TERM 1, ASSESSMENT 3 – PYTHON APPLICATION

ELLA JONES

STUDENT NUMBER:14447

CONCEPT:

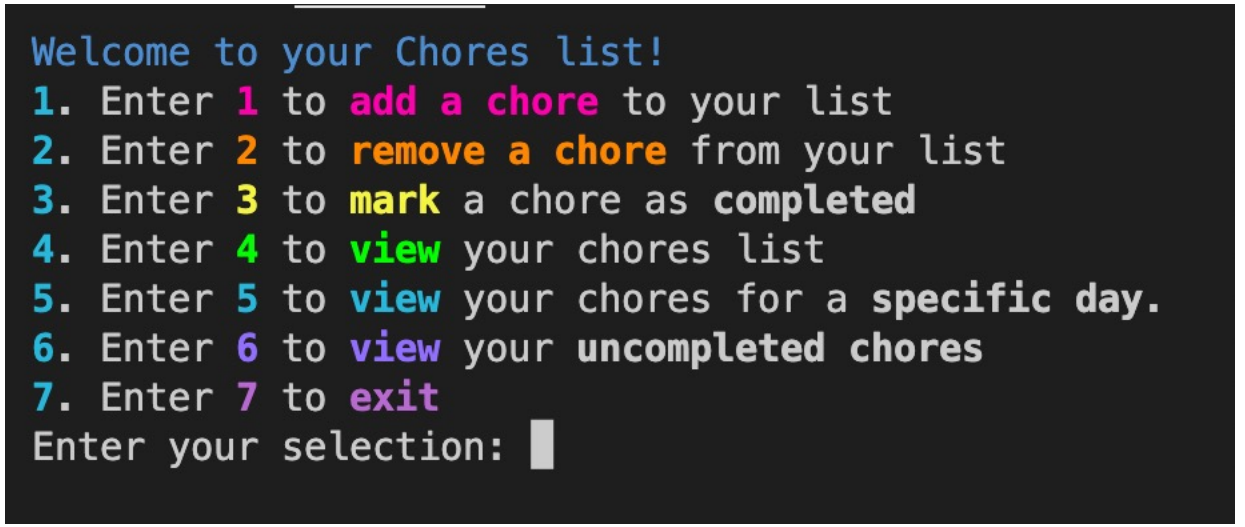
- A chore management application, which allows users to keep track of chores they need to complete and when they need to complete them each week.

FEATURES:

- Menu bar
- Add chore
- Remove chore
- Mark chore as complete
- View the entire chore list
- View chores for a specific day
- View all uncompleted chores

FEATURES: MENU BAR

- Allows user to view their options for the application
- Allows user to select an option from a list



FEATURES: MENU BAR (CODE 1)

```python
# The following function displays the menu bar to the user and allows them to input their selection.
def menu_bar():
    console.print("1. Enter [bold deep_pink1]1[/] to [bold deep_pink1]add a chore[/] to your list")
    console.print("2. Enter [bold dark_orange]2[/] to [bold dark_orange]remove a chore[/] from your list")
    console.print("3. Enter [bold yellow]3[/] to [bold yellow]mark[/] a chore as [bold]completed[/]")
    console.print("4. Enter [bold green1]4[/] to [bold green1]view[/] your chores list")
    console.print("5. Enter [bold bright_cyan]5[/] to [bold bright_cyan]view[/] your chores for a [bold]specific day[/].")
    console.print("6. Enter [bold slate_blue1]6[/] to [bold slate_blue1]view[/] your [bold]uncompleted chores[/]")
    console.print("7. Enter [bold dark_magenta]7[/] to [bold dark_magenta]exit[/]")
    try:
        selection = input("Enter your selection: ").strip()
    except Exception as e:
        print(e)
    else:
        return selection
```

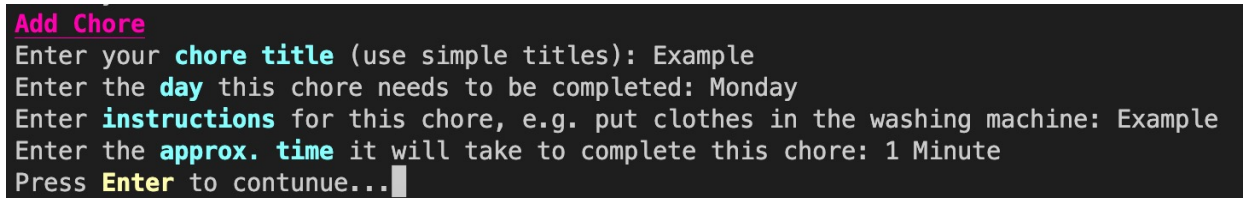
FEATURES: MENU BAR (CODE 2)

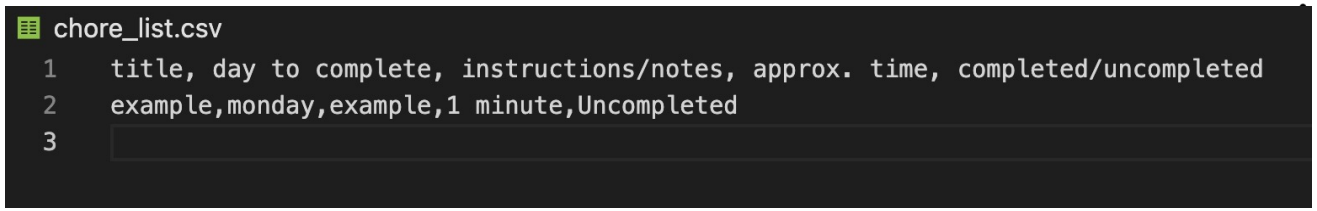
```
user_selection = str()

# The below while loop runs the function for the user's selection. (These functions are located in chore_functions.py)
while user_selection != "7":
    user_selection = menu_bar()

    if (user_selection == "1"):
        add_chore(file_name)
    elif (user_selection == "2"):
        remove_chore(file_name)
    elif (user_selection == "3"):
        mark_chore(file_name)
    elif (user_selection == "4"):
       console.print("View Chores", style="bold underline green1")
       view_chores(file_name)
    elif (user_selection == "5"):
       view_day(file_name)
    elif (user_selection == "6"):
        view_uncompleted(file_name)
    elif (user_selection == "7"):
        break
    else:
       print("Invalid Input")

    input(f"Press {attr('bold')}{fg('wheat_1')}Enter{attr('reset')} to contunue...")
```

FEATURES: ADD CHORE

- Allows user to add chores and input information for each chore
- This information includes:
  - Chore title
  - Day of week to complete (must be a day of the week or the program will throw an error)
  - Instructions for the chore
  - Approx. time it will take to complete

- All of this information is stored in a CSV file.
- Each chore will be in its own line within the CSV file.
- Each chore will automatically add as "uncompleted". The user will need to 'mark' the chore as complete, using the mark_chore() function, for the chore to show as "completed".





FEATURES: ADD CHORE (CODE)

```
# The below function allows the user to add a chore to their chore list (csv file) and to
# input the chore title, day to be completed, instructions, and approx. time to complete.
# All chores are entered as "uncompleted" and are in their own line.
def add_chore(file_name):
    console.print("Add Chore", style="bold underline deep_pink1")
    try:
        chore_title = input(f"Enter your {attr('bold')}{fg('dark_slate_gray_1')}chore title{attr('reset')} (use simple titles): ").lower().strip()
    except Exception as e:
        print(e)
    try:
        chore_day = input(f"Enter the {attr('bold')}{fg('dark_slate_gray_1')}day{attr('reset')} this chore needs to be completed: ").lower().strip()
        check_day(chore_day)
    except WeekDayError as e:
        print(e)
    else:
        try:
            chore_instructions = input(f"Enter {attr('bold')}{fg('dark_slate_gray_1')}instructions{attr('reset')} for this chore, e.g. put clothes in the washing machine: ").lower()
        except Exception as e:
            print(e)
        try:
            chore_time = input(f"Enter the {attr('bold')}{fg('dark_slate_gray_1')}approx. time{attr('reset')} it will take to complete this chore: ").lower()
        except Exception as e:
            print(e)
        try:
            with open(file_name, "a") as chore_file:
                writer = csv.writer(chore_file)
                writer.writerow([chore_title, chore_day, chore_instructions, chore_time, "Uncompleted"])
        except FileNotFoundError as e:
            print("oops, there was an error writing in this file!")
        except Exception as e:
            print(e)
```

FEATURES: REMOVE CHORE

- Allows user to remove a chore from their chores list (csv file)
- This function will ask the user to input the title of the chore they would like to remove.
- This function will remove the chore, as well as the whole line of that chore from the csv file.
- The view_chore() function will be called on before and after the remove_chore() function runs, so the user can see the titles of the chores, and also see that the chore they chose has been removed correctly.

```
Remove Chore
CURRENT LIST OF CHORES:
Chore: 'example1' is UNCOMPLETED. Day to complete: monday. Approx. time to complete: 1 minute. Instructions: example.
Chore: 'example2' is UNCOMPLETED. Day to complete: tuesday. Approx. time to complete: 2 minutes. Instructions: example.
Chore: 'example3' is UNCOMPLETED. Day to complete: wednesday. Approx. time to complete: 3 minutes. Instructions: example.
Chore: 'example4' is UNCOMPLETED. Day to complete: thursday. Approx. time to complete: 4 minutes. Instructions: example.
Chore: 'example5' is UNCOMPLETED. Day to complete: monday. Approx. time to complete: 5 minutes. Instructions: example.
Enter the 'chore title' that you want to remove: example2
EDITED LIST:
Chore: 'example1' is UNCOMPLETED. Day to complete: monday. Approx. time to complete: 1 minute. Instructions: example.
Chore: 'example3' is UNCOMPLETED. Day to complete: wednesday. Approx. time to complete: 3 minutes. Instructions: example.
Chore: 'example4' is UNCOMPLETED. Day to complete: thursday. Approx. time to complete: 4 minutes. Instructions: example.
Chore: 'example5' is UNCOMPLETED. Day to complete: monday. Approx. time to complete: 5 minutes. Instructions: example.
Press Enter to contunue...
```

FEATURES: REMOVE CHORE (CODE)

```
# The below function allows the user to select a chore they would like to remove from
# their chores list (csv file) and deletes the corresponding line from the list.
def remove_chore(file_name):
    console.print("Remove Chore", style="bold underline dark_orange")
    console.print("CURRENT LIST OF CHORES:", style="bold yellow")
    view_chores(file_name)
    try:
        chore_title = input(f"Enter the {attr('bold')}{fg('purple_3')}'chore title'{attr('reset')} that you want to remove: ").lower().strip()
    except Exception as e:
        print(e)
    chore_lists = []
    try:
        with open(file_name, "r") as chore_file:
            reader = csv.reader(chore_file)
            for row in reader:
                if (chore_title != row[0]):
                    chore_lists.append(row)
        with open(file_name, "w") as chore_file:
            writer = csv.writer(chore_file)
            writer.writerows(chore_lists)
    except FileNotFoundError as e:
        print("oops, there was an error removing this chore from your chores list file!")
    except Exception as e:
        print(e)
    console.print("EDITED LIST:", style="bold yellow")
    view_chores(file_name)
```

FEATURES: MARK CHORE

- Allows the user to mark a chore as “complete”
- Will ask the user to input the chore title they want to mark as complete.
- The view_chore() function will be called on before the mark_chore() function runs, so the user can see the titles of the chores.

```
Mark Chore
Chore: 'example1' is UNCOMPLETED. Day to complete: monday. Approx. time to complete: 1 minute. Instructions: example.
Chore: 'example3' is UNCOMPLETED. Day to complete: wednesday. Approx. time to complete: 3 minutes. Instructions: example.
Chore: 'example4' is UNCOMPLETED. Day to complete: thursday. Approx. time to complete: 4 minutes. Instructions: example.
Chore: 'example5' is UNCOMPLETED. Day to complete: monday. Approx. time to complete: 5 minutes. Instructions: example.
Enter the 'chore title' that you want to mark as complete: example3
Enter the day that you completed this chore: wednesday
Enter any notes for this chore (if you would like to leave as blank, enter 'none'): none
Enter the time it took you to complete this chore: 5 minutes
Press Enter to contunue...
```

FEATURES: MARK CHORE (CODE)

```
# This function allows the user to mark a chore as "Completed" and input updated information about
# the chore (e.g. how long it actually took and any notes they have.)
def mark_chore(file_name):
    console.print("Mark Chore", style="bold underline yellow")
    view_chores(file_name)
    try:
        chore_title = input(f"Enter the {attr('bold')}{fg('purple_3')}'chore title'{attr('reset')} that you want to mark as complete: ").lower().stri
    except Exception as e:
        print(e)
    try:
        chore_day = input(f"Enter the {attr('bold')}{fg('dark_slate_gray_1')}day{attr('reset')} that you completed this chore: ").lower().strip()
        check_day(chore_day)
    except WeekDayError as e:
        print(e)
    else:
        try:
            chore_instructions = input(f"Enter any {attr('bold')}{fg('dark_slate_gray_1')}notes{attr('reset')} for this chore (if you would like to l
        except Exception as e:
            print(e)
        try:
            chore_time = input(f"Enter the {attr('bold')}{fg('dark_slate_gray_1')}time{attr('reset')} it took you to complete this chore: ").lower()
        except Exception as e:
            print(e)
        chore_lists = []
        try:
            with open(file_name, "r") as chore_file:
                reader = csv.reader(chore_file)
                for row in reader:
                    if (chore_title == row[0]):
                        chore_lists.append([chore_title, chore_day, chore_instructions,chore_time , "Completed"])
                    else:
                        chore_lists.append(row)
            with open(file_name, "w") as chore_file:
                writer = csv.writer(chore_file)
                writer.writerows(chore_lists)
        except FileNotFoundError as e:
            print("Oops, there was an error in marking your chore in your chores list file!")
        except Exception as e:
            print(e)
```

FEATURES: VIEW CHORES

- Allows the user to view the chore list that is stored in the CSV file.
- Displays each chore as either “Chore: ‘[Chore Title]’ is UNCOMPLETED. Day to complete: [Day]. Approx. time to complete: [Time]. Instructions: [Instructions].” for uncompleted chores, OR;
- “Chore: ‘[Chore Title]’ was completed on [Day] in [Time]. Notes: [Instructions].” for completed chores

```
View Chores
Chore: 'example1' is UNCOMPLETED. Day to complete: monday. Approx. time to complete: 1 minute. Instructions: example.
Chore: 'example3' was COMPLETED on wednesday in 5 minutes. Notes: none.
Chore: 'example4' is UNCOMPLETED. Day to complete: thursday. Approx. time to complete: 4 minutes. Instructions: example.
Chore: 'example5' is UNCOMPLETED. Day to complete: monday. Approx. time to complete: 5 minutes. Instructions: example.
Press Enter to contunue...
```

FEATURES: VIEW CHORES (CODE)

```
# This function allows the user to view their chores list (csv file) in read mode.
def view_chores(file_name):
    try:
        with open(file_name, "r") as chore_file:
            reader = csv.reader(chore_file)
            reader.__next__()
            for row in reader:
                if(row[4] == "Completed"):
                    print(f"Chore: {attr('bold')}{fg('purple_3')}'{row[0]}'{attr('reset')} was {attr('bold')}{fg('pale_green_1a')}COMPLETED{attr('reset')} on {row[1]} in {row[3]}. Notes: {row[2]}.")
                else:
                    print(f"Chore: {attr('bold')}{fg('purple_3')}'{row[0]}'{attr('reset')} is {attr('bold')}{fg('indian_red_1a')}UNCOMPLETED{attr('reset')}. Day to complete: {row[1]}. Approx. time to complete: {row[3]}. Instructions: {row[2]}.")
    except FileNotFoundError as e:
        print("Oops, there was an error in reading your chores list file!")
    except Exception as e:
        print(e)
```

FEATURES: VIEW SPECIFIC DAY

- Allows the user to view the chores for a specific day of the week.
- Asks user to input which day of the week they would like to view.
- If the user inputs something that is not a day of the week, an error is thrown.

```
View chores for a specific day
Enter the day you would like to view: Monday
Monday's Chores:
Chore: 'example1'. Uncompleted. Approx. time to complete: 1 minute. Instructions: example.
Chore: 'example5'. Uncompleted. Approx. time to complete: 5 minutes. Instructions: example.
Press Enter to contunue...
```

FEATURES: VIEW SPECIFIC DAY (CODE)

```
# This function allows the user to input a weekday and view all chores they need to complete
# on that day. If they don't enter a weekday, it will throw the WeekDayError (custom error).
def view_day(file_name):
    console.print("View chores for a specific day", style="bold underline bright_cyan")
    try:
        chore_day = input(f"Enter the {attr('bold')}{fg('dark_slate_gray_1')}day{attr('reset')} you would like to view: ").lower().strip()
        check_day(chore_day)
    except WeekDayError as e:
        print(e)
    else:
        if (chore_day == "monday"):
            console.print("Monday's Chores:", style="bold plum1")
        elif (chore_day == "tuesday"):
            console.print("Tuesday's Chores:", style="bold plum1")
        elif (chore_day == "wednesday"):
            console.print("Wednesday's Chores:", style="bold plum1")
        elif (chore_day == "thursday"):
            console.print("Thursday's Chores:", style="bold plum1")
        elif (chore_day == "friday"):
            console.print("Friday's Chores:", style="bold plum1")
        elif (chore_day == "saturday"):
            console.print("Saturday's Chores:", style="bold plum1")
        elif (chore_day == "sunday"):
            console.print("Sunday's Chores:", style="bold plum1")
        else:
            console.print("Invalid Input", style="red")
        try:
            with open(file_name, "r") as chore_file:
                reader = csv.reader(chore_file)
                reader.__next__()
                for row in reader:
                    if(row[1] == chore_day):
                        print(f"Chore: {attr('bold')}{fg('purple_3')}'{row[0]}'{attr('reset')}. {attr('bold')}{fg('sandy_brown')}{row[4]}{attr('reset')}. Approx. time to complete: {row[3]}. Instructions: {row[2]}.")
        except FileNotFoundError as e:
            print("Oops, there was an error in reading your chores list file!")
        except Exception as e:
            print(e)
```

FEATURES: VIEW UNCOMPLETED CHORES

- Allows the user to view all chores that are listed as "uncompleted" in the csv file.
- The completed chores will not be displayed when this function is called.

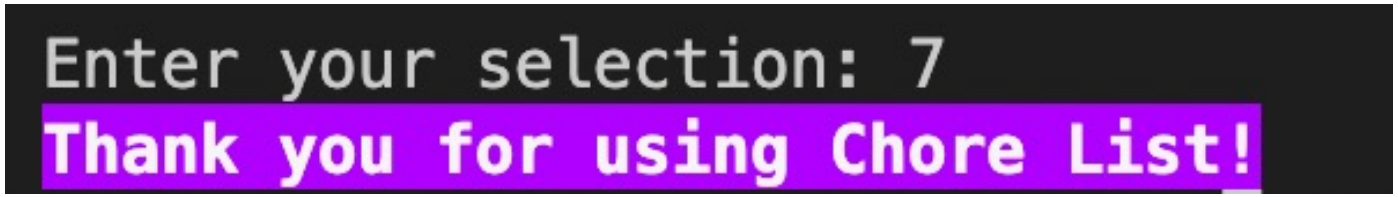
```
UNCOMPLETED CHORES:
Chore: 'example1'. Day to complete: monday. Approx. time to complete: 1 minute. Instructions: example.
Chore: 'example4'. Day to complete: thursday. Approx. time to complete: 4 minutes. Instructions: example.
Chore: 'example5'. Day to complete: monday. Approx. time to complete: 5 minutes. Instructions: example.
Press Enter to contunue...
```

FEATURES: VIEW UNCOMPLETED CHORES (CODE)

```
# This function allows the user to view all of the chores marked as "Uncompleted."
def view_uncompleted(file_name):
    console.print("UNCOMPLETED CHORES:", style="bold underline slate_blue1")
    try:
        with open(file_name, "r") as chore_file:
            reader = csv.reader(chore_file)
            reader.__next__()
            for row in reader:
                if(row[4] == "Uncompleted"):
                    print(f"Chore: {attr('bold')}{fg('yellow')}'{row[0]}'{attr('reset')}. Day to complete: {row[1]}. Approx. time to complete: {row[3]}. Instructions: {row[2]}.")
    except FileNotFoundError as e:
        print("Oops, there was an error in reading your chores list file!")
    except Exception as e:
        print(e)
```

FEATURES: EXIT

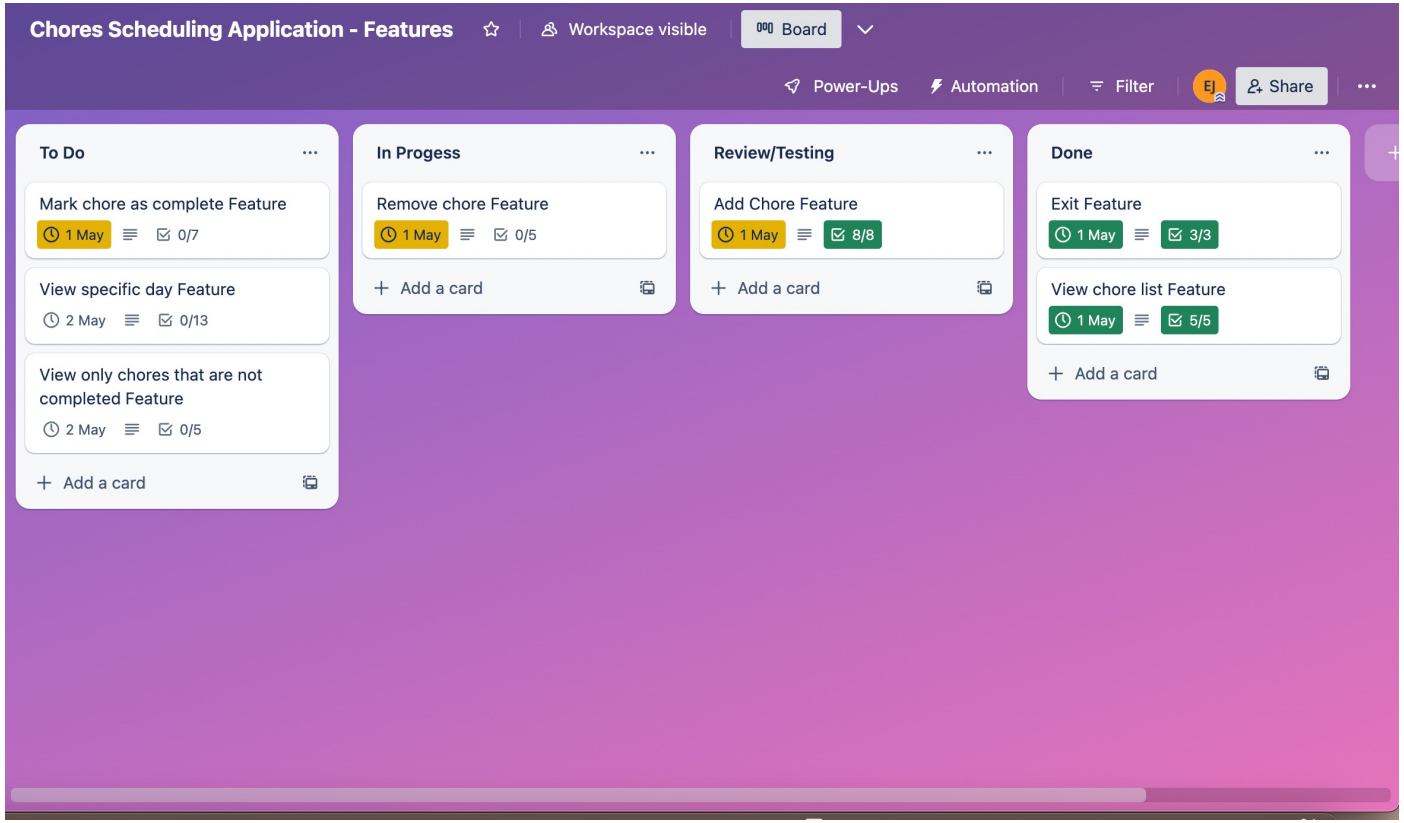
- Allows the user to exit from the application
- Thanks the user for using the application.



WALK THROUGH OF APPLICATION

IMPLEMENTATION PLAN

- Trello Board
  - All features on separate cards
  - Deadlines
  - Checklists
  - Split into 4 sections:
    - To Do
    - In Progress
    - Review/Testing
    - Done



PACKAGES

- All packages are downloaded to the chores-venv/bin folder
- Python Built-In Packages used:
  - CSV – to read, write and append to csv files
- External Packages used
  - Colored – to style the output. Used in conjunction with;
  - Rich – to style the output.
  - Pytest – to create and run tests for two of the functions

TESTS- USING PYTEST

- Test 1 :
  - Checks if the add_chore() function correctly adds a new line to the csv file, using a test csv file.

- Test 2:
  - Checks if the remove_chore() function correctly removes a line from the csv file, using a test csv file.
  - Pytest Fixture (Setup & Teardown)

  - Both tests pass.
  - Both tests use monkeypatch to assign values to the input for each function.

```
(chores-venv) ellajones@192-168-1-114 src % pytest
============================= test session starts ==============================
platform darwin -- Python 3.11.2, pytest-7.3.1, pluggy-1.0.0
rootdir: /Users/ellajones/Desktop/Coding_Academy/Term_1/Assessment_3_TerminallApp/EllaJones_T1A3/src
collected 2 items

test_chores.py .                                                         [ 50%]
test_remove.py .                                                         [100%]

============================== 2 passed in 0.04s ===============================
(chores-venv) ellajones@192-168-1-114 src %
```

TEST 1 - ADD_CHORE (CODE)

```python
import pytest
import csv
from chore_functions import add_chore

test_file_name = "tests/test_chores.csv"

# Test 1:
# This test checks if the add_chore() function (located in chore_functions.py) correctly adds a
# new line to the corresponding csv file (chore_list.csv). This test uses the test_chores.csv file
# (located in the tests folder) to test this function.
def test_add(monkeypatch):
    original_length = 0
    with open(test_file_name) as f:
        reader = csv.reader(f)
        original_length = sum(1 for row in reader)
    inputs = iter(['Test Chore 1', 'monday', 'test', 'test', ])
    monkeypatch.setattr('builtins.input', lambda _: next(inputs))
    add_chore(test_file_name)
    with open(test_file_name) as f:
        reader = csv.reader(f)
        new_length = sum(1 for row in reader)
    print(original_length)
    print(new_length)
    assert new_length == original_length + 1
```

TEST 2 – REMOVE CHORE (FIXTURE CODE)

```python
import pytest
import csv
from chore_functions import remove_chore, view_chores
from pytest import fixture

test_file_name = "tests/test_remove.csv"

# I have used pytest fixture to create a setup that adds a line to the csv file so that
# the remove_chore() test passes.
@fixture(autouse=True, scope='session')
def my_fixture():
    # setup_stuff
    print("--------------------setup------------------------")
    with open(test_file_name, "w") as f:
        writer = csv.writer(f)
        writer.writerow(["test chore", "monday", "test", "test", "Uncompleted"])

    yield
    # teardown_stuff
    print("--------------------teardown------------------------")
```

TEST 2 – REMOVE CHORE (CODE)

```python
# Test 2:
# This test checks if the remove_chore() function (located in chore_functions.py) correctly removes an
# line from the corresponding csv file (chore_list.csv). This test uses the test_remove.csv file
# (located in the tests folder) to test this function.
def test_remove(monkeypatch):
    original_length = 0
    with open(test_file_name) as f:
        reader = csv.reader(f)
        original_length = sum(1 for row in reader)
    monkeypatch.setattr('builtins.input', lambda _: "test chore")
    remove_chore(test_file_name)
    with open(test_file_name) as f:
        reader = csv.reader(f)
        new_length = sum(1 for row in reader)
    print(original_length)
    print(new_length)
    assert new_length == original_length - 1
```

BASH SCRIPTS

- I have two Bash scripts in my source code.

- Bash Script 1: run.sh
  - Clears the terminal
  - Runs the application for the user

- Bash Script 2: run_setup.sh
  - This script creates the chores-venv for the user and activates this venv.
  - Installs the requirements listed in requirements.txt
  - It then runs the run.sh script.

- The run.sh script is run from the run_setup.sh.
- The user only needs to run the run_setup.sh script (by entering ./run_setup.sh in the terminal) for both scripts to run.

BASH SCRIPTS (CODE)

$ run.sh

```
#!/bin/bash

# This bash script clears the setup information from
# the terminal and runs the application for the user.

clear
python3 main.py
```

$ run_setup.sh

```
#!/bin/bash

# This bash script sets up the requirements for the
# application to run and links to the ./run.sh script

python3 -m venv chores-venv
source chores-venv/bin/activate
pip3 install -r requirements.txt

./run.sh
```

ERROR HANDLING

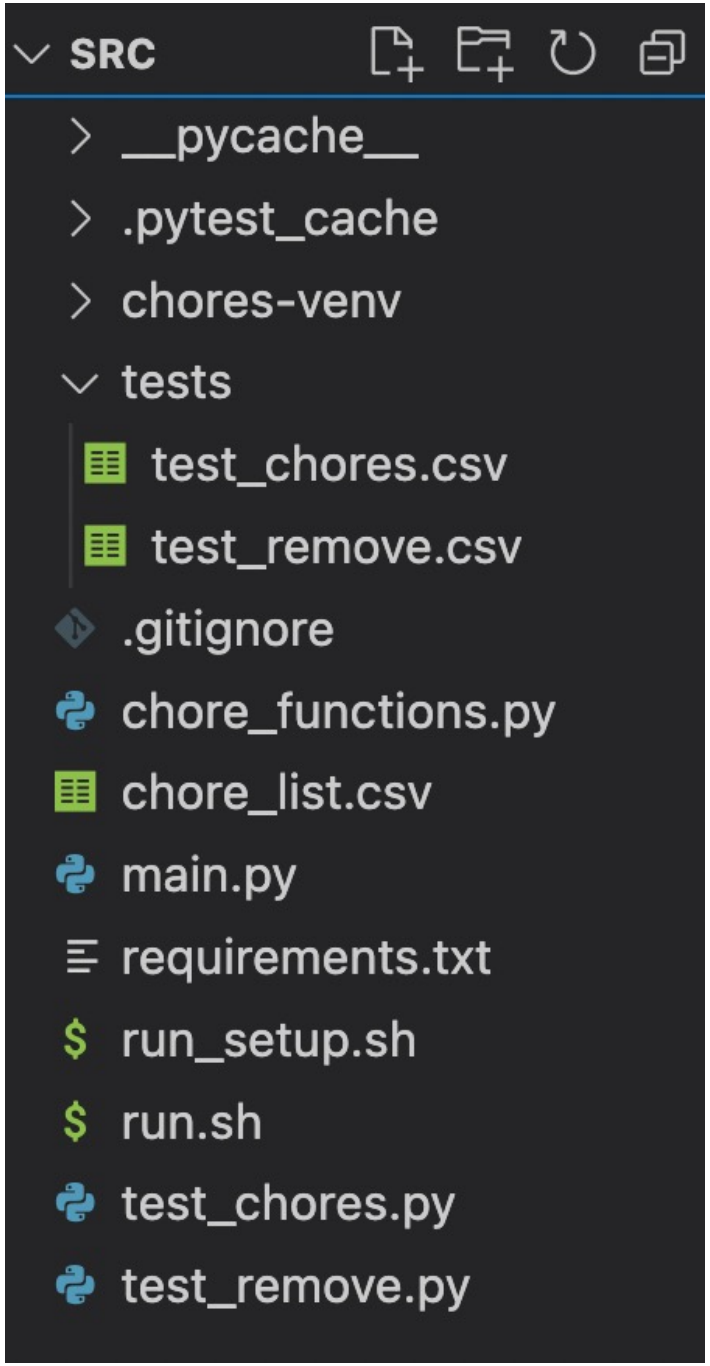
- Where possible, have used try & except blocks
- Mostly to handle any FileNotFoundError errors when files are opened, read, or written into.
- Created a custom exception (WeekDayError) to handle errors when the user inputs a value that is not a weekday, when a weekday is needed.
- Also used if statements (e.g. else: print("Invalid input"))

ERROR HANDLING (EXAMPLES IN CODE)

```
# The below code checks if the chore_list.csv file already exists, if it doesn't exist it it creates
# the file for the user and adds the first line (example line)
try:
    chore_file = open(file_name, "r")
except FileNotFoundError as e:
    chore_file = open(file_name, "w")
    chore_file.write("title, day to complete, instructions/notes, approx. time, completed/uncompleted\n")
    chore_file.close()
except Exception as e:
    print(e)
else:
    chore_file.close()
```

```
# The below custom exception is used on inputs that require a weekday to be
# entered. If a weekday is not entered, an error will be thrown.
class WeekDayError(Exception):
    pass
def check_day(a):
    if not (a == "monday" or a == "tuesday" or a == "wednesday" or a == "thursday" or a == "friday" or a ==
        raise WeekDayError ('That is not a week day, silly!')
```

ORGANISATION



STYLE GUIDE

- Used the Pep8 style guide (as referenced in the readme document).

THE DEVELOPMENT PROCESS

CHALLENGES

- Pytest
- Error Handling

HIGHLIGHTS

- When things worked!
- Styling the output using the Rich & Colored packages