COMP2002 - Artificial Intelligence

Week 1 - Python Exercises

Introduction

The aim of this sheet of exercises is to introduce Python coding for those of you who haven't done it before, and to serve as a refresher for those of you who have. You should complete the exercises ahead of the week 2 seminar session.

The model answers will be published shortly after, giving you enough time to re-attempt the exercises after the demonstration in the seminar. You should complete the exercises in a Jupyter Notebook. You can either install Jupyter on your own device or use the version available in the labs.

Activities

Your task is to go through the following tasks. Please note, you are expected to complete some work on this outside of the timetabled sessions.

Exercise 1 - Bubble Sort

Write a function called bubble that takes a list of values and performs the bubble sort algorithm in-place (meaning you don't return anything, the sort is performed on the variable you pass in). You can remind yourself about the bubble sort algorithm here.
Once you've implemented the function, call it by passing a list of values to assure yourself that it works.

Exercise 2 - Fibonacci Sequence

The Fibonacci sequence is a mathematical progression wherein each element is the sum of the previous two elements. Write a function `fibonacci` that takes an argument N and returns a list containing the first N Fibonacci numbers.

Exercise 3 - Numpy and Matplotlib

Generate a plot of the function $y = \sin(x)$ between the values $-\pi$ and π.

- Use the Numpy function np.linspace to generate 100 values in that region (the Numpy function np.linspace will also be useful).
- The Numpy function np.sin will compute the y value.
- The Matplotlib function plt.plot(x, y) can be used to plot the graph. You can also use plt.scatter(x, y) to get points rather than a line.
- See if you can find out how to change the colour of the line (for plt.plot) and dots (for plt.scatter).

Exercise 4 - More Numpy and Matplotlib

Generate 100 random numbers from the **uniform distribution** (np.random.rand) and 100 samples from the **normal distribution** (np.random.randn). Produce a boxplot showing the two distributions.

Extension task - Produce a histogram showing the two distributions.
You can find more information on the commands you'll need here:

- Numpy random numbers
- Matplotlib boxplot