List of Practicals

1. Write a python program to greet users with welcome messages using print() method.
2. Write a python program to demonstrate the creation of List data structure along with its methods - append(), extend(), insert(), remove(), clear(), index(), count(), sort(), reverse(), and copy().
    a. Demonstrate positive and negative indexing with python List.
    b. Demonstrate slicing operations on python List.
    c. Demonstrate updation on List elements in python.
    d. Demonstrate deletion of a single python list element and multiple elements using slicing operator.
3. Write a python program to demonstrate the creation of tuples along with its methods - count() and index().
    a. Demonstrate positive and negative indexing with python Tuple.
    b. Demonstrate slicing operations on python Tuple.
    c. Demonstrate updation on Tuple elements in python.
4. Write a python program to demonstrate the creation of a Dictionary *student* with the *name*, *age* and *branch* of a student.
    a. Demonstrate the updation of python dictionary.
    b. Demonstrate the removal of elements from the python dictionary.
    c. Demonstrate the use of following dictionary methods - clear(), copy(), get(), items(), keys(), popitem(), and values().
5. Demonstrate the use of basic string methods in python - lower(), upper(), join(), split(), find(), and replace().
6. Write a python program to implement basic arithmetic operations on user entered numbers.
7. Write a python program to count how many times a specific number is occurring in a list. Take user input for both numbers and a list.
8. Write a python program that takes N number of integers from the user in a python list. Create a function that takes the list of user entered numbers and returns MAX and MIN numbers from that list to the user.
9. Write a python program to perform basic matrix operations on user entered matrices.
10. Write a python program to perform basic operations of a calculator. Provide choice for operations to users and make a program iterative. Provide specific exit option to users.
11. Write a python program to demonstrate the use of arbitrary arguments.
12. Create a class named *student* having attributes - *std_name*, *std_age*, *std_branch*, and *std_city*. Create a method named *get_data*() in *student* class that takes user input for these attributes and a method named *display*() that prints the attribute values on the terminal. Call both the methods by creating an instance *std_obj* of the class *student*.

13. Write a python program to demonstrate basic banking operations. Create a class named banking having separate class methods for each operation. Call each method with an instance of the class and attribute values to be taken from the user.
14. Create a class named *employee* having attributes - *emp_name*, *emp_age*, and *emp_city*. Create a method named *get_data*() in *employee* class that takes user input for these attributes. Derive a class named *emp_derived*() from the *employee* class, having an *__init__()* method that displays the attributes of the *employee* class upon instantiation.
15. Create a base class named *university* with its attributes - *name, year_of_estd*, and *city*. Derive following class from the super class *university*: *professor*, *lab_assistant*, *office_assistant*, and *peon*. Make the program choice based for user. The attributes and method of various class are as below:
    - Attributes of *professor* class: *designation*, *highest_qualification*, *area_of_research, year_of_joining, year_of_ experience*, and *name_of_institute*.
    - Methods of professor class: *__init__*() method that gets invoked upon instantiation and takes values of class attributes. The *display*() method that prints class attribute values along with attributes of its super class.
    - Attributes of *lab_assistant* class: *designation* = "Lab Assistant" (static), *highest_qualification*, *additiobnal_skilss*, *year_of_joining*, and *name_of_institue*.
    - Methods of *lab_assistant* class: *__init__*() method that gets invoked upon instantiation and takes values of class attributes. The *display*() method that prints class attribute values along with attributes of its super class.
    - Attributes of *office_assistant* class: *designation* = "Office Assistant" (static), *highest_qualification*, *year_of_joining*, and *name_of_institute*.
    - Methods of *office_assistant* class: __init__() method that gets invoked upon instantiation and takes values of class attributes. The display() method that prints class attribute values along with attributes of its super class.
    - Attributes of *peon* class: *job_role* = "office Peon" (static), *qualification*, *year_of_joining*, and *name_of_institute*.
    - Methods of *peon* class: *__init__*() method that gets invoked upon instantiation and takes values of class attributes. The *display*() method that prints class attribute values along with attributes of its super class.
16. Create three classes named - *C, Python*, and *Web_Designing* each having two primary attributes as *learnings_* and *name_of_professor*. Derive a class named *student* from these classes. The student class hase following methods and attributes:
    a. Global *std_college* attribute with static values.
    b. *__init__*() method with attributes - *std_name*, *std_enrollment_no*, and *std_course*.
    c. *display*() method to display various attribute values of the terminal.
17. Write a python program to demonstrate the use of data hiding.