

Times Tables Challenge Terminal Application

By Nicole Hulett

T1_A3

Purpose and Target Audience

- This is an application designed for students of all ages.
- Educational purpose – to assist users to improve their multiplication skills
- Entertainment purpose – It's a game, so users can play for fun

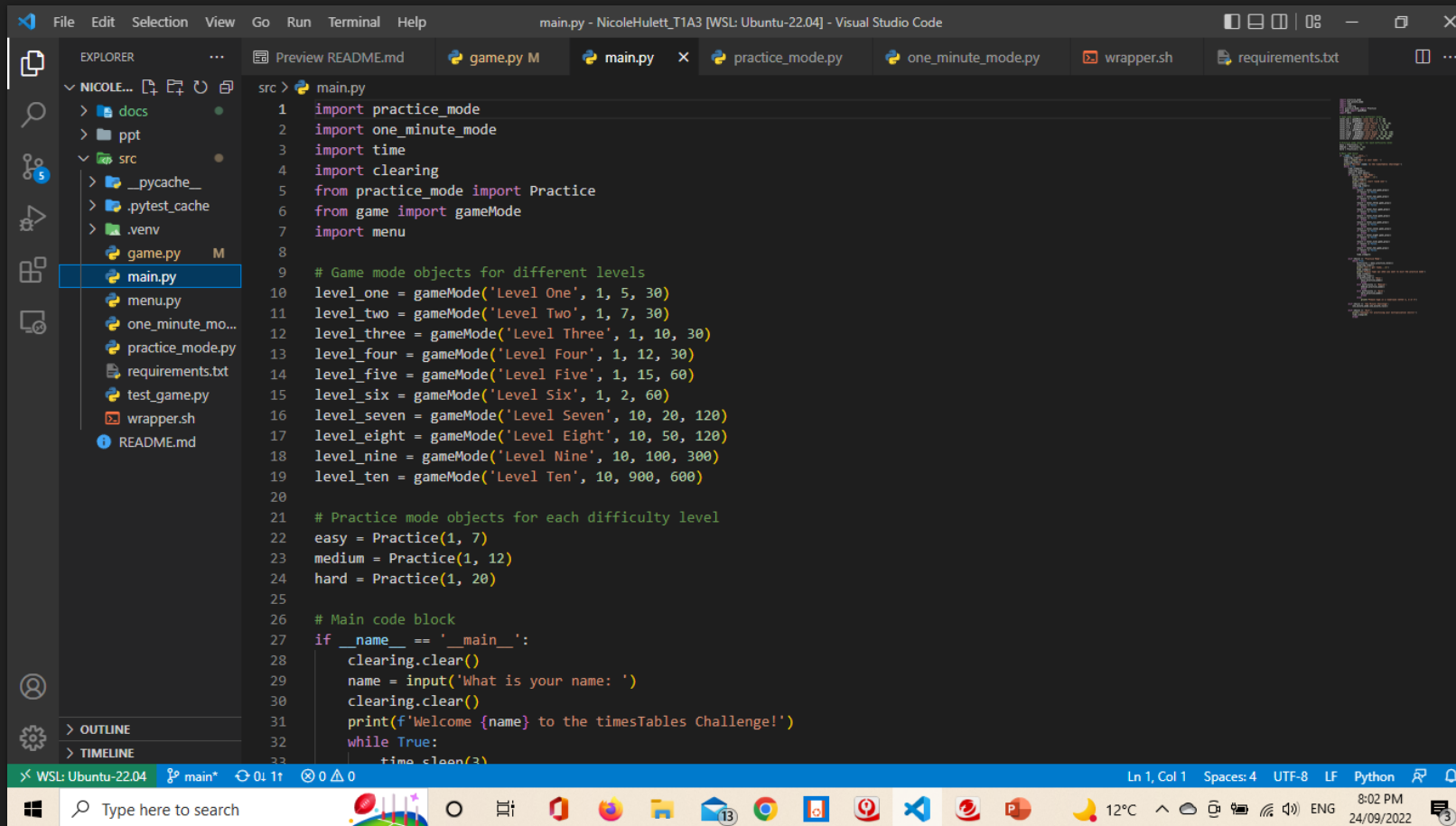
Main Features

- 3 main features:
 - game mode
 - practice mode
 - one minute challenge mode
- Smaller features:
 - The use of a timer in most modes
 - A welcome screen that welcomes the user by name
 - Terminal menus for selections

Structure of App

- User enters name that then leads to a personal welcome message being displayed.
- The main menu has a list of the different modes the user can choose by scrolling and pressing enter.
- The item chosen then calls the appropriate function to execute that mode.
- The code for all the different modes are in their own individual module.
- On ending a mode, the user is taken back to the main page with the menu list of different modes.

Code from main.py



```
1 import practice_mode
2 import one_minute_mode
3 import time
4 import clearing
5 from practice_mode import Practice
6 from game import gameMode
7 import menu
8
9 # Game mode objects for different levels
10 level_one = gameMode('Level One', 1, 5, 30)
11 level_two = gameMode('Level Two', 1, 7, 30)
12 level_three = gameMode('Level Three', 1, 10, 30)
13 level_four = gameMode('Level Four', 1, 12, 30)
14 level_five = gameMode('Level Five', 1, 15, 60)
15 level_six = gameMode('Level Six', 1, 2, 60)
16 level_seven = gameMode('Level Seven', 10, 20, 120)
17 level_eight = gameMode('Level Eight', 10, 50, 120)
18 level_nine = gameMode('Level Nine', 10, 100, 300)
19 level_ten = gameMode('Level Ten', 10, 900, 600)
20
21 # Practice mode objects for each difficulty level
22 easy = Practice(1, 7)
23 medium = Practice(1, 12)
24 hard = Practice(1, 20)
25
26 # Main code block
27 if __name__ == '__main__':
28     clearing.clear()
29     name = input('What is your name: ')
30     clearing.clear()
31     print(f'Welcome {name} to the timesTables Challenge!')
32     while True:
33         time.sleep(3)
```

- Have utilized the Object Orientated Modelling design.
- The classes/blueprints are the different modes of play.
- The instance of class/object is the different levels in game play or different difficulties in practice
- Allows data to be stored and updated more easily.

Code from practice_mode.py

```
File Edit Selection View Go Run Terminal Help practice_mode.py - NicoleHulett_T1A3 [WSL: Ubuntu-22.04] - Visual Studio Code
Preview README.md game.py M practice_mode.py X one_minute_mode.py wrapper.sh requirements.txt
src > practice_mode.py
6 class Practice:
7     def __init__(self, x, y):
8         self.x = x
9         self.y = y
10
11
12     def practise_mode(self):
13         correct_count = 0
14         incorrect_count = 0
15         start_time = time.time()
16         while correct_count < 100:
17             num = self.get_numbers()
18             answer = input(f'num[{0}] * {num[1]} = ')
19             if answer == 'q' or answer == 'Q':
20                 duration = round(time.time() - start_time, 2)
21                 if correct_count > 20 and incorrect_count < 3:
22                     print(f'You are AMAZING! You got {correct_count} correct and made {incorrect_count} error/s in {duration} seconds!!')
23                     break
24                 elif correct_count == 100:
25                     print(f'You are a champion! You got {correct_count} correct and made {incorrect_count} error/s in {duration} seconds!!')
26                     break
27                 else:
28                     print(f'You got {correct_count} correct and made {incorrect_count} error/s in {duration} seconds!!')
29                     break
30             elif num[0] * num[1] == int(answer):
31                 correct_count += 1
32                 num = self.get_numbers()
33                 continue
34             else:
35                 incorrect_count += 1
36                 num = self.get_numbers()
37                 continue
38
```

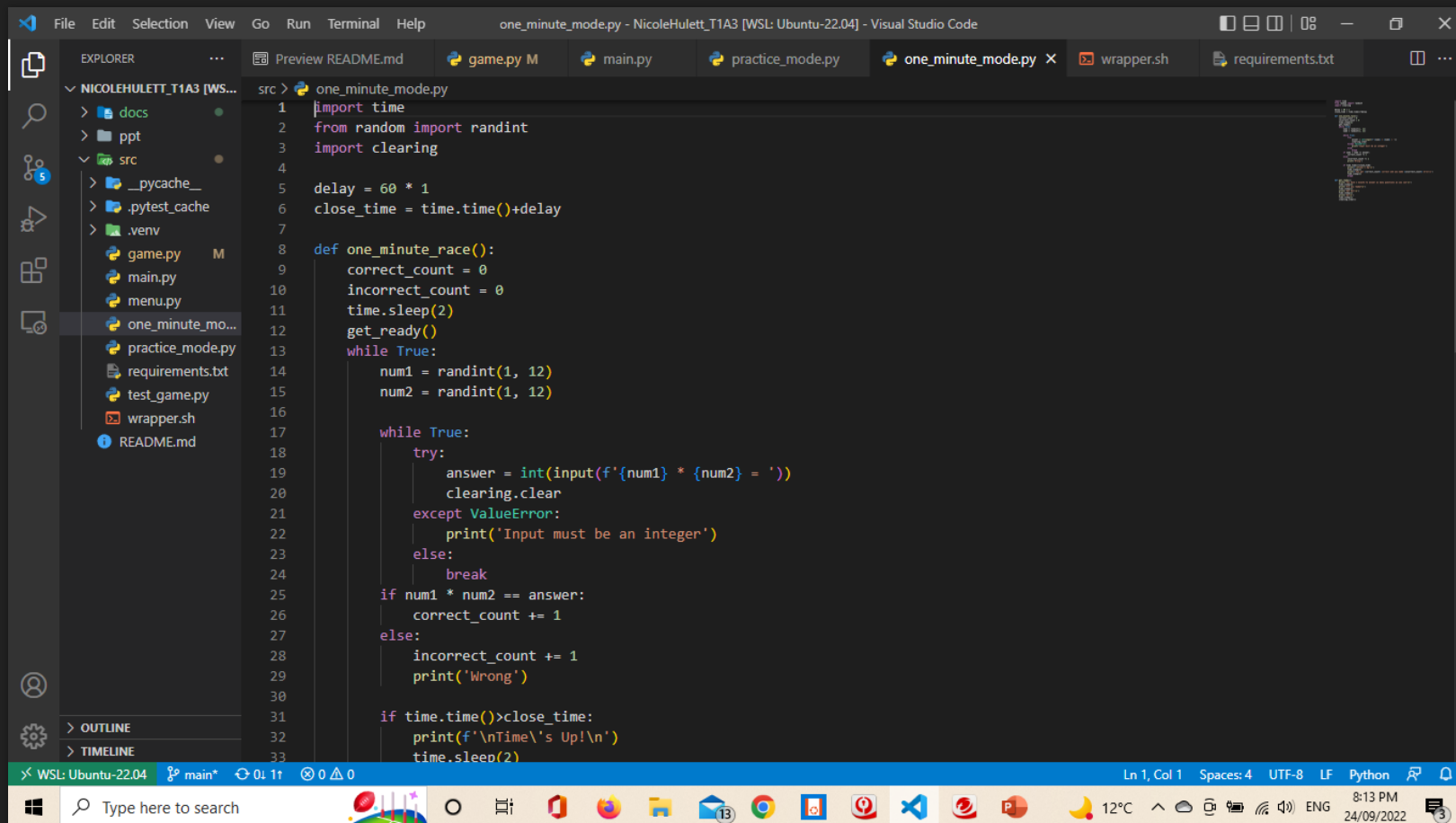
- Attributes are used to get the different number values to multiply according to difficulty level.
- Have used while loops to ensure the repetitive nature of the games.
- Have used if/elif/else to compensate for different scenarios that may arise.

Code from practice_number.py

```
File Edit Selection View Go Run Terminal Help practise_number.py - NicoleHulett_T1A3 [WSL: Ubuntu-22.04] - Visual Studio Code
main.py greeting.py U menu.py practice_mode.py practise_number.py U x README.md test_game.py
src > practise_number.py
5 class RangeError(Exception):
6     pass
7
8 def get_int():
9     number = int(input('What number would you like to practise multiplying by between 1 and 12: '))
10    if not number in range(1, 13):
11        raise RangeError(f'You entered {number}. You must enter a number between 1 and 12')
12    return number
13
14 def practise_number():
15     correct_answer = 0
16     incorrect = 0
17     while True:
18         try:
19             user_number = get_int()
20             break
21         except RangeError as err:
22             print(err.args)
23         except ValueError:
24             print('You must enter a number between 1 and 12')
25     # while correct_answer < 50:
26     for i in range(20):
27         number_two = randint(1, 12)
28         user_answer = input(f'{user_number} * {number_two} = ')
29         answer = user_number * number_two
30         if user_answer == 'q':
31             print(f'You got {correct_answer} correct and made {incorrect} error/s!')
32             time.sleep(3)
33             clearing.clear()
34             return
35         elif answer == int(user_answer):
36             correct_answer += 1
37     else:
```

- Utilised for loop to ensure this mode only had 20 questions asked.
- Raised an exception (RangeError)

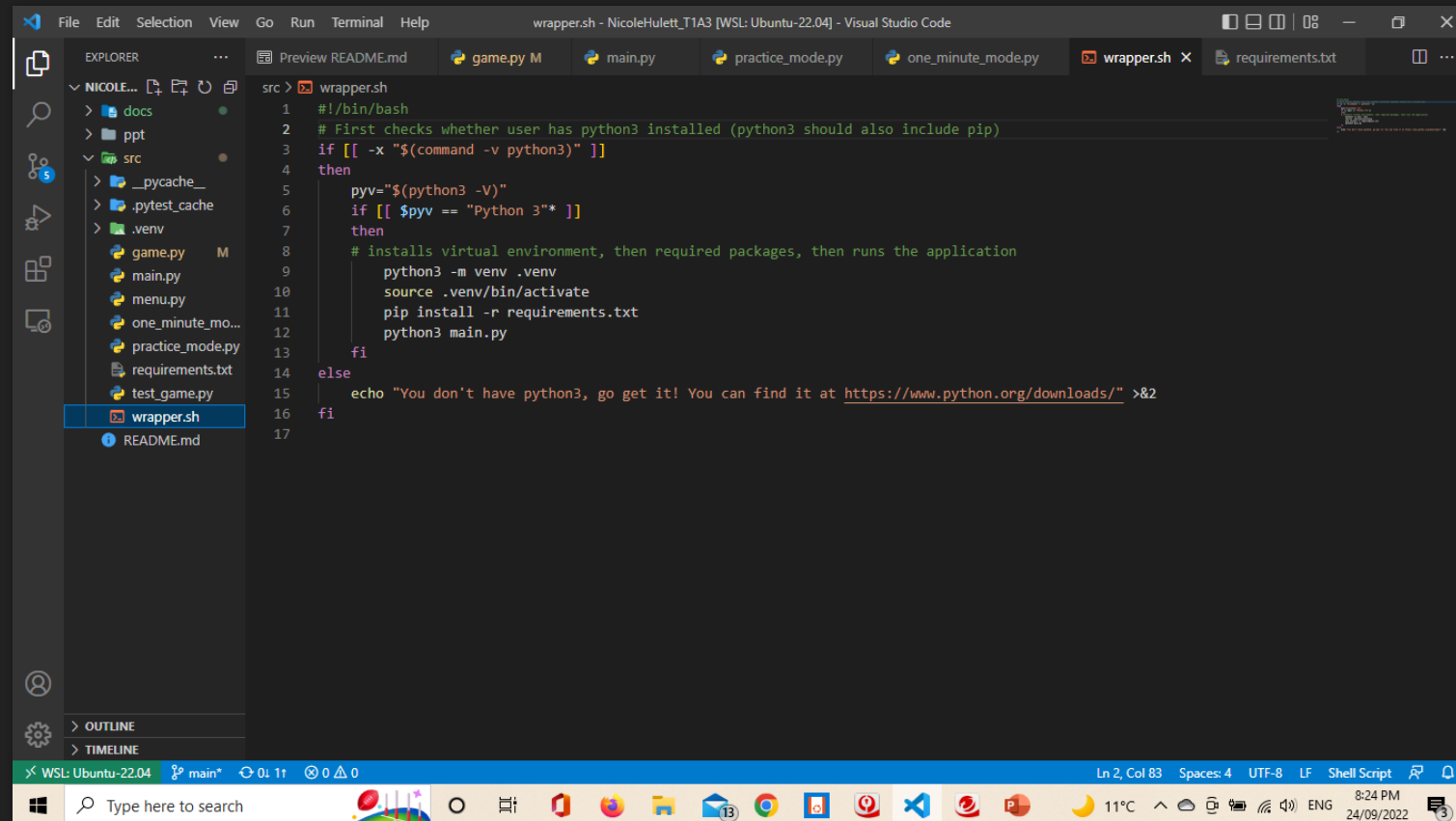
Code from one_minute_challenge.py



```
one_minute_mode.py - NicoleHulett_T1A3 [WSL: Ubuntu-22.04] - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
NICOLEHULETT_T1A3 [WSL: Ubuntu-22.04]
  docs
  ppt
  src
    __pycache__
    .pytest_cache
    .venv
    game.py M
    main.py
    menu.py
    one_minute_mode.py
    practice_mode.py
    requirements.txt
    test_game.py
    wrapper.sh
    README.md
  OUTLINE
  TIMELINE
src > one_minute_mode.py
1 import time
2 from random import randint
3 import clearing
4
5 delay = 60 * 1
6 close_time = time.time()+delay
7
8 def one_minute_race():
9     correct_count = 0
10    incorrect_count = 0
11    time.sleep(2)
12    get_ready()
13    while True:
14        num1 = randint(1, 12)
15        num2 = randint(1, 12)
16
17        while True:
18            try:
19                answer = int(input(f'{num1} * {num2} = '))
20                clearing.clear
21            except ValueError:
22                print('Input must be an integer')
23            else:
24                break
25        if num1 * num2 == answer:
26            correct_count += 1
27        else:
28            incorrect_count += 1
29            print('Wrong')
30
31    if time.time()>close_time:
32        print(f'\nTime\'s Up!\n')
33        time.sleep(2)
```

- Have imported different modules into app.
- This code utilized a countdown timer
- Have also used and try and except to ensure the user is entering a number/integer instead of anything else.

Bash script to execute application



```
src > wrapper.sh
1  #!/bin/bash
2  # First checks whether user has python3 installed (python3 should also include pip)
3  if [[ -x "$(command -v python3)" ]]
4  then
5      pyv="$(python3 -V)"
6      if [[ $pyv == "Python 3"* ]]
7      then
8          # installs virtual environment, then required packages, then runs the application
9          python3 -m venv .venv
10         source .venv/bin/activate
11         pip install -r requirements.txt
12         python3 main.py
13     fi
14 else
15     echo "You don't have python3, go get it! You can find it at https://www.python.org/downloads/" >&2
16 fi
17
```

WSL: Ubuntu-22.04 | main* | 0L 11 | 0 0 Δ 0 | Ln 2, Col 83 | Spaces: 4 | UTF-8 | LF | Shell Script

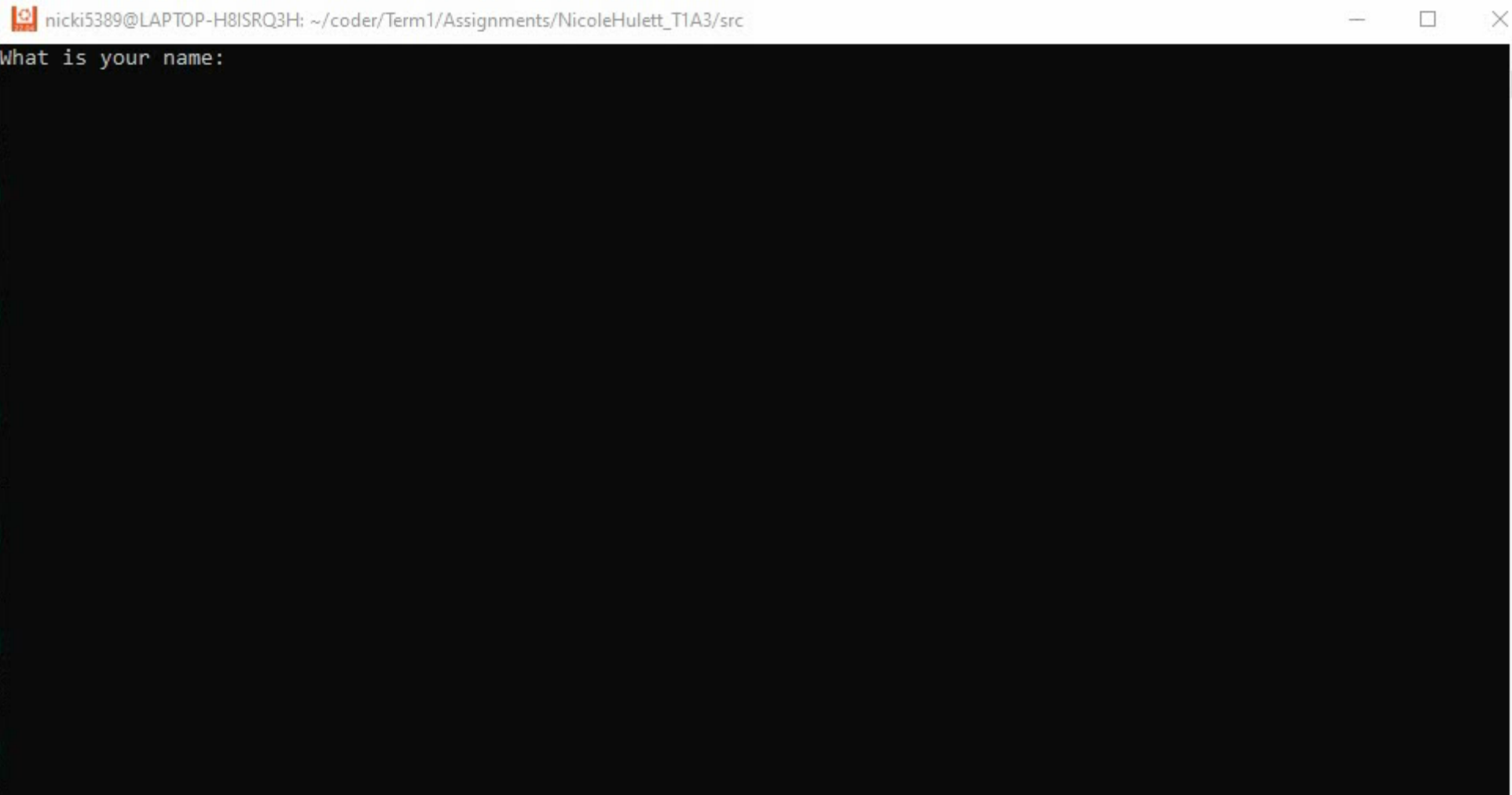
Development/build process

- Developed flow chart to map the basic application
- Created Trello board to help manage the different aspects of building a terminal application
- Worked on each feature individually, ensuring that code was always working
- Developed a main.py and ensured that it called the right modules at the right time.
- Created unit tests to test that the numbers were multiplying correctly and giving the correct answer.
- Also utilized manual testing of all the different features to ensure everything worked how it should and find errors.

Challenges/Ethical Issues/Favourite Parts

- My biggest challenge was the unit testing. Still need to work on that and create more tests.
- My favourite part was the actual coding and seeing everything come together. I also enjoyed using the terminal menu package.
- Ethical issue I was concerned about to begin with, was ensuring that I created an app that wasn't identical to anyone else's.

Now let's see it in action!

A screenshot of a terminal window. The title bar at the top shows the user 'nicki5389@LAPTOP-H8ISRQ3H' and the current directory '~/coder/Term1/Assignments/NicoleHulett_T1A3/src'. The terminal itself has a black background with white text. The prompt 'What is your name:' is visible at the top left of the terminal area.

```
nicki5389@LAPTOP-H8ISRQ3H: ~/coder/Term1/Assignments/NicoleHulett_T1A3/src  
What is your name:
```