117 Python Problem:

Python is a popular high-level programming language known for its simplicity and readability. We are going to explore a variety of programming problems in Python covering different topics such as ***variables, strings, typecasting, data types, loops, lists, dictionaries, and recursion***. Python supports multiple programming paradigms, including object-oriented, functional, and procedural styles. Its syntax emphasizes code readability, which makes it an excellent choice for beginners and experienced developers alike. Here are 115 problems related to basic Python: (Become a Python Developer)

- **Basic Problems:**

1. Variable Swap: Write a Python program to swap the values of two variables without using a temporary variable.
2. Even or Odd: Write a Python program that takes an integer as input and prints whether it is even or odd.
3. String Reverse: Write a Python function to reverse a given string and return the reversed string.
4. Type Conversion: Given a list of integers, write a Python program to convert each element of the list to a string.
5. Temperature Converter: Write a Python program that converts a temperature in Celsius to Fahrenheit. Take the Celsius temperature as input from the user.
6. Data Type Checker: Write a Python function that takes a variable as input and returns the data type of the variable as a string (e.g., "int", "float", "str", "list", etc.).
7. String Palindrome: Write a Python function to check if a given string is a palindrome or not.
8. String Reversal with Slicing: Write a Python function to reverse a given string using slicing.
9. String Concatenation: Write a Python program that takes two strings as input and concatenates them into a single string without using the `+` operator.
10. Typecasting Challenge: Given three variables: `a = '100'`, `b = 25`, and `c = '10.5'`, write a Python program to perform the following operations and print the results: – Convert `a` to an integer and add it to `b`. – Convert `c` to a float and subtract it from the result of the first operation. – Convert the final result to a string and concatenate it with the string " is the answer."

- **Python Conditional Statements:**

11. Positive, Negative, or Zero: Write a Python program that takes a number as input and prints whether it is positive, negative, or zero.
12. Largest of Three Numbers: Write a Python program that takes three numbers as input and prints the largest among them.

13. Leap Year Checker: Write a Python program that takes a year as input and determines if it is a leap year or not.
14. Grades Classification: Write a Python program that takes a student's percentage as input and prints their corresponding grade according to the following criteria: – 90% or above: A+ – 80-89%: A – 70-79%: B – 60-69%: C – Below 60%: Fail
15. Vowel or Consonant: Write a Python program that takes a single character as input and determines whether it is a vowel or a consonant.
16. Time Classification: Write a Python program that takes the time in hours (24-hour format) as input and prints "Good Morning", "Good Afternoon", "Good Evening", or "Good Night" based on the time.
17. Triangle Type Checker: Write a Python program that takes three sides of a triangle as input and determines whether it forms an equilateral, isosceles, or scalene triangle.
18. Quadratic Equation Solver: Write a Python program that takes the coefficients (a, b, c) of a quadratic equation as input and calculates and prints the real roots (if they exist) or a message indicating the complex roots.
19. Number Ranges: Write a Python program that takes an integer as input and prints whether the number falls within the ranges: 0-50, 51-100, 101-150, or above 150.

- **For & While Loops:**

20. Sum of N Numbers: Write a Python program using a for loop to calculate the sum of the first N natural numbers, where N is taken as input from the user.
21. Factorial Calculator: Write a Python program using a while loop to calculate the factorial of a given number N.
22. Table of a Number: Write a Python program using a for loop to print the multiplication table of a given number N.
23. Count Digits in a Number: Write a Python program using a while loop to count the number of digits in a given integer N.
24. Fibonacci Sequence: Write a Python program using a for loop to generate the Fibonacci sequence up to a given limit N.
25. Sum of Even Numbers: Write a Python program using a while loop to calculate the sum of all even numbers between 1 and N, where N is taken as input from the user.
26. Print Patterns: Write a Python program using nested for loops to print various patterns, such as a right-angled triangle, an inverted right-angled triangle, and so on.
27. Prime Number Checker: Write a Python program using a while loop to check if a given number N is prime or not.
28. List Manipulation: Given a list of integers, write a Python program using a for loop to find the sum, average, maximum, and minimum values in the list.
29. Reverse String: Write a Python program using a while loop to reverse a given string.

- **Nested Loops:**

30. Multiplication Table: Write a Python program using nested loops to print the multiplication table from 1 to 10.

31. Print Patterns: Write a Python program using nested loops to print the following pattern:

```
*
**
***
****
*****
```

32. Matrix Multiplication: Write a Python program using nested loops to multiply two matrices.

33. Chessboard Pattern: Write a Python program using nested loops to print a chessboard pattern (alternating "X" and "O" characters) of size 8×8.

34. Number Pyramid: Write a Python program using nested loops to print a number pyramid like the following: 1 22 333 4444 55555

- **List Problems:**

34. List Sum: Write a Python program to find the sum of all elements in a given list of integers.

35. List Average: Write a Python program to calculate the average of all elements in a given list of integers.

36. List Max and Min: Write a Python program to find the maximum and minimum values in a given list of integers.

37. List Sorting: Write a Python program to sort a list of integers in ascending order.

38. List Filtering: Given a list of integers, write a Python program to create a new list that contains only the even numbers from the original list.

39. List Reversal: Write a Python program to reverse a given list without using any built-in functions.

40. List Manipulation: Given two lists of integers, write a Python program to create a new list that contains elements common to both lists.

41. List Element Count: Write a Python program to count the occurrences of a specific element in a given list.

42. List Duplicates Removal: Write a Python program to remove duplicates from a given list while preserving the order of the elements.

43. List Comprehension: Given a list of integers, write a Python program to create a new list that contains the squares of the elements using list comprehension.

- **Nested List Problems:**

44. Matrix Addition: Write a Python program to add two matrices represented as nested lists.

45. Flatten Nested List: Write a Python program to flatten a given nested list and convert it into a single-dimensional list.

46. List Element Frequency: Given a nested list containing lists of integers, write a Python program to count the frequency of each element in the entire nested list.

47. Transpose Matrix: Write a Python program to transpose a given matrix represented as a nested list.
48. List of Lists Concatenation: Given a list of nested lists, write a Python program to concatenate all the sublists into a single flat list.

- **Tuple Problems:**

49. Tuple Concatenation: Write a Python program to concatenate two tuples and create a new tuple.
50. Tuple Unpacking: Given a tuple with three elements (x, y, z), write a Python program to unpack the tuple and assign the values to three variables.
51. Tuple Sorting: Write a Python program to sort a tuple of integers in ascending order.
52. Tuple Frequency Count: Given a tuple containing various elements, write a Python program to count the frequency of a specific element in the tuple.
53. Tuple to List: Write a Python program to convert a tuple into a list. **54. Tuple Reversal:** Write a Python program to reverse a tuple without using any built-in functions.
55. Tuple Slicing: Given a tuple, write a Python program to extract a slice of elements from it.
56. Tuple Operations: Given two tuples of integers, write a Python program to perform element-wise addition, subtraction, and multiplication and create new tuples for each operation.
57. Tuple Membership Test: Write a Python program that takes an element as input and checks if it exists in a given tuple.
58. Tuple Packing: Write a Python program to pack three variables into a single tuple and print the tuple.

- **Nested List Problems:**

59. Nested List Element Access: Given a nested list, write a Python program to access and print specific elements from it.
60. Nested List Flattening: Write a Python program to flatten a nested list and convert it into a single-dimensional list.
61. Nested List Sorting: Given a nested list containing lists of integers, write a Python program to sort the sublists based on their lengths.
62. List of Tuples Conversion: Given a nested list containing tuples of (x, y) coordinates, write a Python program to convert it into a list of x-coordinates and a list of y-coordinates.
63. Matrix Transpose: Write a Python program to transpose a given matrix represented as a nested list.
64. Nested List Concatenation: Given a list of nested lists, write a Python program to concatenate all the sublists into a single flat list.
65. Count Even Numbers: Write a Python program to count the number of even numbers in a nested list.

66. Maximum Element in Nested List: Write a Python program to find the maximum element in a nested list of integers.
67. Diagonal Sum of Matrix: Given a square matrix represented as a nested list, write a Python program to calculate the sum of the elements in the main diagonal.
68. Nested List Element Search: Write a Python program to search for a specific element in a nested list and return its position (row and column indices).

- **Set Problems:**

69. Duplicate Removal: Write a Python program that takes a list of elements as input and creates a new set containing only the unique elements from the list.
70. Set Intersection: Given two sets A and B, write a Python program to find their intersection and print the common elements.
71. Set Union: Given two sets A and B, write a Python program to find their union and print all the distinct elements from both sets.
72. Set Difference: Given two sets A and B, write a Python program to find the difference between set A and set B (i.e., elements present in A but not in B) and print the result.
73. Set Symmetric Difference: Given two sets A and B, write a Python program to find the symmetric difference between the two sets (i.e., elements that are present in either set A or set B, but not in both) and print the result.
74. Set Operations: Given three sets A, B, and C, write a Python program to find and print the intersection of A and B, the union of B and C, and the difference between C and A.
75. Set Subset Check: Given two sets A and B, write a Python program to check if set A is a subset of set B and print the result.
76. Set Superset Check: Given two sets A and B, write a Python program to check if set A is a superset of set B and print the result.
77. Set Length Check: Write a Python program that takes a set as input and prints the number of elements in the set.
78. Set Membership Test: Write a Python program that takes an element as input and checks if it exists in a given set. Print “Found” if the element is present and “Not Found” otherwise.

- **Dictionary Problems:**

79. Dictionary Manipulation: Given a dictionary with student names as keys and their corresponding scores as values, write a Python program to add a new student to the dictionary and update the score of an existing student.
80. Dictionary Keys and Values: Write a Python program that takes a dictionary as input and prints all the keys and values in separate lines.
81. Dictionary Length: Write a Python program to calculate and print the number of key-value pairs in a given dictionary.
82. Dictionary Value Search: Given a dictionary of items and their prices, write a Python program to search for an item based on its price and print the item’s name.

83. Dictionary Merging: Given two dictionaries, write a Python program to merge them into a single dictionary and print the result.
84. Dictionary Key Removal: Given a dictionary of items and their quantities, write a Python program to remove a specific item from the dictionary based on user input.
85. Dictionary Sorting: Given a dictionary with names as keys and corresponding ages as values, write a Python program to sort the dictionary based on age in ascending order.
86. Dictionary Frequency Count: Write a Python program that takes a string as input and creates a dictionary containing each character as a key and its frequency as the value.
87. Dictionary Comprehension: Given a list of integers, write a Python program to create a dictionary where the keys are the elements from the list, and the values are their squares.
88. Dictionary Key Check: Write a Python program that takes a key as input and checks if it exists in a given dictionary. Print "Key Found" if the key is present and "Key Not Found" otherwise.

- **Nested Dictionary Problems:**

89. Access Nested Dictionary: Given a nested dictionary containing student details, write a Python program to access and print specific information such as a student's name, age, and address.
90. Nested Dictionary Length: Write a Python program to calculate and print the total number of key-value pairs in a nested dictionary.
91. Nested Dictionary Update: Given a nested dictionary of employee details, write a Python program to update an employee's salary based on their employee ID.
92. Nested Dictionary Sorting: Given a nested dictionary containing product details (product name, price, and quantity), write a Python program to sort the products based on their prices in ascending order.
93. Nested Dictionary Key Search: Write a Python program that takes a key as input and searches for it in a nested dictionary. If found, print the corresponding value, otherwise, print "Key Not Found."

- **Break & Continue:**

94. Prime Number Checker: Write a Python program that takes a number as input and determines if it is a prime number or not. Use a `for` loop to check for factors. If a factor is found, `break` out of the loop.
95. Even Number Printer: Write a Python program to print all even numbers from 1 to 20. Use a `for` loop and `continue` to skip odd numbers.
96. Password Validator: Write a Python program that takes a password as input and checks if it meets the following criteria: at least 8 characters long, contains both uppercase and lowercase letters, and has at least one digit. If the password is valid, print "Password accepted." If not, use `continue` to prompt the user to enter a valid password.

97. Divisible by 3 or 5: Write a Python program to print all numbers from 1 to 50 that are divisible by either 3 or 5. Use a `for` loop and `continue` to skip numbers that are not divisible by either 3 or 5.
98. Positive Number Sum: Write a Python program that takes positive numbers as input until a negative number is entered. Then, calculate and print the sum of all positive numbers entered. Use a `while` loop and `break` to exit the loop when a negative number is encountered.
99. Word Palindrome Checker: Write a Python program that takes a word as input and checks if it is a palindrome (reads the same forwards and backward). Use `continue` to skip checking the word if its length is less than 3 characters.
100. Odd Number Finder: Write a Python program to find the first odd number from a list of integers. Use a `for` loop and `break` to stop the loop when the first odd number is found.
101. Number Guessing Game: Write a Python program that generates a random number between 1 and 100 and lets the user guess the number. Use a `while` loop, `break` when the correct number is guessed, and `continue` to keep prompting the user until they guess correctly.
102. Vowel Counter: Write a Python program that takes a string as input and counts the number of vowels (a, e, i, o, u) in it. Use a `for` loop and `continue` to skip counting non-vowel characters.
103. Unique Characters: Write a Python program that takes a string as input and checks if it contains all unique characters (no character repeats). Use a `for` loop and `break` when a character repeats.

- **Functions Problems:**

104. Factorial Calculator: Write a Python function called `factorial` that takes an integer as input and returns its factorial. Test the function with different values.
105. Palindrome Checker: Write a Python function called `is_palindrome` that takes a string as input and returns `True` if it is a palindrome and `False` otherwise. Test the function with different words.
106. Even or Odd Checker: Write a Python function called `even_or_odd` that takes an integer as input and returns “Even” if the number is even and “Odd” if the number is odd. Test the function with different numbers.
107. List Sum Calculator: Write a Python function called `list_sum` that takes a list of integers as input and returns the sum of all elements in the list. Test the function with different lists.
108. Greatest Common Divisor (GCD) Calculator: Write a Python function called `gcd` that takes two integers as input and returns their greatest common divisor. Test the function with different pairs of numbers.

109. Leap Year Checker: Write a Python function called `is_leap_year` that takes a year as input and returns `True` if it is a leap year and `False` otherwise. Test the function with different years.

- **Nested Functions Problems:**

110. Math Operations: Write a Python function called `math_operations` that takes three numbers and a string representing an operation ('add', 'subtract', 'multiply', or 'divide'). The function should return the result of the specified operation on the three numbers. Implement the math operations as nested functions.

1111. Greeting Generator: Write a Python function called `greeting_generator` that takes a name as input and returns a greeting message using nested functions. The greeting message should be customizable (e.g., "Hello, {name}! How are you today?").

112. Temperature Converter: Write a Python function called `temperature_converter` that takes a temperature value and a string representing the scale ('C' for Celsius or 'F' for Fahrenheit) as input. The function should convert the temperature from one scale to the other using nested functions and return the converted value.

- **Recursion: [Video]**

113. Factorial Calculation: Write a recursive Python function called `factorial` that takes a non-negative integer as input and returns its factorial.

114. Fibonacci Series: Write a recursive Python function called `Fibonacci` that takes an integer N as input and returns the Nth number in the Fibonacci series. The Fibonacci series is defined as follows: $F(0) = 0$, $F(1) = 1$, and $F(n) = F(n-1) + F(n-2)$ for $n > 1$.

115. The sum of Digits: Write a recursive Python function called `sum_of_digits` that takes an integer as input and returns the sum of its digits.

116. Binary Search: Write a recursive Python function called `binary_search` that takes a sorted list and a target value as input and returns the index of the target value in the list using binary search. If the target value is not in the list, return -1.

117. Power Calculation: Write a recursive Python function called `power` that takes two positive integers, base and exponent, as input and returns the value of base raised to the exponent.

We have explored a variety of programming problems in Python covering different topics such as variables, strings, typecasting, data types, loops, lists, dictionaries, and recursion. These problems are designed to challenge your understanding of Python concepts and provide opportunities to practice problem-solving and coding skills. By working through these problems, you can gain a deeper understanding of Python syntax, data structures, and control flow. Additionally, solving problems involving nested structures, such as nested loops, nested lists, and nested functions, can improve your ability to work with complex data arrangements.