set containing the common elements between two or more sets. The intersection() method can be called on a set object and takes one or more iterable objects (such as sets or lists) as arguments.

Here's an example code to illustrate the intersection() method in sets:

# Creating two sets

set1 = {1, 2, 3, 4, 5}

set2 = {4, 5, 6, 7, 8}

# Performing intersection of the two sets using the intersection() method

set3 = set1.intersection(set2)

# Printing the original sets and the intersection set

print("Set 1:", set1)

print("Set 2:", set2)

print("Intersection Set:", set3)

Q55. What is dictionary ibn Python?

Ans:- A dictionary is a collection of key-value pairs in Python. It is a built-in data type that allows you to store data in an unordered manner. Each key in a dictionary is unique and points to a corresponding value. The key-value pairs are enclosed in curly braces ({}) and separated by a colon (:), and the individual pairs are separated by a comma.

Dictionaries are mutable, which means that you can add, remove or modify items after the dictionary has been created. They are commonly used to store and manipulate data that can be indexed by keys rather than by an ordered sequence of elements.

Q56. How is dictionary different from all other data structures.

Ans:- Dictionaries are different from all other data structures in Python because they store data as key-value pairs. Unlike lists and tuples, which store data in an ordered sequence, dictionaries store data in an unordered manner. The keys in a dictionary must be unique and immutable, while the values can be of any data type.

Sets are another data structure that store data in an unordered manner, but sets only contain unique values without any associated keys. Lists and tuples are ordered sequences of elements that can be accessed using an index, while dictionaries are accessed using the keys.

One of the main advantages of dictionaries is their ability to provide fast access to data based on a key. Dictionaries are implemented using a hash table, which allows for constant-time access to the values associated with a given key.

Overall, dictionaries are a powerful and flexible data structure that are commonly used in Python for storing and accessing data based on key-value pairs.

Q57. How can we delare a dictionary in Python?

Ans:-

# empty dictionary

my\_dict = {}

# dictionary with key-value pairs

my\_dict = {"key1": "value1", "key2": "value2"}

Q58. What will the output of the following?

var = {}

print(type(var))

Ans:- <class 'dict'>

This is because curly braces {} are used to declare both sets and dictionaries in Python, but an empty set is declared using set(), while an empty dictionary is declared using {}. In this case, an empty dictionary is being declared and assigned to the variable var. When the type() function is called on var, it returns <class 'dict'>, indicating that var is of the dictionary data type.

Q59. How can we add an element in a dictionary?

Ans:- You can add an element (key-value pair) to a dictionary in Python by assigning a value to a new or existing key using the square bracket notation or the update() method. Here are examples of both ways to add a new key-value pair to a dictionary:

Using square bracket notation:

my\_dict = {"key1": "value1", "key2": "value2"}

# add a new key-value pair

my\_dict["key3"] = "value3"

# update an existing key-value pair

my\_dict["key1"] = "new\_value1"

Q60. Create a dictionary and access all the values in that dictionary.

Ans:-

# create a dictionary with key-value pairs

my\_dict = {'apple': 3, 'banana': 5, 'orange': 2}

# access all the values in the dictionary using values() method

all\_values = my\_dict.values()

# print all the values

print(all\_values)

Q61. Create a nested dictionary and access all the element in the inner dictionary.

Ans:-

# create a nested dictionary

my\_dict = {'apple': {'price': 2, 'color': 'red'},

'banana': {'price': 1, 'color': 'yellow'},

'orange': {'price': 3, 'color': 'orange'}}

# access all the elements in the inner dictionary

for key in my\_dict:

print(f"Price of {key} is {my\_dict[key]['price']} and color is {my\_dict[key]['color']}")

Q62. What is the use of get() function?

Ans:- The benefit of using get() over directly accessing the dictionary with the square bracket notation ([]) is that if the key is not found, get() will return None (or the default value if provided), while using square brackets notation directly will result in a KeyError if the key is not found. This can be particularly useful in cases where it is not certain whether the key is present in the dictionary.

Q63. What is the use of items() function?

Ans:- The items() function in Python is used to return a list of dictionary's (key, value) tuple pairs. It returns a view object that contains the key-value pairs of the dictionary, i.e., items in the dictionary.

Q64. What is the use of pop() function?

Ans:- The pop() function is useful when you want to remove an item from the dictionary and also want to get its value for further use. If you try to access a key that does not exist in the dictionary, it will raise a KeyError exception. In this case, you can pass a default value as the second argument to pop() to avoid the KeyError. If the key does not exist in the dictionary, the default value will be returned.

Q65. What is the use of popitems() function?

Ans:- The popitems() function is used to remove and return an arbitrary key-value pair from a dictionary. This method is useful when you need to retrieve and remove items from a dictionary in an arbitrary order, such as when you are implementing a queue or a stack. The popitems() method removes the last inserted item in Python 3.7 or later, and a random item in earlier versions.

Q66. What is the use of keys() function?

Ans:- The keys() function is a built-in method in Python that returns a view object that contains the keys of a dictionary. It can be useful when you need to access only the keys of a dictionary without the corresponding values. You can use the keys() method to iterate over the keys of a dictionary, or to check if a particular key is present in the dictionary. The keys() method returns a dynamic view object, which means that any changes made to the dictionary will be reflected in the view object, and vice versa.

Q67. What is the use of values() function?

Ans:- The values() function is a built-in method in Python that returns a view object that contains the values of a dictionary. It can be useful when you need to access only the values of a dictionary without the corresponding keys. You can use the values() method to iterate over the values of a dictionary, or to check if a particular value is present in the dictionary. The values() method returns a dynamic view object, which means that any changes made to the dictionary will be reflected in the view object, and vice versa.

Q68. What are loops in Python?

Ans:- Loops in Python are a way to execute a block of code repeatedly until a certain condition is met. There are two types of loops in Python: for loops and while loops.

Q69. How many type of loop are there in Python?

Ans:- A for loop is used to iterate over a sequence, such as a list or a string, and execute the code inside the loop for each item in the sequence

A while loop is used to repeat a block of code as long as a certain condition is true.

Q70. What is the difference between for and while loops?

Ans:- The main difference between for and while loops in Python is how they control the flow of execution of the loop.

* A for loop is used to iterate over a sequence, such as a list or a string, and execute the code inside the loop for each item in the sequence. The number of iterations is determined by the length of the sequence, and the loop variable takes on the value of each item in the sequence. The for loop is typically used when you know the number of iterations you want to perform in advance.
* A while loop is used to repeat a block of code as long as a certain condition is true. The condition is checked at the beginning of each iteration, and the loop continues until the condition becomes false. The while loop is typically used when you don't know in advance how many times you need to iterate, or when you want to iterate until a certain condition is met.

In general, for loops are more appropriate for iterating over a sequence, while while loops are more appropriate for situations where you need to perform a certain action until a specific condition is met. However, the choice between the two depends on the specific requirements of your program.

Q71. What is the use of continue statement?

Ans:- The continue statement is used in Python to skip the remaining code in the current iteration of a loop (either for or while), and continue with the next iteration of the loop.

Q72. What is the use of break statement?

Ans:- The break statement is used in Python to terminate the execution of a loop (either for or while) prematurely. When the break statement is encountered in a loop, it causes the loop to immediately terminate, and control is passed to the next statement after the loop.

Q73. What is the use of pass statement?

Ans:- The pass statement is a null statement in Python, meaning that it doesn't do anything. It is often used as a placeholder in code, where you need to include a statement for syntactical reasons, but don't want to actually do anything.

Q74. What is the use of range() function?

Ans:- The range() function in Python is used to generate a sequence of numbers, typically used for iterating over a sequence using a for loop.

Q75. How can you loop over a dictionary?

Ans:- You can loop over a dictionary in Python using a for loop. The for loop will iterate over the keys of the dictionary, and you can access the corresponding values using the keys.

**Coding problems**

Q76. Write a Python program to find the factorial of a given number.

num = int(input("Enter a number: "))

factorial = 1

if num < 0:

print("Factorial is not defined for negative numbers.")

elif num == 0:

print("Factorial of 0 is 1.")

else:

for i in range(1, num + 1):

factorial = factorial \* i

print(f"Factorial of {num} is {factorial}.")

Q77. Write a Python program to calculate the simple interest. Formula to calculate simple interest is SI = (P*R*T)/100

p = float(input("Enter the principal amount: "))

r = float(input("Enter the rate of interest: "))

t = float(input("Enter the time duration in years: "))

si = (p \* r \* t) / 100

print(f"The simple interest for principal amount {p}, rate of interest {r}% and time {t} years is {si}.")

Q78. Write a Python program to calculate the compound interest. Formula of compound interest is A = P(1+ R/100)^t.

p = float(input("Enter the principal amount: "))

r = float(input("Enter the rate of interest: "))

t = float(input("Enter the time duration in years: "))

# Calculate the compound interest

a = p \* (1 + (r/100)) \*\* t

# Calculate the total amount

ci = a - p

print(f"The compound interest for principal amount {p}, rate of interest {r}% and time {t} years is {ci}.")

Q79. Write a Python program to check if a number is prime or not.

num = int(input("Enter a number: "))

# Prime number is greater than 1

if num > 1:

# Check for factors

for i in range(2, int(num/2)+1):

if (num % i) == 0:

print(f"{num} is not a prime number.")

break

else:

print(f"{num} is a prime number.")

else:

print(f"{num} is not a prime number.")

Q80. Write a Python program to check Armstrong Number.

num = int(input("Enter a number: "))

# determine the number of digits

num\_digits = len(str(num))

# initialize the sum

sum = 0

# find the sum of the cubes of each digit

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* num\_digits

temp //= 10

# check if the number is an Armstrong number

if num == sum:

print(f"{num} is an Armstrong number.")

else:

print(f"{num} is not an Armstrong number.")

Q81. Write a Python program to find the n-th Fibonacci Number.

n = int(input("Enter the value of n: "))

# Check if the input is valid

if n <= 0:

print("Invalid input. Please enter a positive integer.")

else:

# Initialize the first two numbers

num1 = 0

num2 = 1

# Check if n is 1 or 2

if n == 1:

print("The 1st Fibonacci number is 0.")

elif n == 2:

print("The 2nd Fibonacci number is 1.")

else:

# Find the n-th Fibonacci number

for i in range(2, n):

fib\_num = num1 + num2

num1 = num2

num2 = fib\_num

print(f"The {n}th Fibonacci number is {num2}.")

Q82. Write a Python program to interchange the first and last element in a list.

lst = input("Enter a list of elements separated by space: ").split()

# Convert each element to an integer

lst = [int(i) for i in lst]

# Check if the list has at least two elements

if len(lst) < 2:

print("List must have at least two elements.")

else:

# Interchange the first and last element

lst[0], lst[-1] = lst[-1], lst[0]

print("List after interchanging the first and last element:")

print(lst)

Q83. Write a Python program to swap two elements in a list.

lst = input("Enter a list of elements separated by space: ").split()

# Convert each element to an integer

lst = [int(i) for i in lst]

# Check if the list has at least two elements

if len(lst) < 2:

print("List must have at least two elements.")

else:

# Get the indices of the elements to be swapped

i1 = int(input("Enter the index of the first element to be swapped: "))

i2 = int(input("Enter the index of the second element to be swapped: "))

# Check if the indices are valid

if i1 < 0 or i1 >= len(lst) or i2 < 0 or i2 >= len(lst):

print("Invalid indices.")

else:

# Swap the elements

lst[i1], lst[i2] = lst[i2], lst[i1]

print("List after swapping elements:")

print(lst)

Q84. Write a Python program to find N largest element from a list.

lst = input("Enter a list of elements separated by space: ").split()

# Convert each element to an integer

lst = [int(i) for i in lst]

n = int(input("Enter the value of N: "))

# Use the sorted function to sort the list in descending order

sorted\_lst = sorted(lst, reverse=True)

print(f"The {n} largest elements in the list are:")

print(sorted\_lst[:n])

Q85. Write a Python program to find cumulative sum of a list.

lst = input("Enter a list of elements separated by space: ").split()

# Convert each element to an integer

lst = [int(i) for i in lst]

cumulative\_sum = []

# Calculate the cumulative sum of the elements in the list

for i in range(len(lst)):

if i == 0:

cumulative\_sum.append(lst[i])

else:

cumulative\_sum.append(cumulative\_sum[-1] + lst[i])

print("Cumulative sum of the list:")

print(cumulative\_sum)

Q86. Write a Python program to check if a string is palindrome or not.

string = input("Enter a string: ")

# Convert the string to lowercase and remove all non-alphanumeric characters

string = ''.join(e for e in string if e.isalnum()).lower()

# Check if the string is equal to its reverse

if string == string[::-1]:

print("The string is a palindrome.")

else:

print("The string is not a palindrome.")

Q87. Write a Python program to remove i'th element from a string.

Strings in Python are immutable, which means that you cannot remove an element from a string. However, you can remove a character from a string and create a new string with the remaining characters.

string = input("Enter a string: ")

i = int(input("Enter the index of the character to remove: "))

# Check if the index is valid

if i < 0 or i >= len(string):

print("Invalid index.")

else:

# Remove the character at the specified index

new\_string = string[:i] + string[i+1:]

print("String after removing the character:")

print(new\_string)

Q88. Write a Python program to check if a substring is present in a given string.

string = input("Enter a string: ")

substring = input("Enter a substring: ")

# Check if the substring is present in the string

if substring in string:

print("The substring is present in the string.")

else:

print("The substring is not present in the string.")

Q89. Write a Python program to find words which are greater than given length k.

string = input("Enter a string: ")

k = int(input("Enter the value of k: "))

# Split the string into a list of words

words = string.split()

# Find the words that are greater than length k

long\_words = [word for word in words if len(word) > k]

print(f"The words greater than length {k} are:")

print(long\_words)

Q90. Write a Python program to extract unquire dictionary values.

my\_dict = {'a': 1, 'b': 2, 'c': 1, 'd': 3, 'e': 2}

# Get a list of all the values in the dictionary

values = list(my\_dict.values())

# Get a set of the unique values in the dictionary

unique\_values = set(values)

print("Unique values in the dictionary are:")

print(unique\_values)

Q91. Write a Python program to merge two dictionary.

dict1 = {'a': 1, 'b': 2, 'c': 3}

dict2 = {'d': 4, 'e': 5, 'f': 6}

# Merge the two dictionaries using the update() method

merged\_dict = dict1.copy()

merged\_dict.update(dict2)

print("Merged dictionary:")

print(merged\_dict)

Q92. Write a Python program to convert a list of tuples into dictionary.

Input : [('Sachin', 10), ('MSD', 7), ('Kohli', 18), ('Rohit', 45)]

Output : {'Sachin': 10, 'MSD': 7, 'Kohli': 18, 'Rohit': 45}

my\_list = [('Sachin', 10), ('MSD', 7), ('Kohli', 18), ('Rohit', 45)]

# Convert the list of tuples into a dictionary

my\_dict = dict(my\_list)

print("Dictionary:")

print(my\_dict)

Q93. Write a Python program to create a list of tuples from given list having number and its cube in each tuple.

Input: list = [9, 5, 6]

Output: [(9, 729), (5, 125), (6, 216)]

my\_list = [9, 5, 6]

# Create a list of tuples, where each tuple contains a number and its cube

my\_tuples = [(num, num\*\*3) for num in my\_list]

print("List of tuples:")

print(my\_tuples)

Q94. Write a Python program to get all combinations of 2 tuples.

Input : test\_tuple1 = (7, 2), test\_tuple2 = (7, 8)

Output : [(7, 7), (7, 8), (2, 7), (2, 8), (7, 7), (7, 2), (8, 7), (8, 2)]

from itertools import product

test\_tuple1 = (7, 2)

test\_tuple2 = (7, 8)

# Get all combinations of the two tuples using the product function

combinations = list(product(test\_tuple1, test\_tuple2)) + list(product(test\_tuple2, test\_tuple1))

print("Combinations:")

print(combinations)

Q95. Write a Python program to sort a list of tuples by second item.

Input : [('for', 24), ('Geeks', 8), ('Geeks', 30)]

Output : [('Geeks', 8), ('for', 24), ('Geeks', 30)]

my\_list = [('for', 24), ('Geeks', 8), ('Geeks', 30)]

# Sort the list of tuples by the second item using a lambda function

sorted\_list = sorted(my\_list, key=lambda x: x[1])

print("Sorted list:")

print(sorted\_list)

Q96. Write a python program to print below pattern.

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

n = 5

# Loop through the rows

for i in range(n):

# Loop through the columns

for j in range(i+1):

print("\*", end=" ")

# Move to the next line

print()

Q97. Write a python program to print below pattern.

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

n = 5

# Loop through the rows

for i in range(n):

# Print spaces for the left side of the triangle

for j in range(n-i-1):

print(" ", end="")

# Print stars for the right side of the triangle

for j in range(i+1):

print("\*", end="")

# Move to the next line

print()

Q98. Write a python program to print below pattern.

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

n = 5

# Loop through the rows

for i in range(n):

# Print spaces for the left side of the triangle

for j in range(n-i-1):

print(" ", end="")

# Print stars for the right side of the triangle

for j in range(i+1):

print("\* ", end="")

# Move to the next line

print()

Q99. Write a python program to print below pattern.

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

n = 5

# Loop through the rows

for i in range(n):

# Loop through the columns

for j in range(i+1):

# Print the number for the current column

print(j+1, end=" ")

# Move to the next line

print()

Q100. Write a python program to print below pattern.

A

B B

C C C

D D D D

E E E E E

Give feedback

n = 5

# ASCII value of 'A' is 65

ascii\_value = 65

# Loop through the rows

for i in range(n):

# Loop through the columns

for j in range(i+1):

# Convert the ASCII value to a character and print it for the current column

print(chr(ascii\_value), end=" ")

# Increment the ASCII value for the next row

ascii\_value += 1

# Move to the next line

print()