Week 11: Optional Python revision session

Introduction

This revision session is **completely optional** and is intended for students who would like to consolidate their understanding of the Python material covered in Weeks 6-11. The session takes the place of my usual office hour in Week 11. If you would like to book an office hour appointment with me at a different time this week, please email me.

In this revision session, I will summarize the Python material, and work through the additional exercises shown below. There will also be an opportunity for you to ask questions.

Week 6

In the lecture, I explained how to identify different types of bugs in programs, how to minimize the chance of bugs appearing, and how to debug programs. In the lab, we looked at how to import, and use functions from, Python modules, how to generate random numbers in Python, and how to use familiar mathematical functions and constants in Python. Important concepts included how to generate random numbers uniformly from a given continuous or discrete range, how to store results of random experiments in a list, and the inexact nature of arithmetic using floats.

Additional exercises.

6.1 Identify and correct all syntax errors in the following code.

```
for i in range(10)
    Print("The next number is, i)
```

6.2 Identify the type of any bugs in the following code, and correct them.

```
a, b = 2, 3
print("The mean of a and b is" A+B/2)
```

6.3 Using `random.randrange()`, write a Python program that prints 10 random numbers drawn uniformly from the set 1, 100, 10000, 1000000.

Week 8

In the lab, we looked at how to use the string format method to tailor outputs, how to copy lists and strings, and how to define and use anonymous functions in Python. Important concepts included width and decimal space specification using `format()`, the use of `list()`, and the keyword **`lambda`**.

Additional exercises.

8.1 Write a Python program to print $\sqrt{2}$ to an increasing odd number of decimal places from 1 to 9, on consecutive lines, with each number aligned centrally.

8.2 Write a Python function that takes as arguments an arithmetic function `f` and a list of floats `l`, and returns `l` unchanged along with a new list that is the elementwise operaton of `f` on `l`.

8.3 Using an anonymous function, test your Python function from additional exercise 8.2 by squaring the list `[1.0, 2.0, 3.0, 4.0]`.

Week 9

In the lab, we looked at how to calculate integrals numerically using different approximations, how to use local variables in functions, and how to define recursive functions. Important concepts included passing functions as arguments to other functions, the shorthand notation `+=` and `*=`, the use of **`return`** statements in functions, and the use of recursive functions in calculating factorials.

Additional exercises.

9.1 What will the output of the following be? Type it in and check.

```
1  x = 1
2  def fun1():
3      x = 2
4      def fun2():
5          print(x)
6      fun2()
7  fun1()
8  print(x)
```

Will the output change if lines 4–6 are unindented? How?

9.2 Write a recursive Python function to calculate the nth Fibonacci number. Use your function to calculate the tenth Fibonacci number.

Week 10

In the lab, we looked at how to use the Matplotlib module to plot graphs, and how to create and edit Numpy arrays. Important concepts included the Matplotlib functions `plot()`, `show()`, `set_aspect()`, and the Numpy functions `array()` and `linspace()`.

Additional exercises.

10.1 Using Matplotlib and Numpy arrays, write a program to plot, in green, the graph of $\tan^{-1}(x)$ for $x \in [-10, 10]$. Remember to label your axes.

Week 11

In the lab, we looked at how to draw and save parametrically defined curves and scatter plots, and how to define graphical elements using functions. Important concepts included the Matplotlib functions `axis()`, `savefig()`, and `scatter()`, and use of for loops to vary parameters when plotting.

Additional exercises.

11.1 Write a Python program that plots 100 points (x, y) drawn uniformly at random from $[0, 1] \times [0, 1]$ and plots these points, colouring them green if they lie inside the unit circle $x^2 + y^2 < 1$ and red otherwise, and prints the proportion of points that are green. What mathematical constant does this proportion approximate?