Week 6: Optional Python revision session

Introduction

This revision session is **completely optional** and is intended for students who are still getting to grips with Python. The session takes the place of my usual office hour in Week 6. 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 covered so far, and work through the additional exercises shown below. There will also be an opportunity for you to ask questions.

Week 1

In the lab, we learned about Google Colab and looked at how to carry out simple arithmetic in Python. In the lecture, I explained why we use Python in this course, what a computer program is, and how to calculate a factorial in Python using a while loop.

Week 2

In the lab, we looked at how to define, assign, and print variables, and how to write and edit basic Python programs. Important concepts included assignment versus equality, parallel assignment, and spacing/naming conventions.

Additional exercises.

2.1 Calculate the numerical value of the golden ratio, $\phi = (1 + \sqrt{5})/2$.

2.2 Are the following appropriate variable names in Python? Why / why not?

(a) `1st_name`
(b) `prime_factor`
(c) `GoldenRatio`
(d) `lambda`
(e) `_user_`

2.3 Use parallel assignment and a while loop to print the first ten terms of the sequence that starts $x_0 = 0$, $y_0 = 0$ and for $n \geq 0$ satisfies

$$x_{n+1} = \sqrt{\frac{x_n^2 + y_n^2}{5}}, \quad y_{n+1} = \frac{x_n + y_n + 1}{2}.$$

What happens to x_n and y_n as n increases?

Week 3

In the lab, we looked at how to interrogate and convert between Python variable types, construct Boolean expressions using comparison operators, and write while loops to perform repeated commands. Important concepts included `int()`, `//` and `%`, comparison operators, and incrementing in while loops. In the lecture, I explained the standard elements of a programming language, the process of writing a computer program, and how to use this process to find factors by trial division.

Additional exercises.

3.1 What is the value of `int(-7.3)`?

3.2 How do you write the following conditions in Python?
   (a) $x \leq y$
   (b) a is not equal to b
   (c) $0 < t \leq 10$
   (d) $z \in \{1, 2\}$

3.3 By modifying your while loop in exercise 1.3, find how many terms in the sequence there are before x_n exceeds 0.99.

Week 4

In the lab, we looked at how to write if and else statements to perform conditional operations, how to interpret and resolve type errors, and how to define, interrogate, concatenate, and print strings. Important concepts included type errors, string indices, and print separation.

Additional exercises.

4.1 Write a program that takes in a string, `s`. Your program should print the length of the string if the penultimate character of `s` is "a" or "b"; otherwise, your program should print the entire string twice, with no separation. (You can assume that `s` has at least two characters.)

Week 5

In the lab, we looked at how to write for loops, how to create, modify, iterate over, and pick slices of lists, and how to test if something is a member of a list. Important concepts included the `range()` command, list methods, the `in` command, and naming conventions for constants. In the lecture, I explained how to refactor Python code using for loops, how to improve the performance of code, and how write instructions for producing Pascal's triangle.

Additional exercises.

5.1 Write a program that takes in a list, `l`. Your program should print the index of the first instance of the number 4 in `l` if it is in `l`; otherwise, your program should add the number 4 to the end of `l` then print the last three entries `l`. (How does your code behave if `l` contains less than three entries?)

Week 6

In the lab, we looked at how to define and call Python functions, how to use functions to make the structure of a program simpler, and how to use in-built Python functions and methods. Important concepts included passing arguments to functions, returning values, and methods versus functions. 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.

Additional exercises.

6.1 Write a function that takes in a sentence as a string, `s`, and uses the `split()` and `count()` methods to print out the number of times the word 'and' features in the sentence. (Here we don't care about capitalisation.)