CSB110MC – Python Programming

Lauren Powell

The lab classes consist of tasks that are designed to be completed during the lab class. If you do not complete the tasks during the lab class then you should do them at home and have them ready to be checked in the next lab class.

Each task should be stored in its own `.py` file.

Do not use the interpreted mode for doing the tasks (only use the interpreted mode for experimentation). If you store your python code as `.py` files then you can go back and look at them at a later date.

CSB110MC Lab Class 4 — Thursday 16/03/2023

☐ Task 4.1

Write a function which has as a parameter the radius of a circle. The function should compute and **return** the area of a circle with that radius. Note: the area of a circle with radius r is calculated as follows:

$$\text{Area} = \pi \times r^2$$

Call your function (several times) from a main function to test that it works correctly. Don't forget to call the main function at the bottom of your source code.

Hint: The `math` library allows you to import the constant `pi`.

☐ Task 4.2

Write a function to classify a student's mark as to whether it is a first class mark, a 2(i) mark, a 2(ii) mark, a third class mark or a fail. The function should take as an argument a mark (a float between 0.0 and 100.0) and **return** a string according to the table below.

Mark	Classification
$mark < 40$	Fail
$40 \leqslant mark < 50$	Third
$50 \leqslant mark < 60$	2(ii)
$60 \leqslant mark < 70$	2(i)
$70 \leqslant mark$	First

Write a main function which prompts and allow the user to enter a student's mark as a floating-point number. You should call your classification function with this and print the output in a suitable message.

☐ Task 4.3

Write a function which has as a parameter a list of integers. The function should compute and **return** the maximum (i.e., largest integer) of the list.

Do not use the built-in `max` function for this task.

☐ Task 4.4

Write a function which is capable of converting to and from many types. The function should take as parameters:

- A string representing the source unit, e.g., "cm".
- A string representing the target unit, e.g., "in".
- A float of the value to be converted from the source unit to the target unit.

Your function should be capable of converting the following units:

- Length units:
  - Centimetres (cm)
  - Inches (in)
  - Meters (m)
- Weight units:
  - Kilograms (kg)
  - Pounds (lbs)
  - Stones (st)

Your program should be capable of converting any length unit (listed above) to any other length unit (listed above), and similar for weight units. If your function is asked to convert from a length unit to a weight unit or vice versa then the function should return -1 to indicate an error.

Hint: Break this program down into to sub-functions – one to deal with conversion of lengths and one to deal with conversion of weight. Build these two functions **first**, test them and be sure they work. Then once you are sure they work correctly, you can create a new function (the super conversion function) which simply calls the two functions.

☐ Task 4.5

The Fibonacci numbers are the numbers in the following sequence:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$$

The first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

Write a function which takes an integer n as an argument and returns the n-th Fibonacci number.

Hint: Try thinking recursively!

☐ Task 4.6

Create a **recursive** function that has as parameters a float x and an integer `n`. The function should **return** `x**n`.

Hint: How can you compute `x**n` from `x**(n-1)`?

☐ Task 4.7

This task involves putting reusable blocks of code for your Nim program into functions.

Create a function that has as a parameter a list of integers, representing the number of tokens in an arbitrary number of piles. The function should print a representation of the piles to the screen.

Hint: This function should not **return** a value

Create a function that has as a parameter a list of integers, representing the token piles. The function should **return** the list with a random number of tokens removed from a randomly selected pile.

Hint: One (or more) of the piles may be empty. You should remove tokens from a nonempty pile only.

Create a function that has as a parameter a list of integers, representing the token piles. The function should prompt the user to select a pile and a number of tokens to remove from that pile, then **return** the resulting list.

Hint: The user should not be allowed to select an empty pile.