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**

  Your workspace that you are using for the project, it is simply your current local directory that you are working on. The project folder will be saved locally on your computer.

- **Staging Area**

  Staging area the in-between area where the files go that are going to be a part of the next commit, which lets git know what changes in the file are going to occur for the next commit.

- **Local Repo (head)**

  The local repo is is a ref that points to the tip (latest commit) of a branch. It is the commit or branch you are viewing in a code repository. Local repositories reside on the computers of individuals/ team members.

- **Remote repo (master)**

  The remote repo is remote repositories that are hosted on a server that is accessible for all team members - most likely on the internet or on a local network. Github is a remote repo. It's purpose is to publish your code to the world (or to some people) and allow them to read or write it. The remote repository is only involved when you git push your local commits to a remote repository, or when you git pull someone else's commits from it.

  Your Github account repositiory:

https://github.com/CMcG2020

WORKING DIRECTORY STATES:

- **Staged -** at this stage you have marked a changed file in it's current version as ready to go onto the next commit. It is where the file is 'prepared' for it's commit. You can continue to make changes to the working directory.

- **Modified -** means you have changed the file but not yet committed the changes yet

- **Committed -** the file/ data has been safely stored in your local database. Once you do a commit it takes the files in the staging area and stores it in your git directory.

GIT COMMANDS:

- **Git add**

  git add [file] - used to add the file to the staging area. This moves changes from the working directory to the staging area.

- **Git commit**

  git commit -m "[Add a commit message]" It will take the staged snapshot and commits it to the version history

- **Git push**

  git push [variable name] master - this sends the committed changes of master branch to your remote repo.

- **Git fetch**

  It downloads a branch from another repository, along with all of its connected commits and files. It doesn't try to integrate anything into your local repo, you can inspects changes before merging to your project.

- **Git merge**

  git merge [branch name] - merges a branch into the current branch. After forking the project history with git branch, git merge lets you put it back together again.

- **Git pull**

  git pull [Repository Link] - gets and merges changes from the remote to your working directory.

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.

```
#Check for valid pin.
def verify_pin(pin):
    return pin == 1111

#Give user three attempts to try pin
def log_in():
    tries = 0
    while tries < 3:
        pin =int(input('Please Enter Your 4 Digit Pin: '))
        if verify_pin(pin):
            print("Pin accepted!")
            return True
        else:
            print("Invalid pin")
            tries += 1
    print("To many incorrect tries. Could not log in")
    return False
```

```
def check_input(withdraw):
    try:
        withdraw = int(withdraw) # check if input is a number
        try:
            if withdraw <= 0: # check if withdraw is negative
                raise ValueError("Please enter a valid amount. ")
        except ValueError as ve:
            print (ve)
            return False
    except ValueError:
        print("Invalid input")
        return False
    else:
        return True
```

```
# transaction function to withdraw money
def transaction(withdraw):
    balance = 100 ##set account balance to 100
    try:
        x = balance - withdraw
        if x < 0: # check if balance is negative
            raise ValueError("Insufficient fund!")
    except ValueError as ve:
```

```
        print(ve)
    else:
        balance -= withdraw  # subtract withdraw from balance
        print(f"Your balance is: £{balance}.")
    return balance
def main():
    if log_in():
        withdraw = (input('please enter withdrawal amount: '))
        if check_input(withdraw):
            transaction(int(withdraw))
```

```
main()
```

Output:

```
Please Enter Your 4 Digit Pin: 1111
Pin accepted!
please enter withdrawal amount: 0
Please enter a valid amount.
PS C:\Users\claire\Desktop\CFGdegree\homework> & C:/Python310/python.exe c:/Users/claire/Desktop/CFGdegree/homework/hw_4.py
Please Enter Your 4 Digit Pin: 1111
Pin accepted!
please enter withdrawal amount: -9
Please enter a valid amount.
PS C:\Users\claire\Desktop\CFGdegree\homework>
```

```
PS C:\Users\claire\Desktop\CFGdegree\homework> & C:/P
Please Enter Your 4 Digit Pin: 1234
Invalid pin
Please Enter Your 4 Digit Pin: 1222
Invalid pin
Please Enter Your 4 Digit Pin: 6712
Invalid pin
To many incorrect tries. Could not log in
PS C:\Users\claire\Desktop\CFGdegree\homework>
```

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.

```
from unittest import TestCase, mock
import unittest

from hw_4 import verify_pin, check_input, transaction
#import locally functions to test
```

```
class ATM_test(unittest.TestCase):
```

```
#1.test withdrawl works
    def test_correct_withdrawal(self):
        result = transaction(withdraw=20)
        self.assertEqual(result, 80)
```

```
#2.test if withdrawal doesn't not work
    def test_incorrect_withdrawal(self):
        result = transaction(withdraw=350)
        self.assertEqual(result, 100)
```

```
#3.check correct pin is recognised
    def test_correct_pin(self):
        a = verify_pin(pin=1111)
        self.assertTrue(a)
```

```
#4.test if incorrect pin is recognised
    def test_incorrect_pin(self):
        a = verify_pin(pin=1234)
        self.assertEqual(a, False)
```

```
#5.test if input is a number
    def test_input_is_number(self):
        a = check_input(withdraw="a")
        self.assertEqual(a, False)
```

```
#6.test if input is a negative number
    def test_input_is_negative(self):
        a = check_input(withdraw=-20)
        self.assertEqual(a, False)
```

```python
#7.test if input is a positive number
    def test_input_is_positive(self):
        a = check_input(withdraw=20)
        self.assertEqual(a, True)

#8.test if input is a zero
    def test_input_is_zero(self):
        a = check_input(withdraw=0)
        self.assertEqual(a, False)
```

```python
# -- unit test --
```

```python
if __name__ == '__main__':
    unittest.main(verbosity= 2)
```

```
PS C:\Users\claire\Desktop\CFGdegree\homework> & C:/Python310/python.exe c:/Us
Please Enter Your 4 Digit Pin: 1111
Pin accepted!
please enter withdrawal amount: 10
Your balance is: £90.
test_correct_pin (__main__.ATM_test) ... ok
test_correct_withdrawal (__main__.ATM_test) ... Your balance is: £80.
ok
test_incorrect_pin (__main__.ATM_test) ... ok
test_incorrect_withdrawal (__main__.ATM_test) ... Insufficient fund!
ok
test_input_is_negative (__main__.ATM_test) ... Please enter a valid amount.
ok
test_input_is_number (__main__.ATM_test) ... Invalid input
ok
test_input_is_positive (__main__.ATM_test) ... ok
test_input_is_zero (__main__.ATM_test) ... Please enter a valid amount.
ok

----------------------------------------------------------------------
Ran 8 tests in 0.002s

OK
PS C:\Users\claire\Desktop\CFGdegree\homework>
```