**HOMEWORK WEEK 4**

Madina Baimaganbetova

**TASK 1 (Git and GitHub)**

**Question 1**

Complete definitions for key Git & GitHub terminology

GIT WORKFLOW FUNDAMENTALS

· **Working Directory** – is a folder where code is currently running

· **Staging Area –** some sort of space where you add new versions of the files for the next commit

· **Local Repo (head) –** is a repository stored on your local computer checked out at the latest local commit

· **Remote repo (master) –** is a repository stored outside of your computer somewhere on the internet (GitHub for ex) checked out at the master branch

WORKING DIRECTORY STATES:

· **Staged –** files that we are tracing for commit. After git add before git commit

· **Modified –** files that have been changed, created or deleted

· **Committed –** files that are committed to local history

GIT COMMANDS:

· **Git add-** add a file to staging area so that they can be later committed to your next commit (stage)

· **Git commit -** commit your staged content, add it to the history of your local repository

· **Git push -** send local branch commits to the remote repository branch

· **Git fetch -** fetch down all the branches from remote repository branch

· **Git merge -** merge the specified branch history into the current one

· **Git pull –** fetch all updates for local branches and then merge any commits from the tracking remote branch (some sort of fetch and merge in one)

**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.

**Answer:** Please also find working .py file

#withdraw money

def withdraw(with\_amount, acc\_balance):

    try:

        with\_amount = float(with\_amount)

        if (acc\_balance < with\_amount) or (with\_amount < 0):

            raise Exception

        else:

            acc\_balance -= with\_amount

    except Exception:

        print('Failed withdrawal')

    finally:

        return acc\_balance

#validate pin

def pin\_valid(pin\_input, pin\_correct):

    return pin\_input == pin\_correct

def run(pin\_correct, acc\_balance):

    trial = 0

    while trial < 3: #three attempts for correct PIN

        pin\_input = input('Input your PIN: ')

        if pin\_valid(pin\_input, pin\_correct):

            print(f'Your balance is {acc\_balance}')

            with\_amount = input('Please enter withdrawal amount: ')

            acc\_balance = withdraw(with\_amount, acc\_balance)

            print(f'Your balance is {acc\_balance}')

            break

        else:

            print('Incorrect PIN, Please try again ')

            trial += 1

    if trial >= 3:

        print('Maximum number of attempts was reached! Good bye! ')

if \_\_name\_\_ == '\_\_main\_\_':

    run(pin\_correct='0000', acc\_balance=100)

**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.

**Answer:** You can also find working .py file

from unittest import TestCase, main

from ATM\_task2 import pin\_valid, withdraw

class TestWithdrawFunction(TestCase):

    def test\_withdraw\_less\_than\_balance(self):

        expected = 40.0

        result = withdraw(with\_amount='60', acc\_balance=100)

        self.assertEqual(expected, result)

    def test\_withdraw\_more\_than\_balance(self):

        expected = 100.0

        result = withdraw(with\_amount='120', acc\_balance=100)

        self.assertEqual(expected, result)

    def test\_withdraw\_negative\_amount(self):

        expected = 100.0

        result = withdraw(with\_amount='-100', acc\_balance=100)

        self.assertEqual(expected, result)

    def test\_withdraw\_not\_numerical\_value(self):

        expected = 100.0

        result = withdraw(with\_amount='hello', acc\_balance=100)

        self.assertEqual(expected, result)

class TestPinValidFunction(TestCase):

    def test\_pin\_valid(self):

        expected = True

        result = pin\_valid(pin\_input='0000', pin\_correct='0000')

        self.assertEqual(expected, result)

    def test\_pin\_invalid(self):

        expected = False

        result = pin\_valid(pin\_input='1111', pin\_correct='0000')

        self.assertEqual(expected, result)

if \_\_name\_\_ == '\_\_main\_\_':

    main()