Python Assignments

NOTE: Apprentice should follow the criteria and complete the number of questions mentioned. There would be multiple questions for each topic, such as in Topic 1 (*args and **kwargs), there are 2 questions for each of *args and **kwargs topics. There is also tags mentioned [*args] and [**kwargs]. Apprentice should check the criteria and should complete the mentioned number of questions per tag.

1. *args and **kwargs

Criteria: Complete at least **one question** for each tag

- **[*args]** Write a Python function that takes an arbitrary number of positional arguments and returns the sum of all the numbers. Test your function with various input cases.
- **[*args]** Write a Python function concat_strings that takes any number of strings as arguments and returns a single concatenated string.
- **[**kwargs]** Write a Python function **calculate_total_cost** that calculates the total cost of items purchased from a store. The function should accept multiple keyword arguments, where the key is the item name, and the value is the item's price. The function should return the total cost of all items.
    - **Note:** You should use **kwargs
- **[**kwargs]** Create a function **create_student_report** that takes the student's name as the first argument, the student's age as the second argument, and an arbitrary number of keyword arguments for the subjects and their respective scores. The function should return a dictionary with the student's information and a list of subjects along with their scores.

2. Map, Filter, and Reduce

Criteria: Complete at least **one question** for each tag

- **[map]** Write a Python function **square_numbers** that takes a list of integers as input and uses the map function to return a new list containing the square of each element.

- **[map]** Create a function **convert_to_uppercase** that takes a list of strings as input and uses the map function to return a new list with all the strings converted to uppercase.

- **[filter]** Implement a function called **filter_prime_numbers** that takes a list of integers as input and uses the filter function to return a new list containing only the prime numbers.

- **[filter]** Write a Python function **filter_long_strings** that takes a list of strings as input and uses the filter function to return a new list containing strings with more than 5 characters.

- **[reduce]** Write a Python function **calculate_factorial** that takes an integer as input and uses the reduce function to return the factorial of that number.

- **[reduce]** Implement a function called **concatenate_strings** that takes a list of strings as input and uses the reduce function to return a single string containing the concatenation of all the elements.

3. Ternary Operators

Criteria: Complete at least **three questions**

- Write a Python function called **check_odd_even** that takes an integer as input and uses a ternary operator to return "Even" if the number is even, and "Odd" if the number is odd.

- Create a Python function **check_leap_year** that takes a year as input and uses a ternary operator to determine if it's a leap year. Return "Leap Year" if it is, otherwise "Not a Leap Year." (A leap year is divisible by 4, except for years divisible by 100 but not divisible by 400).

- Write a function **find_bigger_number** that takes three integers as input and uses a ternary operator to return the larger number. If all numbers are equal, return "Equal."

- Implement a function called **check_prime** that takes a positive integer as input and uses a ternary operator to determine if it's a prime number. Return "Prime" if it is, otherwise "Not Prime."

4. Collections

Criteria: Complete **all** the questions

- Given an array of strings (str), group the anagrams together. You can return the answer in any order.
  An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

```
Input: strs = ["eat","tea","tan","ate","nat","bat"]
Output: [["bat"],["nat","tan"],["ate","eat","tea"]]
```

5. Comprehension

Criteria: Complete at least **two question** for each tag

- **[list comprehension]** Given a list of strings, create a new list that contains only the strings with more than 5 characters using list comprehension.

- **[list comprehension]** Given two lists of integers, create a list that contains the product of each element of the first list with the corresponding element in the second list using list comprehension.

- **[list comprehension]** Given three lists list1, list2, and list3, each containing integers, write a Python program using list comprehension to generate a new list of unique triplets (x, y, z) where x is from list1, y is from list2, and z is from list3, such that x + y + z = 0.

- **[dictionary comprehension]** Given two lists - one containing keys and another containing values, create a dictionary using dictionary comprehension.

- **[dictionary comprehension]** Given a dictionary with students' names as keys and their respective scores as values, create a new dictionary that contains only the students who scored more than 80 using dictionary comprehension.

- **[dictionary comprehension]** Create a dictionary using dictionary comprehension to represent the ASCII values of lowercase alphabets (a-z) where the keys are the alphabets, and the values are their corresponding ASCII values.

    - Hint: use **ord()** function to get the ASCII value of each alphabet

- **[set comprehension]** Given a list of numbers, create a set using set comprehension that contains only unique even numbers.

- **[set comprehension]** Given a list of words, write a Python program to create a set using set comprehension that contains all the unique characters present in the words.

- **[set comprehension]** Given two strings, write a Python program to create a set using set comprehension that contains all the characters that are common in both strings.

6. Exception Handling

Criteria: Complete at least **4 questions**

- Write a Python program that takes two integers as input and performs division (**num1 / num2**). Handle the **ZeroDivisionError** and display a custom error message when the second number is zero.

- Implement a program that takes user input for a filename, opens the file in read mode, and displays its contents. Handle the **FileNotFoundError** and display an error message if the file is not found.

- Write a Python program that takes a user input and converts it to an integer. Handle the **ValueError** and display a custom error message when the input cannot be converted to an integer.

- Write a Python program that takes two integers as input and performs division (num1 / num2). Handle both **ValueError** (if non-integer input) and **ZeroDivisionError** and display appropriate error messages.

- Write a Python program that takes user input for age. Create a custom exception **InvalidAgeError** to handle cases where the age is below 0 or above 120.

    - **Hint:** Create new class **InvalidAgeError** that inherits **Exception** class

- Implement a program that reads user input for a password. Create a custom exception **WeakPasswordError** to handle cases where the password is shorter than 8 characters.

    - **Hint: WeakPasswordError** that inherits **Exception** class

7. File I/O

Criteria: Complete **all** the questions

- Implement a program that reads a CSV file named "data.csv," containing columns "Name" and "Age." Create a new CSV file called "adults.csv" with only the rows of individuals who are 18 years or older.

- Create a function **add_to_json** that takes a filename and a dictionary as input. The function should read the JSON data from the file, add the new dictionary to it, and write the updated data back to the same file.

    - Sample Json:

```
[
  {
    "name": "Ram",
    "age": 30
  },
  {
    "name": "Sita",
    "age": 25
  }
]
```

- Create a function **search_log** that takes a log file and a search keyword as input. The function should find and display all lines containing the search keyword.

8. Object Oriented Programming

Criteria: Complete **all** the questions

- Create a Python class to represent a University. The university should have attributes like name, location, and a list of departments. Implement **encapsulation** to protect the internal data of the University class. Create a Department class that **inherits** from the University class. The Department class should have attributes like department name, head of the department, and a list of courses offered. Implement **polymorphism** by defining a common method for both the University and Department classes to display their details.

- Build a Python class to represent a simple banking system. Create a class for a BankAccount, and another for Customer. The BankAccount class should have a constructor to initialize the account details (account number, balance, account type). The Customer class should have a **constructor** to set the customer's details (name, age, address) and create a BankAccount object for each customer. Implement a **destructor** for both classes to display a message when objects are destroyed.