University of Huddersfield
School of Computing and Engineering

CFM2103 Mathematical Programming

Practical **Week 10**

Work through the questions included below. If you get stuck, please revisit the information on the slides from the previous weeks before you ask for help.

---

1. Write a class named `Person` with data attributes for a person's name, address, and telephone number. Next, write a class named `Customer` that is a subclass of the `Person` class. The `Customer` class should have a data attribute for a customer number, and a Boolean data attribute indicating whether the customer wishes to be on a mailing list. Demonstrate an instance of the `Customer` class in a simple program.

2. Write an `Employee` class that keeps data attributes for the following pieces of information:

   - employee name;
   - employee number.

   Next, write a class named `ProductionWorker` that is a subclass of the `Employee` class. The former class should keep data attributes for the following information:

   - shift number (an integer, such as 1, 2, or 3);
   - hourly pay rate.

   The workday is divided into two shifts: day and night. The shift attribute will hold an integer value representing the shift that the employee works. The day shift is shift 1, while the night shift is shift 2. Write appropriate accessor and mutator methods for each class.

   Once you have written the classes, write a program that creates an object of the `ProductionWorker` class and prompts the user to enter data for each of the object's data attribute. Store the data in the object, then use the object's accessor methods to retrieve it and display it on the screen.

3. Create a class called `BookstoreItem`. Every object instantiated from this class has a title, a description (i.e., action, cookery, DIY, etc), and a price. If you "print" an object from this class, the title of the item gets displayed on the screen. The aforementioned attributes should be *private* and your class must provide appropriate setter and getter methods.

   (a) Once you have implemented the above class, create a *subclass* called `Book`. Objects from this class will have two additional attributes: an author (or several authors), and a format (e.g., 'hardback' or 'paperback'). Make sure your constructor can deal with several authors for one book (i.e., you'll need to store such attributes in a list or a similar data structure).

   (b) Demonstrate the functionality of your classes by instantiating a couple or more objects and testing their various methods.

4. The Maclaurin expansion for the function $\tan^{-1}(x)$ is

$$\tan^{-1}(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \ldots .$$

Use *inheritance* in conjunction with the (super)class `Taylor` from the notes to provide your own implementation for the above function. Compare the output of your code with that of the built-in function $\tan^{-1}$ by displaying the appropriate values on the computer screen.

5. Use the class `Forward1` from the notes to calculate the second derivative of the function

$$f(x) = x^2 \sin(x) .$$

Display neatly on the screen the results of your code as well as the exact values for

$$x = 0.3 ,\ 0.7 ,\ 1.3 ,\ 2.6 .$$

6. A point in the plane can be represented by the class:

```
class Point(object):
    """
    Cartesian representation of a point in 2D
    """
    def __init__(self, x, y):
        self.x, self.y = x, y

    def __str__(self):
        return "({:.2f}, {:.2f})".format(self.x, self.y)
```

(a) Extend the `Point` class to also contain the representation of the point in polar coordinates. To this end, create a subclass `PolarPoint`, whose constructor takes the polar representation of a point, (r, θ), as arguments. Store r and θ as data attributes of this new subclass, and call the superclass constructor with the corresponding x and y values.

(b) Add a __str__ method in the class `PolarPoint`, which prints out r, θ, x, y (together with some suitable text message). Verify your implementation by initialising three points and printing them.

[There is a short handout on Brightspace that explains what polar coordinates are.]