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

```python
import practice_mode
import one_minute_mode
import time
import clearing
from practice_mode import Practice
from game import gameMode
import menu

# Game mode objects for different levels
level_one = gameMode('Level One', 1, 5, 30)
level_two = gameMode('Level Two', 1, 7, 30)
level_three = gameMode('Level Three', 1, 10, 30)
level_four = gameMode('Level Four', 1, 12, 30)
level_five = gameMode('Level Five', 1, 15, 60)
level_six = gameMode('Level Six', 1, 2, 60)
level_seven = gameMode('Level Seven', 10, 20, 120)
level_eight = gameMode('Level Eight', 10, 50, 120)
level_nine = gameMode('Level Nine', 10, 100, 300)
level_ten = gameMode('Level Ten', 10, 900, 600)

# Practice mode objects for each difficulty level
easy = Practice(1, 7)
medium = Practice(1, 12)
hard = Practice(1, 20)

# Main code block
if __name__ == '__main__':
    clearing.clear()
    name = input('What is your name: ')
    clearing.clear()
    print(f'Welcome {name} to the timesTables Challenge!')
    while True:
        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

```python
class Practice:
    def __init__(self, x, y):
        self.x = x
        self.y = y


    def practise_mode(self):
        correct_count = 0
        incorrect_count = 0
        start_time = time.time()
        while correct_count < 100:
            num = self.get_numbers()
            answer = input(f'{num[0]} * {num[1]} = ')
            if answer == 'q' or answer == 'Q':
                duration = round(time.time() - start_time, 2)
                if correct_count > 20 and incorrect_count < 3:
                    print(f'You are AMAZING! You got {correct_count} correct and made {incorrect_count} error/s in {duration} seconds!')
                    break
                elif correct_count == 100:
                    print(f'You are a champion! You got {correct_count} correct and made {incorrect_count} error/s in {duration} seconds!!')
                    break
                else:
                    print(f'You got {correct_count} correct and made {incorrect_count} error/s in {duration} seconds!')
                    break
            elif num[0] * num[1] == int(answer):
                correct_count += 1
                num = self.get_numbers()
                continue
            else:
                incorrect_count += 1
                num = self.get_numbers()
                continue
```

- 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

```python
class RangeError(Exception):
    pass

def get_int():
    number = int(input('What number would you like to practise multiplying by between 1 and 12: '))
    if not number in range(1, 13):
        raise RangeError(f'You entered {number}. You must enter a number between 1 and 12')
    return number

def practice_number():
        correct_answer = 0
        incorrect = 0
        while True:
            try:
                user_number = get_int()
                break
            except RangeError as err:
                print(err.args)
            except ValueError:
                print('You must enter a number between 1 and 12')
        # while correct_answer < 50:
        for i in range(20):
                    number_two = randint(1, 12)
                    user_answer = input(f'{user_number} * {number_two} = ')
                    answer = user_number * number_two
                    if user_answer == 'q':
                        print(f'You got {correct_answer} correct and made {incorrect} error/s!')
                        time.sleep(3)
                        clearing.clear()
                        return
                    elif answer == int(user_answer):
                        correct_answer += 1
                    else:
```

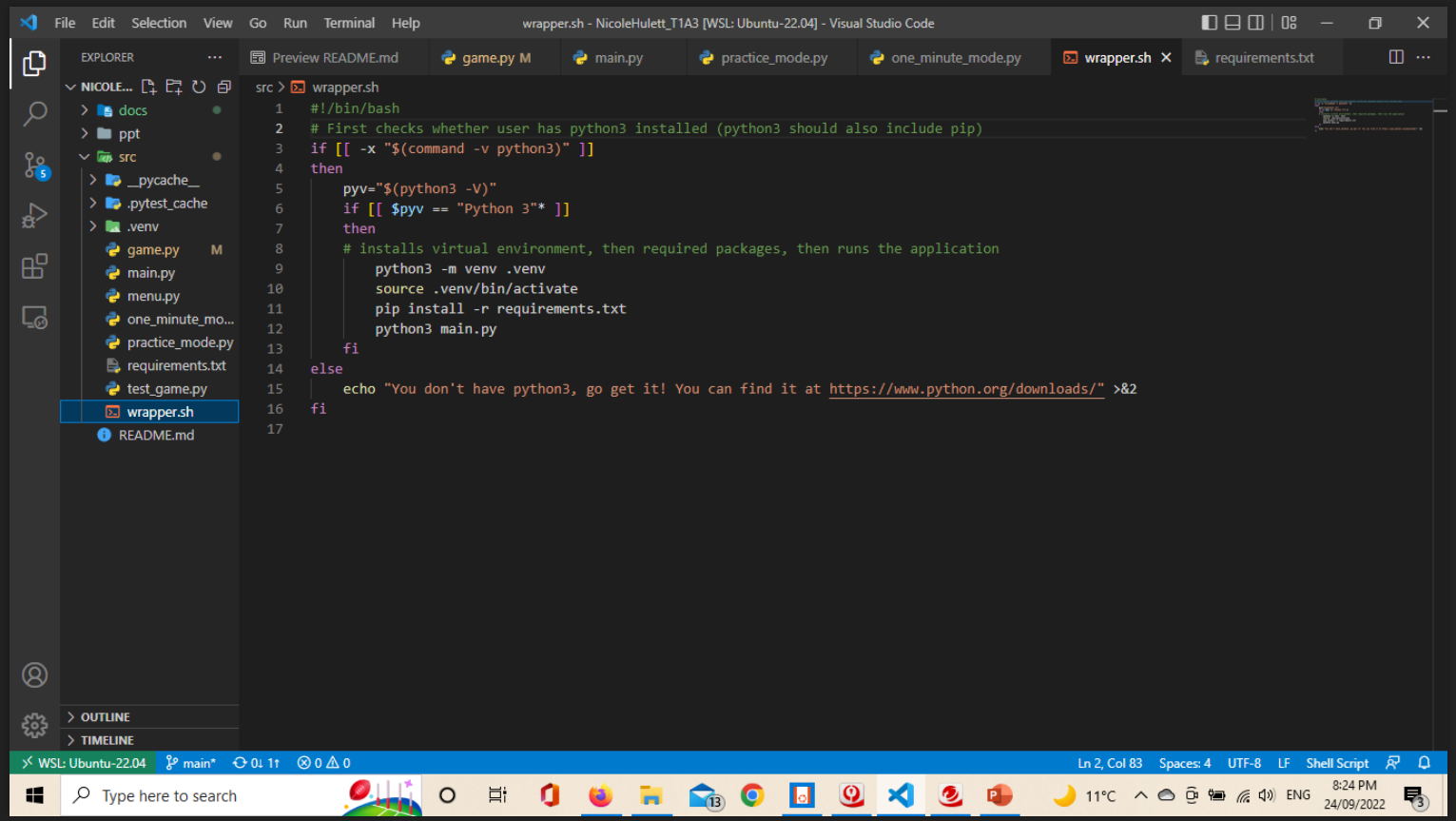
- Utilised for loop to ensure this mode only had 20 questions asked.
- Raised an exception (RangeError)

Code from one_minute_challenge.py

```python
import time
from random import randint
import clearing

delay = 60 * 1
close_time = time.time()+delay

def one_minute_race():
    correct_count = 0
    incorrect_count = 0
    time.sleep(2)
    get_ready()
    while True:
        num1 = randint(1, 12)
        num2 = randint(1, 12)

        while True:
            try:
                answer = int(input(f'{num1} * {num2} = '))
                clearing.clear
            except ValueError:
                print('Input must be an integer')
            else:
                break
        if num1 * num2 == answer:
            correct_count += 1
        else:
            incorrect_count += 1
            print('Wrong')

        if time.time()>close_time:
            print(f'\nTime\'s Up!\n')
            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



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!