**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 local working directory is the folder where the hidden. git folder is located. It is the project folder where you run the git init command or the folder you cloned from GitHub or Git Lab. Only the files inside the project folder can be tracked by git.**

· **Staging Area- The staging area is a place to record things before committing.** **It's the area between the working directory and the. git directory. All the files which are ready for a commit are stored here.**

· **Local Repo (head)-** **The local repository is a Git repository that is stored on your computer.**

· **Remote repo (master)-** **The remote repository is a Git repository that is stored on some remote computer. The remote repository is usually used by teams as a central repository into which everyone pushes the changes from his local repository and from which everyone pulls changes to his local repository.**

WORKING DIRECTORY STATES:

· **Staged -** **means that modified file has been market in its current version to go into the next commit snapshot.**

· **Modified - When any change to the file occurs, the state of the file changes from committed to modified. This means that the document has changed since its last committed version which is saved to our local database. We can see this state as 'we're currently working on this file, there will still be more changes'.**

· **Committed - This state indicates that the file is safely stored in the local database.**

GIT COMMANDS:

· **Git add -** **This command adds a file to the staging area. - Usage: git add [file]**

· **Git commit -** **This command records or snapshots the file permanently in the version history. -** **Usage: git commit -m “[ Type in the commit message]”**

· **Git push-** **This command sends the committed changes of master branch to your remote repository. -** **Usage: git push [variable name] master**

· **Git fetch-** **only downloads new data from a remote repository - but it doesn't integrate any of this new data into your working files. Fetch is great for getting a fresh view on all the things that happened in a remote repository. Due to its "harmless" nature, you can rest assured: fetch will never manipulate, destroy, or screw up anything. This means you can never fetch often enough. -** **git fetch origin**

**· Git pull- in contrast to git fetch, is used with a different goal in mind: to update your current HEAD branch with the latest changes from the remote server. This means that pull not only downloads new data; it also directly integrates it into your current working copy files. -** **Usage: git pull [Repository Link]**

· **Git merge-** **This command merges the specified branch’s history into the current branch. -** **Usage: git remote add [variable name] [Remote Server Link]**

**TASK 2 (Exception Handling)**

**Question 1**

**Simple ATM program**

Using exception handling code blocks such as try/ except / else / finally, write a program that simulates an ATM machine to withdraw money.

(NB: the more code blocks the better, but try to use at least two key words e.g. try/except)

**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.

**Graphical user interface, text, application, email

Description automatically generated**

**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.