Python Lab – Classes and Advanced Algorithms

You will need to use all the Python constructs you have learnt to date and any self-learning you have completed to implement to following exercises.

1. Create a *Rectangle* class with the following:

    a. __init__() - a constructor that takes as parameters the length and width of the rectangle
    b. perimeter() – a function that returns the rectangle perimeter with the following algorithm - 2*(width + length)
    c. area() – a function that returns the area of the rectangle with the following algorithm - width*length

2. Create a *BankAccount* class that supports the following methods

    a. __init__ - takes as input the bank account balance or 0 if no input is provided
    b. withdraw() – takes an amount as a parameter and subtracts from the balance
    c. deposit() – takes an amount as a parameter and adds to the balance
    d. get_balance() – returns the balance on the account

    Execute commands to:

    a. Display the bank account balance
    b. Deposit 5000 into the bank
    c. Withdraw 10,000 from the bank
    d. Display the bank account balance

3. Create a *Person* class with the following:

    a. __init__() - a constructor that that's as input the persons name and birth year.
    b. get_age() – returns the age of the person. You can hardcode 2014 as the current year and assume the Person is born in January of their birth year.
    c. get_name() – returns the name of the person
    d. get_details() - returns the name and age of the person

4. Create a subclass of Person (from exercise 3) called *Student*. The *Student* class will have an extra *student_number* property. Overwrite the *get_details()* method from Person so that it returns the student_number.

   The *Student* class will also have a *mature()* method that returns true or false to indicate whether they are a mature student (we define a mature student here as someone who is over the age of 23).

5. Write a function *negatives()* that takes a list *nums = [1, 2, -2, -3, 5, 4]* as input and *prints*, one per line, the negative values in the list. The function should not return anything.

6. Write an anonymous function (i.e., a lambda) that takes a positive integer *n* as a parameter and prints 2^n.

7. Write a function *even()* that takes a positive integer *n* as input and prints on screen all the numbers between 2 (inclusive) and *n* which are divisible by 2 or by 3. For example:

(calling)	even(17)
(prints)	2, 3, 4, 6, 8, 9, 10, 12, 14, 15, 16

8. Write a function *month()* that takes a number between 1 and 12 as input and returns a three character abbreviation of the corresponding month. For example:

(calling)	month(1)
(prints)	'Jan'

9. Every book has a unique ID known as an ISBN (e.g., 0-123-12345-0). Write a program to determine that a String is a valid ISBN:

   a. Check that there are 13 characters
   b. Of the first 12 characters, three are separators.

10. Write a program that asks the user to enter their name and student number. The program should then ask the user to enter the subjects they are studying and to indicate when they are finished by entering the word "done". The program should print all the details back to the user.

11. Write a function *intersect()* that takes two lists, each containing no duplicate values, and returns a list containing values that are present in both lists (i.e., the intersection of the two input lists)

(calling)	intersect ([3, 5, 1, 7, 9], [4, 2, 6, 3, 9])
(returns)	[3, 9]

12. Implement a function *pay()* that takes two arguments: an hourly wage and the number of hours an employee worked last week. Any hours worked beyond 40 is overtime and should be paid 1.5 times the regular hourly wage. Your function should compute and return the employees pay.

13. Implement a function *avg()* that takes as input a list that contains a set of number lists. Each number list represents the grades a particular student received for a course. For example, here is a list for a class of 4 students.

    [[95, 92, 86, 87], [66, 54], [89, 72, 100], [33, 0, 0]]

    The function avg() should print, 1 per line, each student's average grade. You may assume that every list of grades is nonempty but you may not assume that every student has the same number of grades.

14. Write a program that allows the user to convert from either degrees Celsius to Fahrenheit or from degrees Fahrenheit to Celsius. You should allow the user to select which conversion is being done and print out the appropriate conversion message.

°F to °C	Deduct 32, then multiply by 5, then divide by 9
°C to °F	Multiply by 9, then divide by 5, then add 32

    For example, 20 degrees Celsius is equal to 68 degrees Fahrenheit

15. Write a *read_file()* function that opens and reads all the contents of an external file. You should create a myfile.txt that contains 5 lines of text that your Python function reads and prints out each line of text. Ensure all possible exceptions are appropriately handled.