Python Basic Exercises

Variables

1. Basic Variable Assignment

   - Create a variable `age` and assign it your age as a number.

   - Create a variable `name` and assign it your name as a string.

   - Print both variables.

2. Simple Calculations

   - Create two variables `x` and `y` and assign them the values `10` and `5`, respectively.

   - Create a new variable `sum` and set it to the sum of `x` and `y`.

   - Print `sum`, the result of subtracting `y` from `x`, multiplying `x` and `y`, and dividing `x` by `y`.

3. Swapping Variables

   - Given two variables `a = 3` and `b = 7`, swap their values so that `a` contains `7` and `b` contains `3`.

   - Print the variables to confirm the swap.

4. Calculating Area

   - Write a program that calculates the area of a rectangle.

   - Define variables `length` and `width`, assign them values, then create a variable `area` as `length * widt

   - Print the area.

Strings

5. Basic String Operations

   - Create a variable `greeting` and set it to 'Hello, world!'.

   - Print the length of the string, the first character, and the last character.

6. String Concatenation

   - Create two variables, `first_name` and `last_name`, and set them to your first and last names.

   - Concatenate them into a new variable `full_name` with a space in between, then print `full_name`.

7. String Formatting

- Use an f-string to print a sentence that says, 'My name is [name] and I am [age] years old,' where `nam

8. Upper and Lower Case
   - Create a variable `quote` and set it to a famous quote, e.g., 'To be or not to be, that is the question.'
   - Print the quote in all uppercase letters, then print it in all lowercase letters.
9. String Slicing
   - Given a variable `word = 'Python'`, print:
     - The first three characters.
     - The last three characters.
     - The word in reverse.
10. Replacing Text
    - Create a variable `sentence` and set it to 'I love programming in Python.'
    - Replace 'Python' with another programming language of your choice and print the new sentence.
11. Checking Substrings
    - Create a variable `text = 'The quick brown fox jumps over the lazy dog'`.
    - Check if the word 'fox' is in the text and print `True` or `False`.
    - Check if the word 'cat' is in the text and print `True` or `False`.

Lists

12. Creating and Printing Lists
    - Create a list with the names of three fruits, then print the list.
    - Add a new fruit to the list, and print the updated list.
    - Remove the first fruit in the list, and print the list again.
13. Accessing List Elements
    - Given the list `animals = ['cat', 'dog', 'rabbit', 'hamster']`, print:
      - The first animal in the list.
      - The last animal in the list.
      - The total number of animals in the list.

14. Modifying Lists

   - Create a list of five numbers, e.g., `[5, 10, 15, 20, 25]`.

   - Change the second number to `12` and print the list.

   - Add `30` to the end of the list and print the list again.

   - Remove the last item in the list and print the list.

15. List Slicing

   - Create a list of the first ten numbers.

   - Print the first five numbers.

   - Print the last three numbers.

   - Print the list in reverse order.

16. List Comprehension

   - Create a new list containing the squares of each number in the list `[1, 2, 3, 4, 5]`. Expected output: `[1

17. Counting Occurrences

   - Write a program that counts the occurrences of a specific item in a list. For example:

     fruits = ['apple', 'banana', 'apple', 'orange', 'banana', 'apple']

     - Count how many times `apple` appears in the list.

18. Finding the Index

   - Given the list `colors = ['red', 'blue', 'green', 'yellow', 'blue']`, find and print the index of the first occurren

19. List Concatenation

   - Create two lists of three numbers each, e.g., `[1, 2, 3]` and `[4, 5, 6]`.

   - Concatenate the two lists into one and print the result.

20. Removing Specific Items

   - Write a function `remove_all(lst, value)` that removes all occurrences of a given value from a list. For e

     numbers = [1, 2, 2, 3, 4, 2]

     remove_all(numbers, 2) # Expected result: [1, 3, 4]

21. List Sorting

   - Given a list of numbers, sort it in ascending order without using the `sort()` function. Try to implement it