HOMEWORK WEEK 4
(handout for students)

TASK 1 (Git and GitHub)

Question 1

Complete definitions for key Git & GitHub terminology

GIT WORKFLOW FUNDAMENTALS

· **Working Directory –** the current working directory is the folder which your script is operating and you can see your files when they are checked out. It is also known as the file path on your computer that sets the default location of any files you read.

· **Staging Area –** it is the area which maintains files that will be part of the next commit. So once you have chosen the files you want to commit it lets git know what changes in the file will occur for the next commit.

· **Local Repo (head) –** it is normally the git repository found on your computer as it is responsible for making the local changes. Once the code is committed a version or snapshot is created in your local repo.

· **Remote repo (master) –** remote repositories are versions of your project that are hosted on a code hosting service such as GitHub or an internal server on the internet or network. When you collaborate with others this involves you to manage the remote repositories by pushing and pulling data to and from where you need to share work.

WORKING DIRECTORY STATES:

· **Staged –** Before committing a file we need to stage it first. A staged file is added to the index which can be later used to record these changes in small commits.

· **Modified –** When you edit the files this is what Git calls modified as they have been changed since the last time you committed them. After you complete your work you are able to choose which modified files you want to stage and then commit all the staged changes.

- **Committed –** These are the files that you have chosen to commit and if you would like to see which files these were you can perform a git log which will give you a list of all the commits made to your repository.

GIT COMMANDS:

- **Git add –** once the files have been added this puts the files in a staged area. Therefore this command helps to stage changes to the project that will be stored in a commit.

- **Git commit –** captures a committed snapshot of the project's currently staged changes which are also known as safe versions of the project that will not be changed by Git unless asked.

- **Git push –** this is used to upload files from the local repository to the remote repository after committing them. One the file has been modified the git push allows you to share your work with your other team members that have access to your remote repo.

- **Git fetch –** this is used to download the commits and files from the remote repository to your local repository but it will not update your local repo's working state. This command is used when you want to see the other files that people have been working on.

- **Git merge –** if the changes on one branch have been approved you can use this command to merge it into the local branch so the current branch will be updated to reflect the merge. Merging allows git to put forked history back together and this command takes the multiple sequences of commits which can be formed by different git branches and integrates them into a single branch. Git merge is often used with git checkout to helps you to navigate between the branches.

- **Git pull –** this command is used to update your current HEAD branch with any recent changes from the remote server. This command is used to download the remote data for the active local branch and to integrate it into your current working copy of the files where you create a merge commit. It is better to use a git pull when local changes have been committed first hence it is a clean working copy.

TASK 2 (Exception Handling)

Question 1

Simple ATM program

Using exception handling code blocks such as try/ except / else / finally (NB: the more the better, but try to use at least two key words e.g. try/except) write a program that simulates and ATM machine to withdraw money.

Tasks:

1. Prompt user for a pin code

2. If the pin code is correct then proceed to the next step, otherwise ask a user to type in a password again. You can give a user a maximum of 3 attempts and then exit a program.

3. Set account balance to 100.

4. Now we need to simulate cash withdrawal

5. Accept the withdrawal amount

6. Subtract the amount from the account balance and display the remaining balance (NOTE! The balance cannot be negative!)

7. However, when a user asks to 'withdraw' more money than they have on their account, then you need to raise an error an exit the program.

TASK 3 (Testing)

Question 1

Use the Simple ATM program to write unit tests for your functions.

You are allowed to re-factor your function to 'untangle' some logic into smaller blocks of code to make it easier to write tests.

Try to write at least 5 unit tests in total covering various cases.