Python Lab Assignment – 1

1. Write a prime generator program using only primes and using python loops.
2. Write a discount coupon code using dictionary in Python with different rate coupons for each day of the week.
3. Print first 10 odd and even numbers using iterators and compress. You can use duck typing.
4. Write a regular expression to validate a phone number.
5. Write first seven Fibinacci numbers using generator next function/ yield in python. Trace and memorize the function. Also check whether a user given number is Fibinacci or not.
6. Write a simple program which loops over a list of user data (tuples containing a username, email and age) and adds each user to a directory if the user is at least 16 years old. You do not need to store the age. Write a simple exception hierarchy which defines a different exception for each of these error conditions:

   - the username is not unique
   - the age is not a positive integer
   - the user is under 16
   - the email address is not valid (a simple check for a username, the @ symbol and a domain name is sufficient)

   Raise these exceptions in your program where appropriate. Whenever an exception occurs, your program should move onto the next set of data in the list. Print a different error message for each different kind of exception.
7. Write a function *findfiles* that recursively descends the directory tree for the specified directory and generates paths of all the files in the tree.
8. Create a list of all the numbers up to N=50 which are multiples of five using anonymous function.
9. Enumerate the sequence of all lowercase ASCII letters, starting from 1, using enumerate.
10. Write a code which yields all terms of the geometric progression a, aq, aq^2 , aq^3 , ...

    When the progression produces a term that is greater than 100,000, the generator stops (with a return statement). Compute total time and time within the loop.
11. Search for palindrome and unique words in a text using class method and string method.

12. Create a BankAccount class. Your class should support these methods: deposit, withdraw, get_balance, change_pin. Create one SavingsAccount class that behaves just like a BankAccount class, but also has an interest rate and a method that increases the balance by the appropriate amount of interest. Create another FeeSavingsAccount class that behaves just like a SavingsAccount, but also charges a fee every time you withdraw money. The fee should be set in the constructor and deducted before each withdrawal.

13. Write an operator overloading for len which shows string length for any given string and return only length of repetitive words with the text if the text has some repetitive parts. Determine the most frequently occurring words using most_common.

14. Implement a priority queue that sorts items by a given priority and always returns the item with the highest priority on each pop operation.

15. Make a list of the largest or smallest N items in a collection.

16. Create a dictionary that maps stock names to prices, which will keep insertion order.Find minimum price, maximum price and sort items according to their prices in first dictionary. Create another second stock dictionary. Find items that are only in first dictionary and find items whose prices do not match. Remove duplicate items from first dictionary. Sort both dictionaries for incrementing prices. Group items in first dictionary by price in multiple of 500. Find an item with price=800 from both dictionaries.

17. Write a function that flattens a nested dictionary structure like one obtain from Twitter and Facebook APIs or from some JSON file.

```
nested = {

        'fullname': 'Alessandra',

        'age': 41,

        'phone-numbers': ['+447421234567', '+447423456789'],

        'residence': {

                'address': {

                        'first-line': 'Alexandra Rd',

                        'second-line': '',

                        Testing, Profiling, and Dealing with Exceptions

                        [ 230 ]

                },

                'zip': 'N8 0PP',

                'city': 'London',

                'country': 'UK',

        },
```

}

18. Use parameterized or nose_parameterized to compute power of following values:

(2, 2, 4),

(2, 3, 8),

(1, 9, 1),

(0, 9, 0). Use pytest to check errors.

19. Use profile/cprofile to check pythogorian triples code in python. Think about time complexity of the code.

20. Write a python program to

   i. read lines from a file, break into tokens and convert the tokens to unique numerical values using python dictionary.

   ii. Convert lines of different lengths into lines of same length (maximum length). Use padding if and when required.

21. Write a python program to identify and extract numerical chunks from a text file and convert them into words; e.g.; 1992 → "*nineteen hundred and ninety two*".