# Python mega assignment

# Assignment Part-1

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

**ANSWER:** Python is a high-level programming language because it is designed to allow humans to write code that will interact with a computer system without having any knowledge of processor and hardware.

Q2. Why is Python called a dynamically typed language?

**ANSWER:** Python is a dynamically typed language because of the type of the variable is determined only during runtime. We don’t need to specially mentioned the type of the variable in python.

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

**ANSWER:**

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Beginner-friendly | Slower than compiled languages |
| Extensive Libraries | High memory consumption |
| Large Community | Complex multithreading |
| Highly Scalable | Garbage collection leads to potential memory losses |
| Flexible and Extensible | Dynamically-typed language |

Q4. In what all domains can we use Python?

**ANSWER:** We can use Python in web development, data science, OS development, Scientific programming, gaming and etc

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

**ANSWER:** Variables are the names we give to computer memory locations which are used to store values in a computer program. We can declare them using following syntax variablename = variablevalue. Example: varstring = “This is Python variable declaration”

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

**ANSWER:** Using the input() function we can take an input from the user in Python.

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

**ANSWER:** Python takes all the input as a string input by default.

Q8. What is type casting?

**ANSWER:** Type casting is when we assign a value of one primitive data type to another type.

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

**ANSWER:** In Python user can take multiple values or inputs in one line by two methods. Using split() method Using List comprehension.

* Using split() method: This function helps in getting multiple inputs from user. It breaks the given input by the specified separator. If a separator is not provided then any white space is a separator. Generally, user use a split() method to split a Python string but one can use it in taking multiple input. Syntax: input().split(separator, maxsplit). Example: x, y, z = input("Enter a three value: ").split()
* Using List comprehension: List comprehension is an elegant way to define and create list in Python. We can create lists just like mathematical statements in one line only. It is also used in getting multiple inputs from a user. Example: x, y = [int(x) for x in input("Enter two values: ").split()]

Q10. What are keywords?

**ANSWER:** Keywords are predefined, reserved words used in programming that have special meanings to the compiler/interpreter.

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

**ANSWER:** No, we can’t use keywords as a variable. Keywords are reserved words that have a special meaning to the compiler/interpreter. Compiler/Interpreter reserves these words for its own use so they are not available as names for variables or methods.

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

**ANSWER:** Indentation refers to the spaces at the beginning of a code line. Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important. Python uses indentation to indicate a block of code.

Q13. How can we throw some output in Python?

**ANSWER:** Using the output() function we can throw output in Python.

Q14. What are operators in Python?

**ANSWER:** Operators are used to perform operations on variables and values. Python divides the operators in the multiple groups like Arithmetic operators, Assignment operators, Comparison operators, Logical operators, Identity operators, Membership operators and Bitwise operators.

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

**ANSWER:** / is regular division(returns float) and // is floor division(returns int).

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

iNeuroniNeuroniNeuroniNeuron

**ANSWER:** print("iNeuroniNeuroniNeuroniNeuron")

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

**ANSWER:**

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

if number % 2 == 0:

print(number, "is even")

else:

print(number, "is odd")

Q18. What are boolean operator?

**ANSWER:** In Python, Boolean operators are used to perform logical operations between two Boolean expressions, which evaluate to either True or False. There are three Boolean operators in Python: and, or, not

Q19. What will the output of the following?

1 or 0

**ANSWER:** 1

0 and 0

**ANSWER:** 0

True and False and True

**ANSWER:** False

1 or 0 or 0

**ANSWER:** 1

Q20. What are conditional statements in Python?

**ANSWER:** Conditional statements in Python are used to execute different parts of code based on certain conditions. In Python, we use if, elif and else statements to create conditional statements.

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

**ANSWER:** The if keyword is used to check whether a particular condition is True or False. If the condition is True, the code block following the if statement is executed. If the condition is False, the code block following the if statement is skipped. The elif keyword is used to add additional conditions to the if statement. If the condition in the if statement is False, the program checks the condition following elif. If the condition in the elif statement is True, the code block following the elif statement is executed. If the condition in the elif statement is False, the program moves on to the next elif statement, if there is one. The else keyword is used to specify what code should be executed if none of the conditions specified in the if or elif statements are True.

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".

**ANSWER:**

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]

**ANSWER:**

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

# Initialize sum variable to 0

sum = 0

# loop through each number in the list

for num in numbers:

# check if the number is even

if num % 2 == 0:

# add the even number to the sum variable

sum += num

# print the sum of even numbers

print("Sum of even numbers:", sum)

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

**ANSWER:**

# take input from the user

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

num3 = float(input("Enter the third number: "))

# check which number is the greatest

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]

**ANSWER:**

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

# loop through each number in the list

for num in numbers:

# Check if the number is divisible by 5

if num % 5 == 0:

# if the number is greater than 150, skip to the next number

if num > 150:

continue

# if the number is greater than 500, stop the loop

elif num > 500:

break

# otherwise, print the number

else:

print(num)

# Python mega assignment

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

**ANSWER:** In Python, a string is a sequence of characters enclosed in either single or double quotes. We can also declare multi-line strings using triple quotes. We can declare a string in Python by assigning a value to a variable using quotes.

For example: my\_string = "Hello, World!"

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

**ANSWER:** In Python, we can access individual characters of a string using its index. String indexing in Python is zero-based, which means the first character has an index of 0, the second has an index of 1, and so on.

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"

**ANSWER:**

string = "Big Data iNeuron"

desired\_output = string.split()[-1]

print(desired\_output) # Output: iNeuron

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

string = "Big Data iNeuron"

desired\_output = "norueNi"

**ANSWER:**

string = "Big Data iNeuron"

desired\_output = string.split()[-1][::-1]

print(desired\_output) # Output: norueNi

Q30. Reverse the string given in the above question.

**ANSWER:**

string = "Big Data iNeuron"

reversed\_string = string[::-1]

print(reversed\_string) # Output: norueNi ataD giB

Q31. How can you delete entire string at once?

**ANSWER:**

In Python, we cannot delete the entire string at once. However, we can delete a string variable by using the del keyword followed by the variable name.

For Example:

my\_string = "Hello, world!"

del my\_string

Q32. What is escape sequence?

**ANSWER:** An escape sequence is a combination of characters in a string that represent a special character or symbol that cannot be represented directly. It starts with a backslash (\) followed by a character or sequence of characters that represent the desired special character.

Q33. How can you print the below string?

'iNeuron's Big Data Course'

**ANSWER:** print("iNeuron's Big Data Course") or print('iNeuron\'s Big Data Course')

Q34. What is a list in Python?

**ANSWER:** A list is a collection of ordered, mutable and heterogeneous elements enclosed in square brackets separated by commas. Each element in a list can be of any data type, including other lists. Lists are one of the most commonly used data structures in Python and are very versatile.

Q35. How can you create a list in Python?

**ANSWER:** In Python, we can create a list by enclosing a sequence of comma-separated elements inside square brackets []. Here is an example: my\_list = [1, 2, 3, "apple", True]

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

**ANSWER:** In Python, we can access the elements in a list by using their index values. List indexes start at 0, so the first element in the list has an index of 0, the second element has an index of 1, and so on.

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

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

**ANSWER:**

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

print(lst[4][2])

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

**ANSWER:** A tuple is created using parentheses instead of square brackets. A tuple is immutable, which means that once it is created, its contents cannot be modified.

The main difference between tuples and lists is that tuples cannot be modified once they are created, while lists can be modified by adding, removing, or changing elements.

Q41. How can you create a tuple in Python?

**ANSWER:** A tuple is created in Python by enclosing comma-separated values in parentheses (). Here's an example: my\_tuple = (1, 2, 3, "apple", "banana")

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

**ANSWER:** We cannot add an element to a tuple once it has been created, because tuples are immutable.

my\_tuple = (1, 2, 3, "apple", "banana")

my\_tuple.append("Mohan") # If we try to add an element to a tuple using the append() or insert() methods, we will get a TypeError because these methods are not defined for tuples.

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

**ANSWER:** No, two tuples cannot be appended because tuples are immutable, meaning their elements cannot be modified after creation. We can only create a new tuple by concatenating two or more tuples.

Here is an example code for concatenating two tuples:

tuple1 = (1, 2, 3)

tuple2 = (4, 5, 6)

tuple3 = tuple1 + tuple2

print(tuple3) # Output (1, 2, 3, 4, 5, 6)

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

**ANSWER:**

my\_tuple = tuple(input("Enter elements of the tuple separated by spaces: ").split())

print("Count of elements in the tuple:", len(my\_tuple))

Q45. What are sets in Python?

**ANSWER:** In Python, sets are unordered collection of unique elements/items. The elements in a set must be unique, immutable, and hashable. Sets are represented using curly braces {} or set() constructor function.

Q46. How can you create a set?

**ANSWER:** We can create a set in Python by enclosing a comma-separated sequence of elements within curly braces {}, or by using the set() constructor function.

Here are a few examples:

# Creating a set using curly braces

my\_set = {1, 2, 3, 4, 5}

# Creating a set using the set() constructor

my\_set = set([1, 2, 3, 4, 5])

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

**ANSWER:**

my\_set = {"apple", "banana", "cherry"}

my\_set.add("iNeuron")

print(my\_set) # Output {'banana', 'apple', 'iNeuron', 'cherry'}

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

**ANSWER:** The add() function can only add one element at a time to a set. If we want to add multiple values to a set, we can use the update() function.

Here's an example code:

my\_set = {1, 2, 3}

my\_set.update([4, 5, 6])

print(my\_set) # Output {1, 2, 3, 4, 5, 6}

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

**ANSWER:** The main difference between add() and update() is that add() can add only one element at a time, whereas update() can add multiple elements at once.

Q50. What is clear() in sets?

**ANSWER:** The clear() method in sets is a built-in function that removes all the elements from the set, making it an empty set. This method modifies the original set and does not return any value.

Q51. What is frozen set?

**ANSWER:** In Python, a frozen set is an immutable collection of hashable elements, which means once it is created, its contents cannot be changed. A frozen set is similar to a set in Python, but the difference is that it cannot be modified after it has been created. Frozen sets are created by calling the built-in function frozenset(), which takes an iterable object as its argument and returns a new frozen set object.

Q52. How is frozen set different from set?

**ANSWER:** The main difference between a frozen set and a regular set in Python is that a frozen set is immutable, while a regular set is mutable. Once a frozen set is created, its elements cannot be added, removed or modified, while a regular set can be modified by adding or removing elements.

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

**ANSWER:** In Python, the union() method is used to combine two or more sets and return a new set that contains all the unique elements from the input sets. The union() method does not modify the original sets but returns a new set as the result of the operation.

# Creating two sets

set1 = {1, 2, 3, 4}

set2 = {3, 4, 5, 6}

# Using the union() method to combine the two sets

new\_set = set1.union(set2)

# Printing the new set

print(new\_set) #output {1, 2, 3, 4, 5, 6}

As we can see, the union() method combines the two input sets (set1 and set2) and returns a new set (new\_set) that contains all the unique elements from the input sets. In this case, new\_set contains the elements 1, 2, 3, 4, 5, and 6.

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

**ANSWER:** In Python, the intersection() method is used to find the common elements between two or more sets and return a new set that contains only those elements that are present in all the input sets. The intersection() method does not modify the original sets but returns a new set as the result of the operation.

# Creating two sets

set1 = {1, 2, 3, 4}

set2 = {3, 4, 5, 6}

# Using the intersection() method to find the common elements

new\_set = set1.intersection(set2)

# Printing the new set

print(new\_set) #output {3, 4}

As we can see, the intersection() method finds the common elements between the two input sets (set1 and set2) and returns a new set (new\_set) that contains only those elements that are present in both sets. In this case, new\_set contains the elements 3 and 4, which are present in both set1 and set2.

Q55. What is dictionary ibn Python?

**ANSWER:** In Python, a dictionary is a built-in data structure that allows us to store and retrieve key-value pairs. Dictionaries in Python are created using curly braces ({}) and consist of a series of key-value pairs, where each key is unique and maps to a value.

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

**ANSWER:** A dictionary in Python is different from other data structures because it is a mutable, unordered collection of key-value pairs, accessed by keys, does not allow duplicate keys, and is not indexable.

Q57. How can we declare a dictionary in Python?

**ANSWER:** In Python, we can declare a dictionary using curly braces {} or the built-in dict() constructor.

Q58. What will the output of the following?

var = {}

print(type(var))

**ANSWER:** <class 'dict'>

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

**ANSWER:**

* Using square bracket notation: We can add a new key-value pair to a dictionary using the square bracket notation by assigning a value to a new key.

# Create an empty dictionary

my\_dict = {}

# Add a new key-value pair to the dictionary

my\_dict['name'] = 'John'

# Print the updated dictionary

print(my\_dict)

* Using the update() method: We can also add one or more key-value pairs to a dictionary using the update() method. The update() method takes a dictionary, a list of tuples, or keyword arguments as input.

# Create a dictionary with some key-value pairs

my\_dict = {'name': 'John', 'age': 25}

# Add one or more key-value pairs to the dictionary using the update() method

my\_dict.update({'city': 'New York', 'country': 'USA'})

# Print the updated dictionary

print(my\_dict)

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

**ANSWER:**

# Create a dictionary

my\_dict = {'name': 'John', 'age': 25, 'city': 'New York'}

# Access all the values in the dictionary

for value in my\_dict.values():

print(value)

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

**ANSWER:**

# Create a nested dictionary

my\_dict = {'person1': {'name': 'John', 'age': 25, 'city': 'New York'},

'person2': {'name': 'Jane', 'age': 30, 'city': 'Los Angeles'},

'person3': {'name': 'Bob', 'age': 40, 'city': 'Chicago'}}

# Access all the elements in the inner dictionary

for person, details in my\_dict.items():

print("Details of {person}:")

for key, value in details.items():

print("\t{key}: {value}")

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

**ANSWER:** In Python, the get() function is a dictionary method that is used to retrieve the value of a specified key in a dictionary. It takes one or two arguments - the key whose value is to be retrieved and an optional default value to be returned if the key is not present in the dictionary.

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

**ANSWER:** In Python, the items() function is a dictionary method that returns a view object that contains a list of tuple pairs, where each tuple represents a key-value pair in the dictionary. The order of the tuple pairs is not guaranteed, and it may vary each time the items() function is called.

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

**ANSWER:** In Python, the pop() function is a dictionary method that is used to remove and return the value associated with a specified key in a dictionary. It takes one argument - the key of the item to be removed.

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

**ANSWER:** In Python, the popitems() function is a dictionary method that is used to remove and return an arbitrary (key, value) tuple from a dictionary. The method does not take any arguments.

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

**ANSWER:** The keys() function is a built-in Python method that can be used to extract all the keys from a dictionary. It returns a view object that displays a list of all the keys in the dictionary.

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

**ANSWER:** The values() function is a built-in Python method that can be used to extract all the values from a dictionary. It returns a view object that displays a list of all the values in the dictionary.

Q68. What are loops in Python?

**ANSWER:** In Python, loops are a way to execute a block of code repeatedly based on certain conditions.

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

**ANSWER:** There are two types of loops in Python:

* for loop: The for loop is used to iterate over a sequence (such as a list, tuple, or string) and execute a block of code for each item in the sequence.
* while loop: The while loop is used to execute a block of code repeatedly as long as a certain condition is true. The condition is checked before each iteration of the loop, and the loop will continue until the condition is false.

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

**ANSWER:** The main difference between for and while loops in Python is that a for loop is used for iterating over a sequence of items, whereas a while loop is used for executing a block of code repeatedly as long as a certain condition is true.

Q71. What is the use of continue statement?

**ANSWER:** The continue statement is a control flow statement in Python that is used to skip over the remaining statements in the current iteration of a loop and move on to the next iteration.

Q72. What is the use of break statement?

**ANSWER:** The break statement is a control flow statement in Python that is used to terminate the execution of a loop prematurely.

Q73. What is the use of pass statement?

**ANSWER:** The pass statement is a null operation in Python that does nothing. It is used as a placeholder in situations where a statement is required syntactically, but no action is required or desired.

The pass statement is typically used in the following situations:

* Empty function or class definition
* Placeholder for code to be written later

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

**ANSWER:** The range() function is a built-in function in Python that is used to generate a sequence of numbers.

Q75. How can you loop over a dictionary?

**ANSWER:** In Python, we can loop over a dictionary using a for loop. When we loop over a dictionary, the loop iterates over the keys of the dictionary by default. To access the values corresponding to the keys, we can use the dictionary indexing operator ([]) with the key as the index.

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

**ANSWER:**

# Take input from the user

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

# Initialize the factorial variable to 1

factorial = 1

# Calculate the factorial of the number

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

factorial \*= i

# Print the factorial

print("The factorial of", num, "is", factorial)

Q77. Write a Python program to calculate the simple interest. Formula to calculate simple interest is SI = (PRT)/100

**ANSWER:**

# Take input from the user

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

rate = float(input("Enter the annual interest rate: "))

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

# Calculate the simple interest

simple\_interest = (principal \* rate \* time) / 100

# Print the result

print("The simple interest is:", simple\_interest)

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

**ANSWER:**

# Take input from the user

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

rate = float(input("Enter the annual interest rate: "))

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

# Calculate the compound interest

amount = principal \* (1 + rate / 100) \*\* time

# Calculate the interest earned

interest = amount - principal

# Print the result

print("The compound interest is:", interest)

print("The total amount is:", amount)

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

**ANSWER:**

# Take input from the user

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

# Check if the number is prime or not

if num > 1:

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

if num % i == 0:

print(num, "is not a prime number")

break

else:

print(num, "is a prime number")

else:

print(num, "is not a prime number")

Q80. Write a Python program to check Armstrong Number.

**ANSWER:**

# Take input from the user

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

# Calculate the number of digits

num\_digits = len(str(num))

# Calculate the sum of the cubes of the digits

sum = 0

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* num\_digits

temp //= 10

# Check if the number is an Armstrong number or not

if num == sum:

print(num, "is an Armstrong number")

else:

print(num, "is not an Armstrong number")

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

**ANSWER:**

def fibonacci(n):

if n <= 1:

return n

else:

return fibonacci(n-1) + fibonacci(n-2)

# Calling the function

n = 10

print("The", n, "th Fibonacci number is:", fibonacci(n))

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

**ANSWER:**

def interchange\_first\_last(lst):

if len(lst) < 2:

return lst

else:

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

return lst

# Calling the function

lst = [1, 2, 3, 4, 5]

print("Original List: ", lst)

print("List after interchange: ", interchange\_first\_last(lst))

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

**ANSWER:**

def swap\_elements(lst, i, j):

if i < 0 or i >= len(lst) or j < 0 or j >= len(lst):

return lst

else:

lst[i], lst[j] = lst[j], lst[i]

return lst

# Calling the function

lst = [1, 2, 3, 4, 5]

print("Original List: ", lst)

print("List after swapping elements: ", swap\_elements(lst, 1, 3))

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

**ANSWER:**

def find\_n\_largest(lst, n):

if len(lst) < n:

return "List has fewer than {} elements".format(n)

else:

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

return sorted\_lst[:n]

# Calling the function

lst = [12, 45, 2, 41, 31, 10, 8, 6, 4]

n = 3

print("The {} largest elements in the list are: {}".format(n, find\_n\_largest(lst, n)))

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

**ANSWER:**

def cumulative\_sum(lst):

if len(lst) == 0:

return lst

else:

result\_lst = [lst[0]]

for i in range(1, len(lst)):

result\_lst.append(result\_lst[i-1] + lst[i])

return result\_lst

# Calling the function

lst = [1, 2, 3, 4, 5]

print("Original List: ", lst)

print("Cumulative Sum List: ", cumulative\_sum(lst))

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

**ANSWER:**

def is\_palindrome(string):

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

return True

else:

return False

# Calling the function

string1 = "racecar"

print("Is {} a palindrome? {}".format(string1, is\_palindrome(string1)))

string2 = "hello"

print("Is {} a palindrome? {}".format(string2, is\_palindrome(string2)))

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

**ANSWER:**

def remove\_ith\_element(string, i):

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

return "Invalid index"

else:

return string[:i] + string[i+1:]

# Calling the function

string = "hello world"

i = 4

print("Original String: ", string)

print("String after removing {}'th element: ".format(i), remove\_ith\_element(string, i))

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

**ANSWER:**

def is\_substring(string, substring):

if substring in string:

return True

else:

return False

# Calling the function

string = "hello world"

substring1 = "world"

print("Is {} a substring of {}? {}".format(substring1, string, is\_substring(string, substring1)))

substring2 = "goodbye"

print("Is {} a substring of {}? {}".format(substring2, string, is\_substring(string, substring2)))

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

**ANSWER:**

def find\_words\_greater\_than\_k(string, k):

word\_list = string.split()

result\_list = []

for word in word\_list:

if len(word) > k:

result\_list.append(word)

return result\_list

# Calling the function

string = "The quick brown fox jumps over the lazy dog"

k = 4

print("Words greater than length {}: {}".format(k, find\_words\_greater\_than\_k(string, k)))

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

**ANSWER:**

def get\_unique\_values(dict\_obj):

values\_list = list(dict\_obj.values())

return list(set(values\_list))

# Calling the function

dict\_obj = {1: 'apple', 2: 'banana', 3: 'apple', 4: 'cherry', 5: 'banana'}

print("Original Dictionary: ", dict\_obj)

print("Unique values in the dictionary: ", get\_unique\_values(dict\_obj))

Q91. Write a Python program to merge two dictionary.

**ANSWER:**

def merge\_dicts(dict1, dict2):

merged\_dict = {\*\*dict1, \*\*dict2}

return merged\_dict

# Calling the function

dict1 = {1: 'apple', 2: 'banana'}

dict2 = {3: 'cherry', 4: 'durian'}

print("Original Dictionaries: ")

print("Dict1: ", dict1)

print("Dict2: ", dict2)

print("Merged Dictionary: ", merge\_dicts(dict1, dict2))

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}

**ANSWER:**

def tuples\_to\_dict(tuple\_list):

dict\_obj = {}

for t in tuple\_list:

dict\_obj[t[0]] = t[1]

return dict\_obj

# Calling the function

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

print("Original List of Tuples: ", tuple\_list)

print("Dictionary: ", tuples\_to\_dict(tuple\_list))

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)]

**ANSWER:**

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)]

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)]

**ANSWER:**

from itertools import product

def get\_combinations(tuple1, tuple2):

return list(product(tuple1, tuple2)) + list(product(tuple2, tuple1))

# Calling the function

test\_tuple1 = (7, 2)

test\_tuple2 = (7, 8)

print("Tuple 1: ", test\_tuple1)

print("Tuple 2: ", test\_tuple2)

print("All Combinations: ", get\_combinations(test\_tuple1, test\_tuple2))

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)]

**ANSWER:**

def sort\_tuples\_by\_second\_item(tuple\_list):

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

# Calling the function

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

print("Original List of Tuples: ", tuple\_list)

print("Sorted List of Tuples: ", sort\_tuples\_by\_second\_item(tuple\_list))

Q96. Write a python program to print below pattern.

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

**ANSWER:**

def print\_pattern(n):

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

print('\* ' \* i)

# Calling the function

print\_pattern(5)

Q97. Write a python program to print below pattern.

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**ANSWER:**

def print\_pattern(n):

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

print(' '\*(n-i) + '\*'\*i)

# Calling the function

print\_pattern(5)

Q98. Write a python program to print below pattern.

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

**ANSWER:**

def print\_pattern(n):

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

print(' '\*(n-i) + '\* '\*i)

# Calling the function

print\_pattern(5)

Q99. Write a python program to print below pattern.

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

**ANSWER:**

def print\_pattern(n):

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

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

print(j, end=' ')

print()

# Calling the function

print\_pattern(5)

Q100. Write a python program to print below pattern.

A

B B

C C C

D D D D

E E E E E

**ANSWER:**

def print\_pattern(n):

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

print((chr(64+i)+' ')\*i)

# Calling the function

print\_pattern(5)