Student ID: 730091091
ECM1421DA Exam

1.

a.

```
def long_name(list_of_names):
    """Reads a non-empty list of names stored as strings and returns the first one
with a length of more than 35 characters.
    If none is found, then return the string ‘long name not found’."""
    # Iterating through names
    for name in list_of_names:
        # Checking if length is > 35
        if len(name) > 35:
            return name
    # When there are no long names
    return 'long name not found'
```

b.

```
import math

def overall_mark(marks):
    '''Takes the first, second and third assessment marks as a list and returns the
overall mark for the module.'''
    # Calculating the weighted marks
    weighted_marks = [0, 0, 0]
    weightings = [0.3, 0.3, 0.4]
    for x in range (3):
        weighted_marks[x] = math.ceil(marks[x] * weightings[x])
    # Returning overall mark
    return weighted_marks[0] + weighted_marks[1] + weighted_marks[2]
```

c. Python indexing starts from 0, but the code is iterating through indexes 1-4. This method will skip the first item in the list (index 0). This can be corrected by changing the 1 to a 0.

d. There is no validation or error handling, meaning an invalid input would break the program. Validation can be implemented by checking if the lengths of the two lists are the same before the for loop. Additionally, a try and except block can be used, placing the print statement into the try block and printing an error message within the except. This would ensure the program does not crash if there is a problem with inputs of only some of the averages, and that the other averages can still be calculated.

e. continue and break have been used incorrectly. continue should be used to end the current iteration and continue to the next iteration, while break should be used to exit out of the while loop. You would want to break when a valid stock code is entered and continue if an invalid stock code is entered, not the other way around.

2.

Student ID: 730091091
ECM1421DA Exam

a.

i. Assert statements are used to test parts of code. If the condition of the assert statement is true, the program continues. If it is false, however, the program returns an assertion error. In this scenario, the first assert statement checks to see if passing the parameters 1, 2 and 3 into add_3_numbers will return 6, and the second assert statement checks if passing 1, 2 and -3 returns 6.

ii. When the script is run, it will first execute the first assert statement. Since 1+2+3=6, the condition will be true and the program will continue. The second assert statement is then executed. 1+2+(-3) does not equal 6, so the condition is false. An assertion error will be outputted to the console and the program will halt.

iii. This line of code makes sure that the statements within it are only executed if the Python script is run directly. If the script is imported as a module by another program, then the lines within this if statement will not be executed. This prevents unintended execution of these lines of code.

iv. var_a, var_b and var_c are all assigned outside of the function, so these variables have a global scope. The values of these variables are then passed into add_3_nbrs as parameters and are assigned to the local variables nbr1, nbr2 and nbr3. So at this point. Within the function, a new local variable, var_c, is assigned the value -3. This means there are now two variables named var_c, one global and one local. The local variable nbr3 is assigned the value 3, which is the same as before so no change. Finally, the function returns var_a + var_b. + var_c. Since there is both a global and local variable named var_c, the local variable takes precedence. Therefore, the function returns 1 + 2 + (-3), which is 0. This results in the first assert statement passing, but the second assert statement failing, raising an assertion error.

v. Test runners allow testers/developers to run multiple tests easily. Any failed tests are handled and do not cause a crash. Most test runners provide a short report of test results. Tests can be run from the command line and therefore automated.

vi. A framework is a collection of code libraries that provide a foundation and standard way of doing something. It is not a single program but rather a set of reusable components that work together. The libraries establish a structure and common practice for achieving a specific goal.

vii. The as keywords establishes an alias for the pytest keyword that can be used throughout the rest of the program. In this case, instead of typing pytest.somefunction(), you can use pyt.somefunction(), which can save time when developing and make the code more readable.
viii. Using the as keyword allows the developer to change a keyword such as pytest to something shorter and easier to type, such as pyt. This can reduce the number of spelling mistakes the developer makes and therefore reduces the number of syntax errors. Using the as keyword can also help reduce the number of naming conflicts.

3.

a. Tkinter is a standard Python framework that allows you to create a GUI (Graphical User Interface). Tkinter works by providing a set of tools and widgets that are used to build a user interface. A Tkinter app would act a a thick client, with the Tkinter widgets having to be installed and maintained on the user's workstation. This implementation also uses the user's workstation resources to run the GUI. In comparison, Flask is an open-source framework used for building web applications. Flash allows us to use Python as a server-side language. The GUI built with HTML and CSS, as well as Python and Javascript, can be run on the server side, allowing the user to access the application on a thin client. Little to no installation is required on the client workstation, and the client can use a web browser to access the application.
b. GUIs are mostly static. They work by displaying widgets on the screen, such as images, text, buttons and text input fields. The program waits for the user to provide an input, such as a click of a button or an input of text into a text field before 'doing something'. This is in practice the same as the event loop in event-driven programming. Most of the time is spent waiting for an 'event', which is the same as these interactions with the GUI. Once an event takes place, then code is run by the event handler.
c. The first benefit is cost. Cloud platforms offer a pay-as-you-go model, removing the need for upfront investment in expensive hardware and software licenses. You only pay for the resources you use, reducing costs. The second benefit is scalability. Cloud databases can easily scale up or down based on your changing needs. During periods of high demand, you can quickly add resources to handle the extra load, and during slower times, you can scale down to avoid unnecessary costs. The final benefit is security and reliability. Cloud providers offer robust security features and disaster recovery solutions that are often beyond the reach of in-house IT teams. Data backups are automated and geographically distributed,

minimising the risk of data loss due to hardware failure or natural disasters.

d. ODBC is a standard API (Application Programming Interface) for accessing databases. PYODBC is the Python specific driver for the ODBC API. Importing PYODC will allow the Python program to connect to the cloud-based SQL Server database by simply providing a connection string. Once connected, SQL statements can be executed on the database remotely from Python, allowing the program to retrieve, insert, update or delete data as well as create, alter, execute and delete objects such as tables, views and stored procedures.

e.