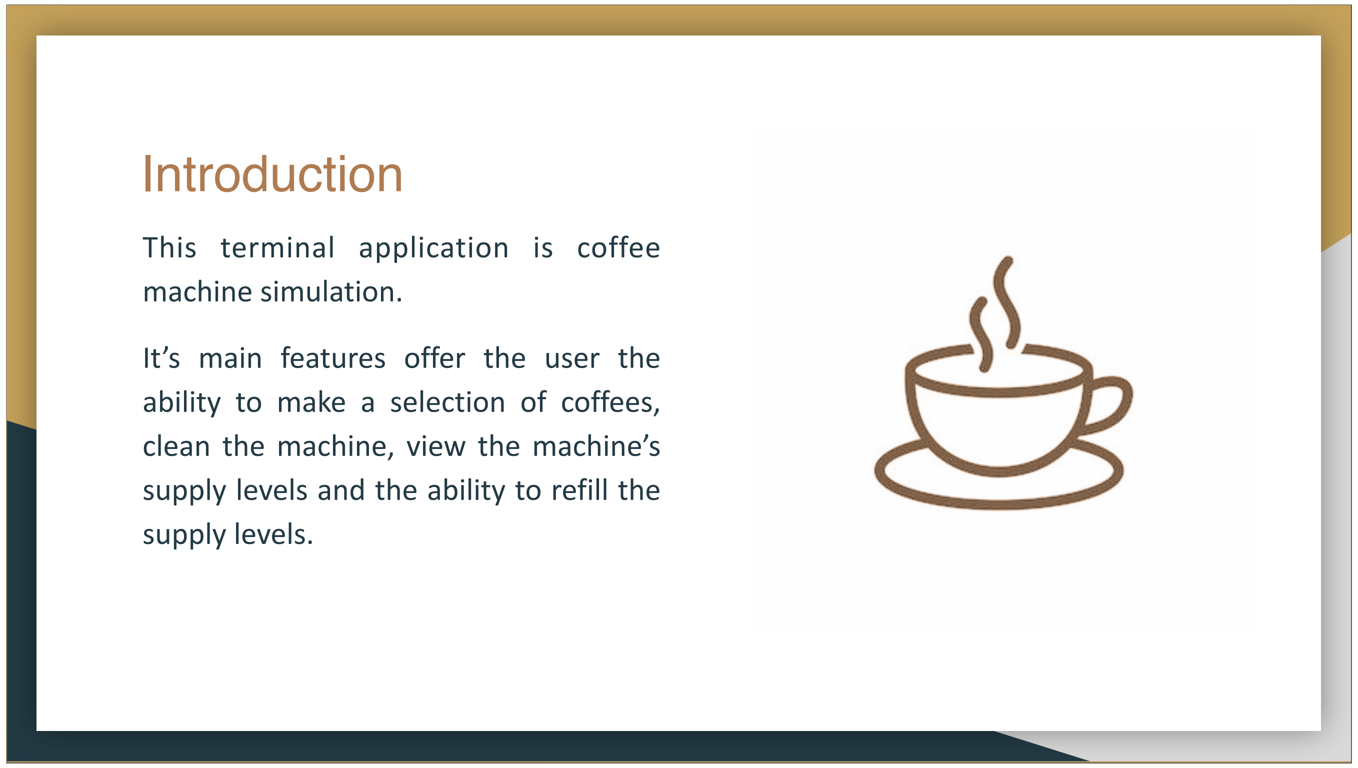
T1A3 TERMINAL APPLICATION

Presented by Derdriu McAteer

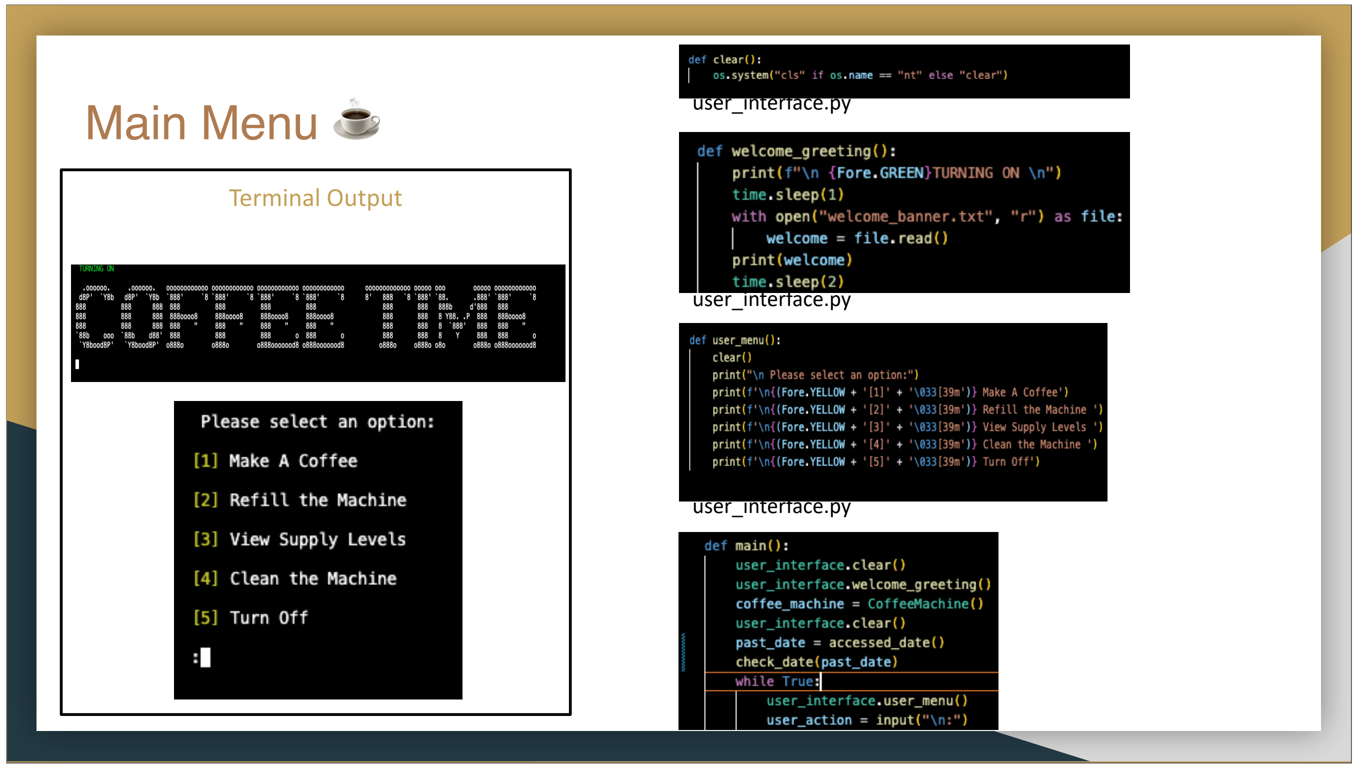


Introduction

This terminal application is coffee machine simulation.

It's main features offer the user the ability to make a selection of coffees, clean the machine, view the machine's supply levels and the ability to refill the supply levels.

This terminal application is coffee machine simulation. It's main features offer the user the ability to make a selection of coffees, clean the machine, view the machine's supply levels and the ability to refill the supply levels.



Main Menu ☕

Terminal Output

```
 TURNING ON

  .oooooo.     .oooooo.   oooooooooooo oooooooooooo oooooooooooo oooooooooooo      oooooooooooo ooooo ooo        ooooo oooooooooooo
 d8P'  `Y8b   d8P'  `Y8b  `888'     `8 `888'     `8 `888'     `8 `888'     `8      8'   888   `8 `888' `88.       .888' `888'     `8
888          888      888  888          888          888          888                   888       888   888b     d'888   888
888          888      888  888oooo8     888oooo8     888oooo8     888oooo8              888       888   8 Y88. .P  888   888oooo8
888          888      888  888    "     888    "     888    "     888    "              888       888   8  `888'   888   888    "
`88b    ooo  `88b    d88'  888          888          888       o  888       o           888       888   8    Y     888   888       o
 `Y8bood8P'   `Y8bood8P'  o888o        o888o        o888ooooood8 o888ooooood8          o888o     o888o o8o        o888o o888ooooood8

▌
```

```
 Please select an option:

[1] Make A Coffee

[2] Refill the Machine

[3] View Supply Levels

[4] Clean the Machine

[5] Turn Off

:█
```

```python
def clear():
    os.system("cls" if os.name == "nt" else "clear")
```

user_interface.py

```python
def welcome_greeting():
    print(f"\n {Fore.GREEN}TURNING ON \n")
    time.sleep(1)
    with open("welcome_banner.txt", "r") as file:
        welcome = file.read()
    print(welcome)
    time.sleep(2)
```

user_interface.py

```python
def user_menu():
    clear()
    print("\n Please select an option:")
    print(f'\n{(Fore.YELLOW + '[1]' + '\033[39m')} Make A Coffee')
    print(f'\n{(Fore.YELLOW + '[2]' + '\033[39m')} Refill the Machine ')
    print(f'\n{(Fore.YELLOW + '[3]' + '\033[39m')} View Supply Levels ')
    print(f'\n{(Fore.YELLOW + '[4]' + '\033[39m')} Clean the Machine ')
    print(f'\n{(Fore.YELLOW + '[5]' + '\033[39m')} Turn Off')
```

user_interface.py

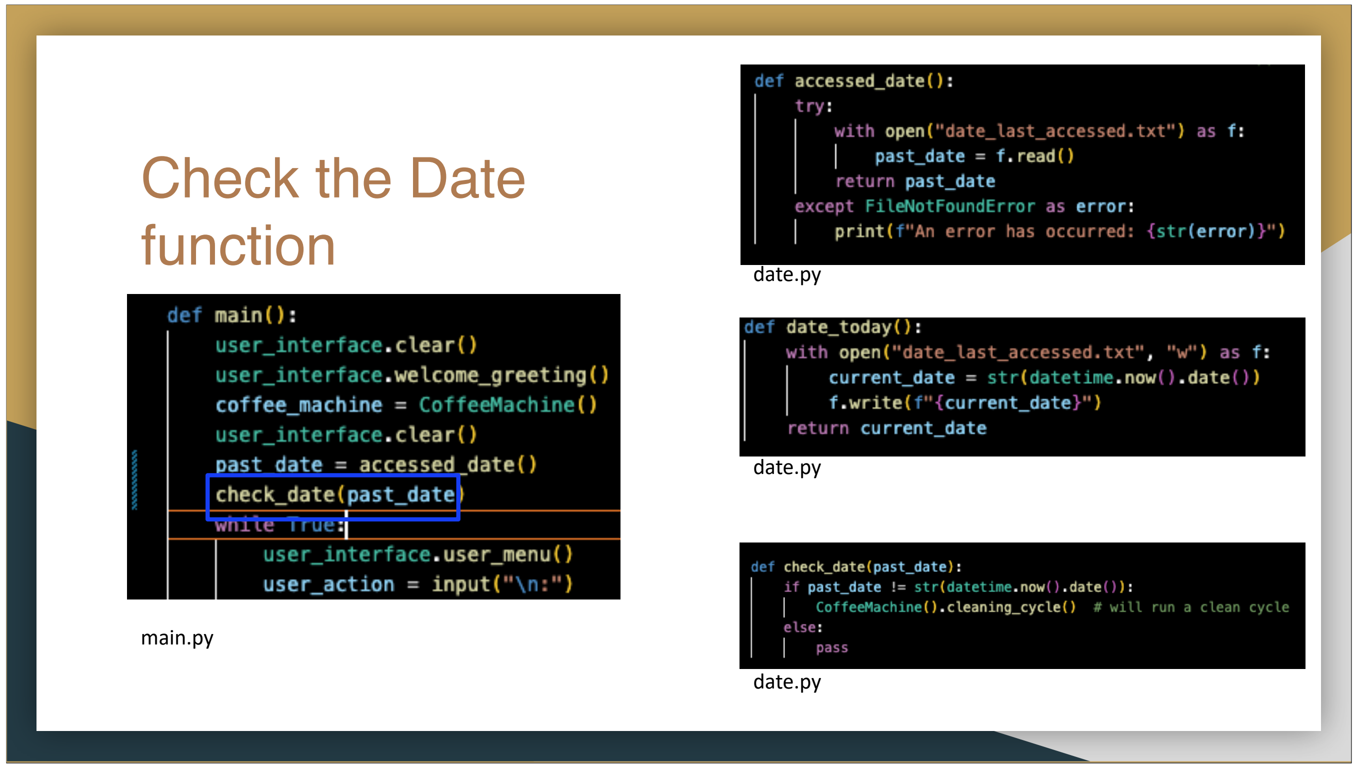
```python
def main():
    user_interface.clear()
    user_interface.welcome_greeting()
    coffee_machine = CoffeeMachine()
    user_interface.clear()
    past_date = accessed_date()
    check_date(past_date)
    while True:
        user_interface.user_menu()
        user_action = input("\n:")
```

Firstly let's discuss the main user interface. Upon initiating the program the terminal screen will clear. This is achieved through the use of the clear() function within the user_interface module, as shown in the top right corner. This function uses the system() function from the os package to clear the terminal screen. It checks the user's operating system (Windows or other) and executes the appropriate command ('cls' for Windows and 'clear' for other systems) to clear the terminal screen accordingly.

After the terminal is cleared the welcome_greeting function is initiated where a 'TURNING ON' message is printed on screen and then after 1 second an ascii banner saying 'COFFEE TIME' appears. After 2 seconds the terminal screen again clears and the user_menu appears with the 5 options the user can select.

The user can select 1 of 5 options as shown in the terminal output box. Depending on their selection the terminal screen will change to run various function which will be discussed. Note that a match case statement is utilised here to match the user's input to the various options.

The sleep function has been achieved by importing the time library. The number within the time.sleep() function indicates how many seconds the screen should hold before changing. The colour that is shown in the terminal window is achieved through importing the colorama package. I utilised this because the application becomes more engaging with colour added.



Check the Date function

```python
def main():
    user_interface.clear()
    user_interface.welcome_greeting()
    coffee_machine = CoffeeMachine()
    user_interface.clear()
    past_date = accessed_date()
    check_date(past_date)
    while True:
        user_interface.user_menu()
        user_action = input("\n:")
```

main.py

```python
def accessed_date():
    try:
        with open("date_last_accessed.txt") as f:
            past_date = f.read()
        return past_date
    except FileNotFoundError as error:
        print(f"An error has occurred: {str(error)}")
```

date.py

```python
def date_today():
    with open("date_last_accessed.txt", "w") as f:
        current_date = str(datetime.now().date())
        f.write(f"{current_date}")
    return current_date
```

date.py

```python
def check_date(past_date):
    if past_date != str(datetime.now().date()):
        CoffeeMachine().cleaning_cycle()  # will run a clean cycle
    else:
        pass
```

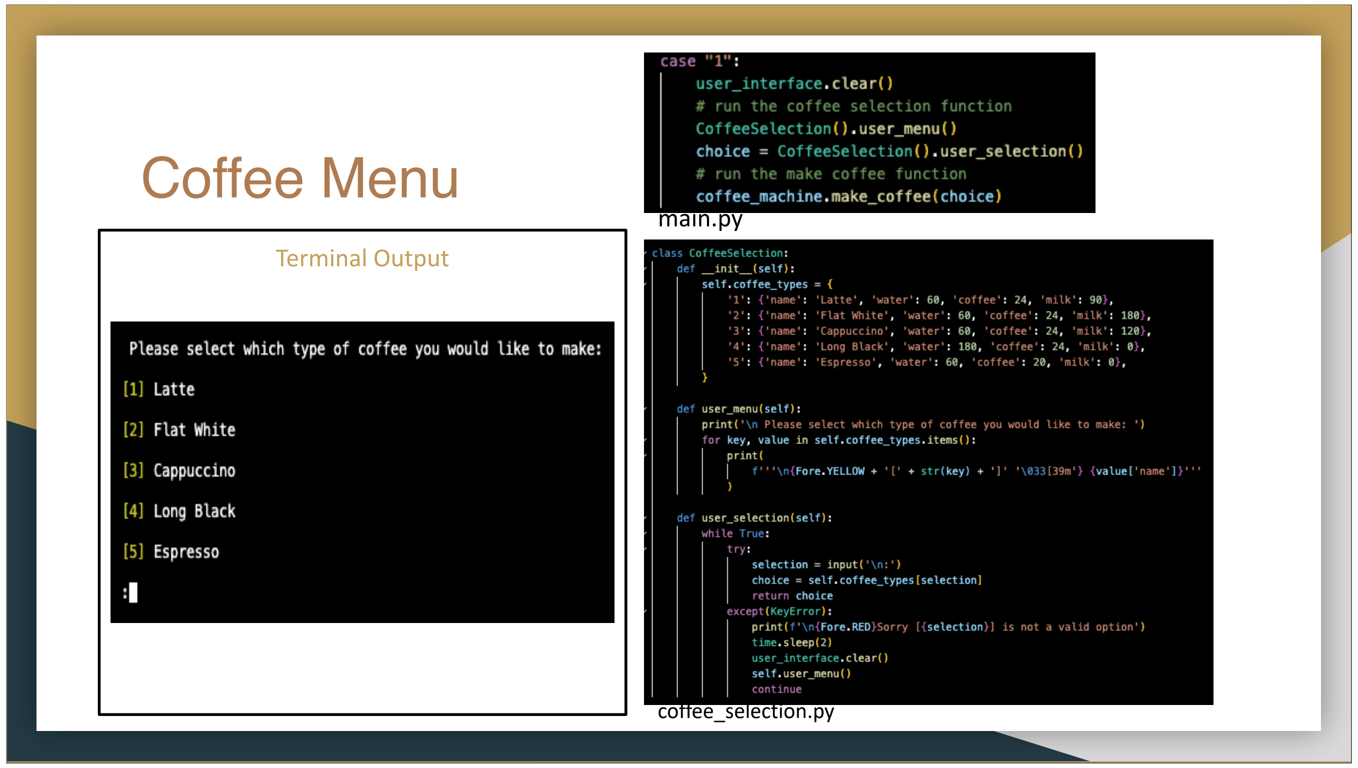
date.py

A feature which occurs behind the scene of the terminal application upon initiating is the check_date function. The purpose of this function is to compare the date that the user is accessing the application to the date the application was last used. If that function finds that the current date is different from the date the application was last accessed it will initiated the cleaning_cycle function within the Coffee Machine class. This feature will be discussed later.

The purpose of this feature is to mimic how modern day coffee machines will initiate a self cleaning function if the machine hasn't been used within a certain amount of time.

The check_date function works alongside two other functions. Firstly the accessed_date function will open and read the date stored within the date_last_accessed.txt file. If there is no date within this file or the date doesn't match the current date then the check_date function will run the cleaning cycle. The date is stored within the txt file through the execution of the date_today function. This function will open the txt file and write in the current date. This date_today function is run in a later stage and will be discussed later.

The datetime library was imported to access the functions relating to dates.



Coffee Menu

Terminal Output

```
 Please select which type of coffee you would like to make:

[1] Latte

[2] Flat White

[3] Cappuccino

[4] Long Black

[5] Espresso

:
```

```python
case "1":
    user_interface.clear()
    # run the coffee selection function
    CoffeeSelection().user_menu()
    choice = CoffeeSelection().user_selection()
    # run the make coffee function
    coffee_machine.make_coffee(choice)
```

main.py

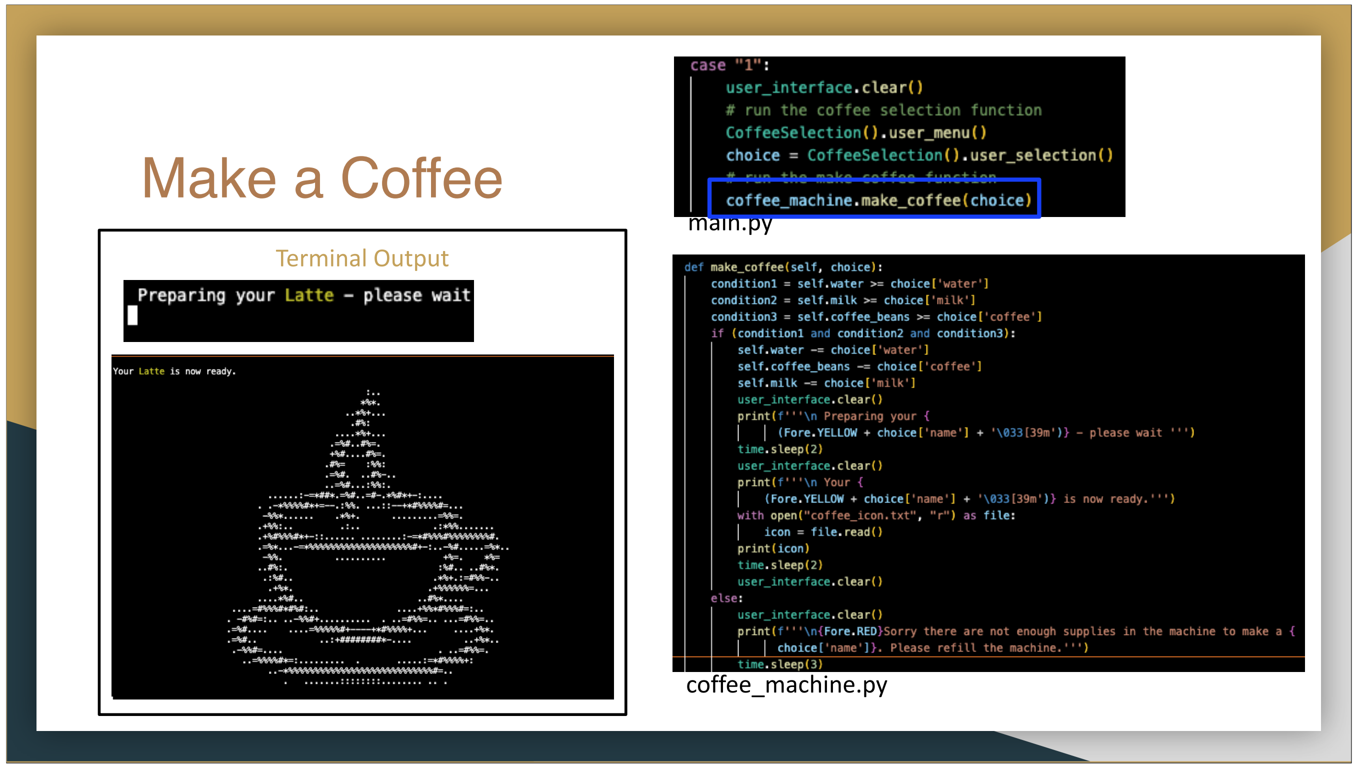
```python
class CoffeeSelection:
    def __init__(self):
        self.coffee_types = {
            '1': {'name': 'Latte', 'water': 60, 'coffee': 24, 'milk': 90},
            '2': {'name': 'Flat White', 'water': 60, 'coffee': 24, 'milk': 180},
            '3': {'name': 'Cappuccino', 'water': 60, 'coffee': 24, 'milk': 120},
            '4': {'name': 'Long Black', 'water': 180, 'coffee': 24, 'milk': 0},
            '5': {'name': 'Espresso', 'water': 60, 'coffee': 20, 'milk': 0},
        }

    def user_menu(self):
        print('\n Please select which type of coffee you would like to make: ')
        for key, value in self.coffee_types.items():
            print(
                f'''\n{Fore.YELLOW + '[' + str(key) + ']' '\033[39m'} {value['name']}'''
            )

    def user_selection(self):
        while True:
            try:
                selection = input('\n:')
                choice = self.coffee_types[selection]
                return choice
            except(KeyError):
                print(f'\n{Fore.RED}Sorry [{selection}] is not a valid option')
                time.sleep(2)
                user_interface.clear()
                self.user_menu()
                continue
```

coffee_selection.py

If the user selects 1 from the main menu it will clear the terminal screen and then display the output as seen in the terminal output box. This menu is prompting the user to select one of five types of coffee. This menu display is achieved through the user of two main features. The CoffeeSelection class is created with the constructor containing a dictionary called coffee_types, which holds the name, water, coffee and milk requirements for each type of coffee. The keys in the dictionary are the numbers 1 through 5 stored as strings. The user_menu function uses a for loop to display the keys (number) and their corresponding name value.

Following the user_menu function being run the user_selection function is then run. The function allows the user to input their choice of coffee. The user must enter a number between 1 and 5 and the user_selection function checks whether the imputed value matched a key within the coffee_types dictionary. If the key is matched then the function will store the corresponding value (name, water, coffee, milk) in the variable of choice which is to be accessed by a future function within the CoffeeMachine class.

Error handling has been used in the user_selection function to ensure that if the user inputs any value other than the numbers between 1 & 5 then an error message will be displayed and they will be prompted to try again.



Make a Coffee

Terminal Output

```
 Preparing your Latte - please wait
```

```
Your Latte is now ready.
```

```python
case "1":
    user_interface.clear()
    # run the coffee selection function
    CoffeeSelection().user_menu()
    choice = CoffeeSelection().user_selection()
    # run the make coffee function
    coffee_machine.make_coffee(choice)
```

main.py

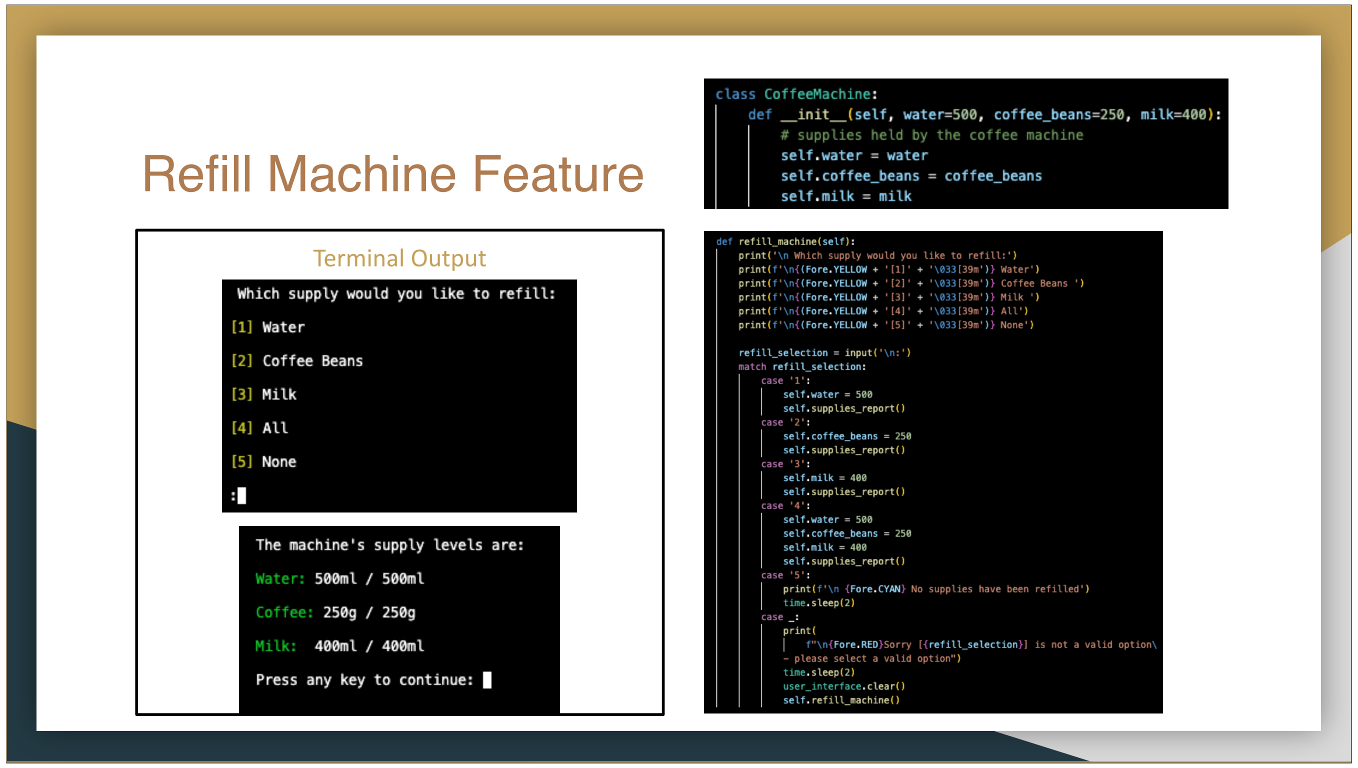
```python
def make_coffee(self, choice):
    condition1 = self.water >= choice['water']
    condition2 = self.milk >= choice['milk']
    condition3 = self.coffee_beans >= choice['coffee']
    if (condition1 and condition2 and condition3):
        self.water -= choice['water']
        self.coffee_beans -= choice['coffee']
        self.milk -= choice['milk']
        user_interface.clear()
        print(f'''\n Preparing your {
                (Fore.YELLOW + choice['name'] + '\033[39m')} - please wait ''')
        time.sleep(2)
        user_interface.clear()
        print(f'''\n Your {
            (Fore.YELLOW + choice['name'] + '\033[39m')} is now ready.''')
        with open("coffee_icon.txt", "r") as file:
            icon = file.read()
        print(icon)
        time.sleep(2)
        user_interface.clear()
    else:
        user_interface.clear()
        print(f'''\n{Fore.RED}Sorry there are not enough supplies in the machine to make a {
                choice['name']}. Please refill the machine.''')
        time.sleep(3)
```

coffee_machine.py

Continuing on from the coffee menu, if the user selects 1 from the main menu a third and final function is run. This is the make_coffee function within the CoffeeMachine class. As mentioned in the previous slide the user's selection of coffee has been used to store the corresponding value dictionary as the variable choice. So within this dictionary is the name, amount of water, milk and coffee required to make the coffee selection.

The make_coffee function begins by utilising an if statement to comparing each of the water, coffee and milk requirements for the coffee selection to the machine's own water, coffee and milk supplies. If any of the coffee's requirements are higher than any of the machine's supply levels then the if statement is passed and the else statement is triggered notifying the user that there are insufficient supplies and returns them to the main menu. However, if the supply levels are sufficient then the if statement proceeds to minus the amount of supplies needed to fufill the requirements from the machine's supplies and then processed to print out the view as seen within the terminal output box. The coffee cup art is displayed for 2 seconds before the terminal screen is cleared and the user is returned to the main menu.

Note that the coffee cup art is stored with the coffee_icon text file and is read and printed with the if statement.



Refill Machine Feature

Terminal Output

```
Which supply would you like to refill:

[1] Water

[2] Coffee Beans

[3] Milk

[4] All

[5] None

:
```

```
The machine's supply levels are:

Water: 500ml / 500ml

Coffee: 250g / 250g

Milk:  400ml / 400ml

Press any key to continue:
```

```python
class CoffeeMachine:
    def __init__(self, water=500, coffee_beans=250, milk=400):
        # supplies held by the coffee machine
        self.water = water
        self.coffee_beans = coffee_beans
        self.milk = milk
```

```python
def refill_machine(self):
    print('\n Which supply would you like to refill:')
    print(f'\n{(Fore.YELLOW + '[1]' + '\033[39m')} Water')
    print(f'\n{(Fore.YELLOW + '[2]' + '\033[39m')} Coffee Beans ')
    print(f'\n{(Fore.YELLOW + '[3]' + '\033[39m')} Milk ')
    print(f'\n{(Fore.YELLOW + '[4]' + '\033[39m')} All')
    print(f'\n{(Fore.YELLOW + '[5]' + '\033[39m')} None')

    refill_selection = input('\n:')
    match refill_selection:
        case '1':
            self.water = 500
            self.supplies_report()
        case '2':
            self.coffee_beans = 250
            self.supplies_report()
        case '3':
            self.milk = 400
            self.supplies_report()
        case '4':
            self.water = 500
            self.coffee_beans = 250
            self.milk = 400
            self.supplies_report()
        case '5':
            print(f'\n {Fore.CYAN} No supplies have been refilled')
            time.sleep(2)
        case _:
            print(
                f"\n{Fore.RED}Sorry [{refill_selection}] is not a valid option\
            - please select a valid option")
            time.sleep(2)
            user_interface.clear()
            self.refill_machine()
```

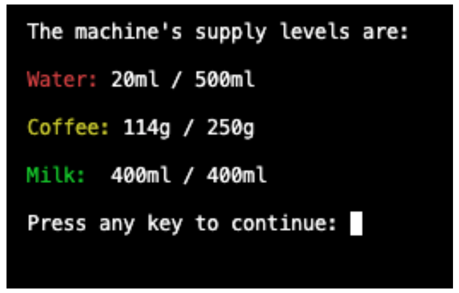
If the user selects 2 from the main menu the refill_machine function within the CoffeeMachine class will be executed. This feature allows the user select which supply they would like to refill. They can choose individual supplies or refill all at once. The user also has the ability to select none which will return them to the main menu.

This feature works through the use of a match case statement. Where follow the printing of the user's options the user is prompted to select an option. Depending on the user's input the corresponding supply will be refilled to the amount as set in the class's constructor. Each case between 1 and 4 will also trigger the supplies_report function which will be discussed next. But in summary this report just shows the machine's current supply levels.

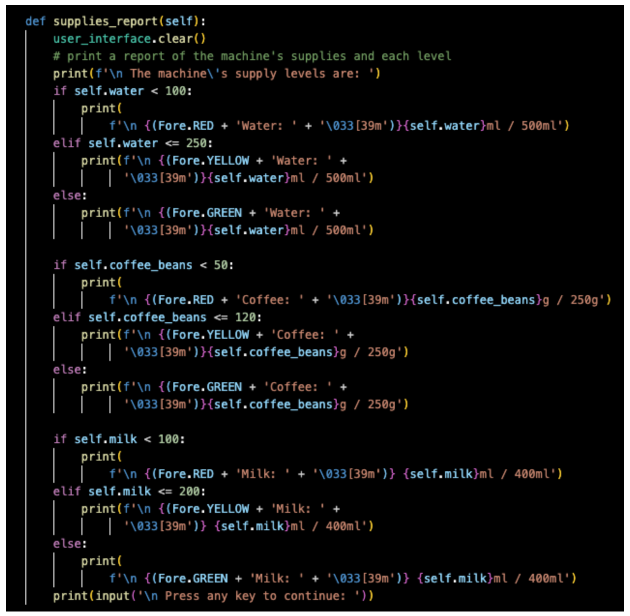
If the user's input doesn't match any case then through the use of error handling an error statement is displayed and the user is prompted to select a valid option.

Supply Report

Terminal Output



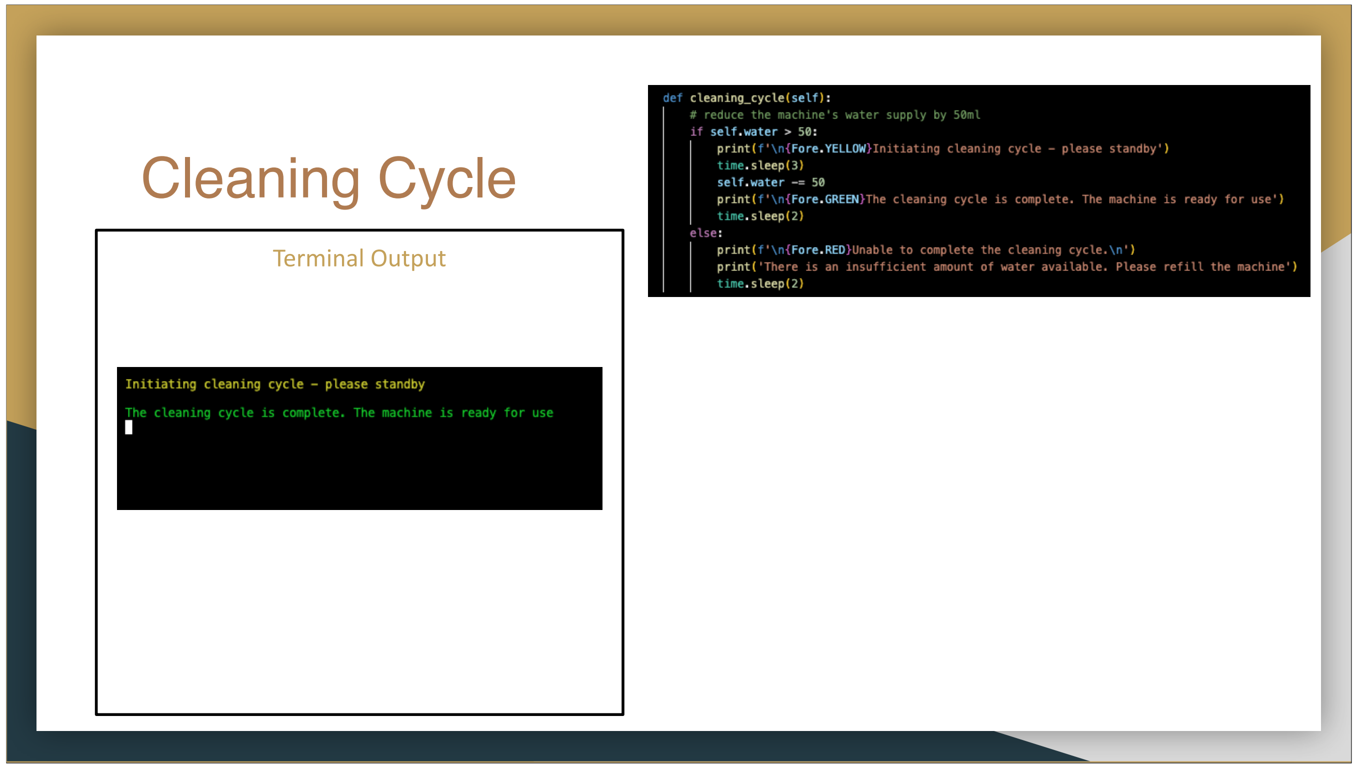
```
The machine's supply levels are:

Water: 20ml / 500ml

Coffee: 114g / 250g

Milk:  400ml / 400ml

Press any key to continue: 
```



```python
def supplies_report(self):
    user_interface.clear()
    # print a report of the machine's supplies and each level
    print(f'\n The machine\'s supply levels are: ')
    if self.water < 100:
        print(
            f'\n {(Fore.RED + 'Water: ' + '\033[39m')}{self.water}ml / 500ml')
    elif self.water <= 250:
        print(f'\n {(Fore.YELLOW + 'Water: ' +
                '\033[39m')}{self.water}ml / 500ml')
    else:
        print(f'\n {(Fore.GREEN + 'Water: ' +
                '\033[39m')}{self.water}ml / 500ml')

    if self.coffee_beans < 50:
        print(
            f'\n {(Fore.RED + 'Coffee: ' + '\033[39m')}{self.coffee_beans}g / 250g')
    elif self.coffee_beans <= 120:
        print(f'\n {(Fore.YELLOW + 'Coffee: ' +
                '\033[39m')}{self.coffee_beans}g / 250g')
    else:
        print(f'\n {(Fore.GREEN + 'Coffee: ' +
                '\033[39m')}{self.coffee_beans}g / 250g')

    if self.milk < 100:
        print(
            f'\n {(Fore.RED + 'Milk: ' + '\033[39m')} {self.milk}ml / 400ml')
    elif self.milk <= 200:
        print(f'\n {(Fore.YELLOW + 'Milk: ' +
                '\033[39m')} {self.milk}ml / 400ml')
    else:
        print(
            f'\n {(Fore.GREEN + 'Milk: ' + '\033[39m')} {self.milk}ml / 400ml')
    print(input('\n Press any key to continue: '))
```

If the user selects 3 from the main menu or they match options 1-4 in the refill function then the supply_report function within the CoffeeMachine class will be executed. The purpose of this feature is to allow the user to see the level of each of the machine's supplies. I utilised the colorama package to indicate the severity of each supply's level. Red indicates the supply is very low, yellow indicates the supply is reaching a lower state of full and green means the supply is close to full. I also added a 'press any key to continue' input to prompt the user that if they have finished looking at the report to press any key and it will take them back to the main menu. I added this prompt instead of a sleep timer because it was hard to judge at what speed a user will read this report.

If time had permitted I would have liked to have added the ability to export this report into a txt file.



Cleaning Cycle

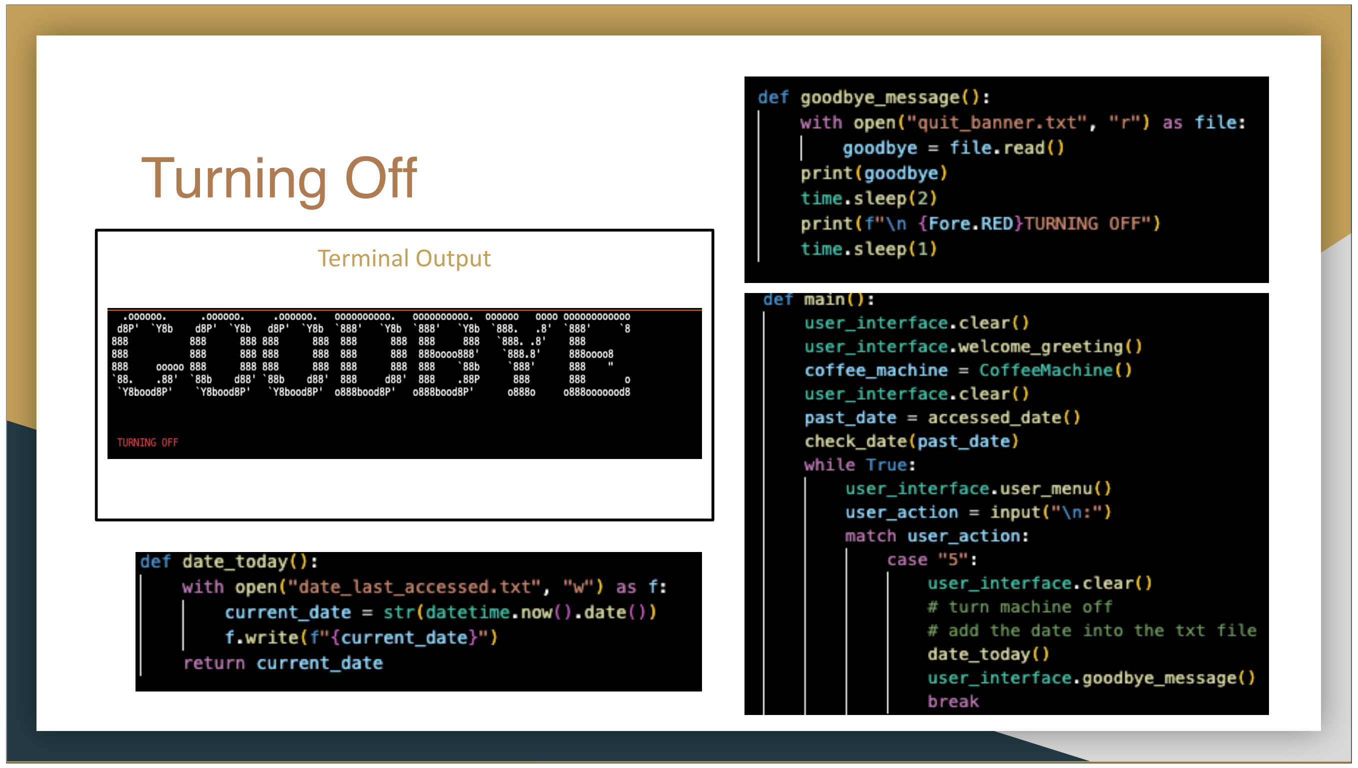
Terminal Output

```
Initiating cleaning cycle - please standby

The cleaning cycle is complete. The machine is ready for use
```

```python
def cleaning_cycle(self):
    # reduce the machine's water supply by 50ml
    if self.water > 50:
        print(f'\n{Fore.YELLOW}Initiating cleaning cycle - please standby')
        time.sleep(3)
        self.water -= 50
        print(f'\n{Fore.GREEN}The cleaning cycle is complete. The machine is ready for use')
        time.sleep(2)
    else:
        print(f'\n{Fore.RED}Unable to complete the cleaning cycle.\n')
        print('There is an insufficient amount of water available. Please refill the machine')
        time.sleep(2)
```

As previously mentioned the application allows the user to run a cleaning cycle. If the user selects 4 from the main menu or the check_date function finds that the current date is different from the date the application was last accessed, then the cleaning_cycle function within the CoffeeMachine class will be executed.

The cleaning cycle feature utilises an if/else statement and determines that if the machine's water supply is greater than 50 then the cleaning cycle can proceed however if the machine's water supply is less than 50 then an error message is displayed to the user and returns them to the main menu. If the cleaning cycle does proceed then an initiating cleaning cycle message is printed on the terminal screen followed by a cleaning cycle completed message 3 seconds later. Also if the cleaning cycle does proceed then the water supply is reduced by 50.



Turning Off

Terminal Output

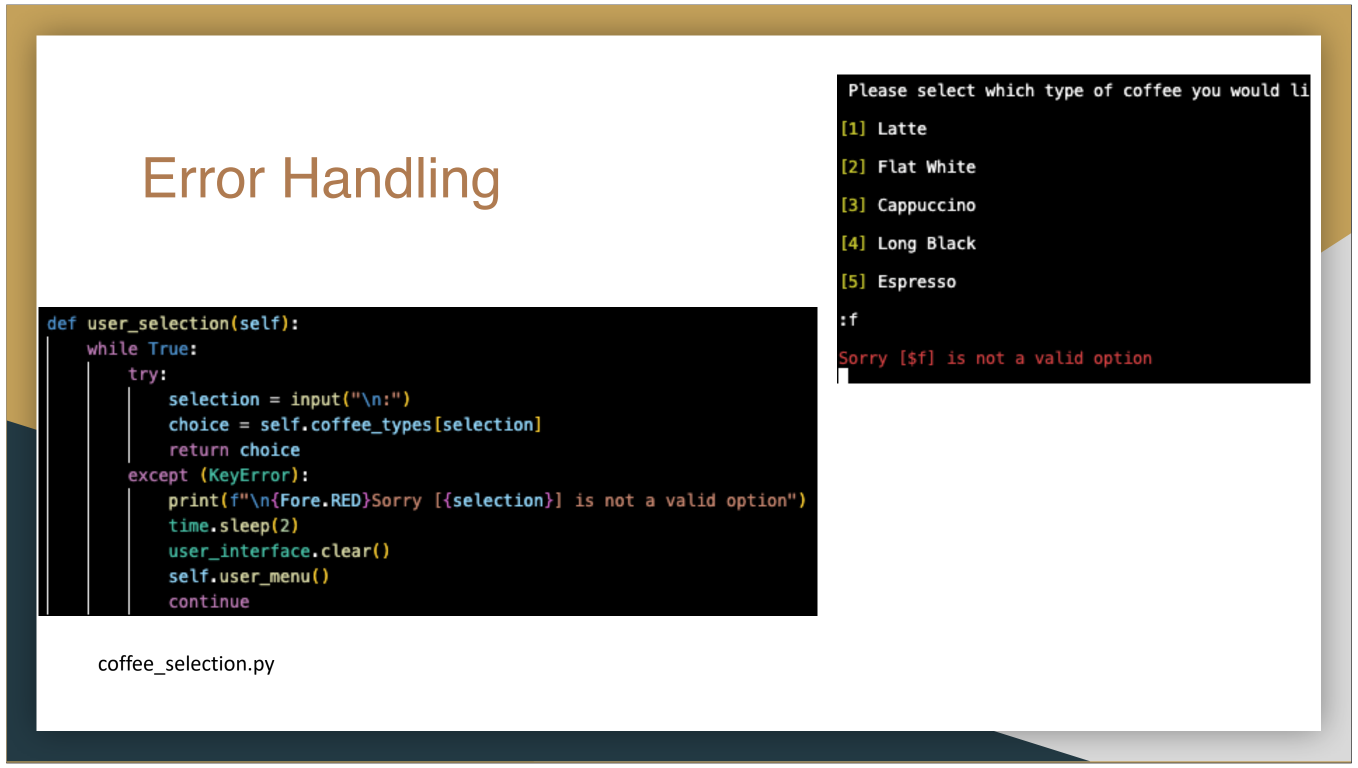
```
  .oooooo.      .oooooo.     .oooooo.   oooooooooo.   oooooooooo.  oooooo   oooo oooooooooooo
 d8P'  `Y8b    d8P'  `Y8b   d8P'  `Y8b  `888'   `Y8b  `888'   `Y8b  `888.   .8'  `888'     `8
888           888      888 888      888  888      888  888     888   `888. .8'    888
888           888      888 888      888  888      888  888oooo888'    `888.8'     888oooo8
888     ooooo 888      888 888      888  888      888  888    `88b     `888'      888    "
`88.    .88'  `88b    d88' `88b    d88'  888     d88'  888    .88P      888       888       o
 `Y8bood8P'    `Y8bood8P'   `Y8bood8P'  o888bood8P'   o888bood8P'      o888o     o888ooooood8


TURNING OFF
```

```python
def date_today():
    with open("date_last_accessed.txt", "w") as f:
        current_date = str(datetime.now().date())
        f.write(f"{current_date}")
    return current_date
```

```python
def goodbye_message():
    with open("quit_banner.txt", "r") as file:
        goodbye = file.read()
    print(goodbye)
    time.sleep(2)
    print(f"\n {Fore.RED}TURNING OFF")
    time.sleep(1)
```

```python
def main():
    user_interface.clear()
    user_interface.welcome_greeting()
    coffee_machine = CoffeeMachine()
    user_interface.clear()
    past_date = accessed_date()
    check_date(past_date)
    while True:
        user_interface.user_menu()
        user_action = input("\n:")
        match user_action:
            case "5":
                user_interface.clear()
                # turn machine off
                # add the date into the txt file
                date_today()
                user_interface.goodbye_message()
                break
```

Finally the application features a turning off function. From the main menu if the user enters 5 the while statement nesting the match case statement will break, thus exiting the program. Prior to the break occurring the terminal screen is cleared and two functions are run. Firstly the date_today function is executed. As previously mentioned this function will open the date_last_accessed.txt file and write in the current date that the program is being accessed. This date is then utilised in the check_date function if the application is then accessed at a later date. And finally the goodbye_message function is executed. An ascii banner saying 'GOODBYE' appears for 2 seconds after reading the quit_banner.txt file, and then a red 'TURNING OFF' message is displayed for 1 second prior to the program exiting.



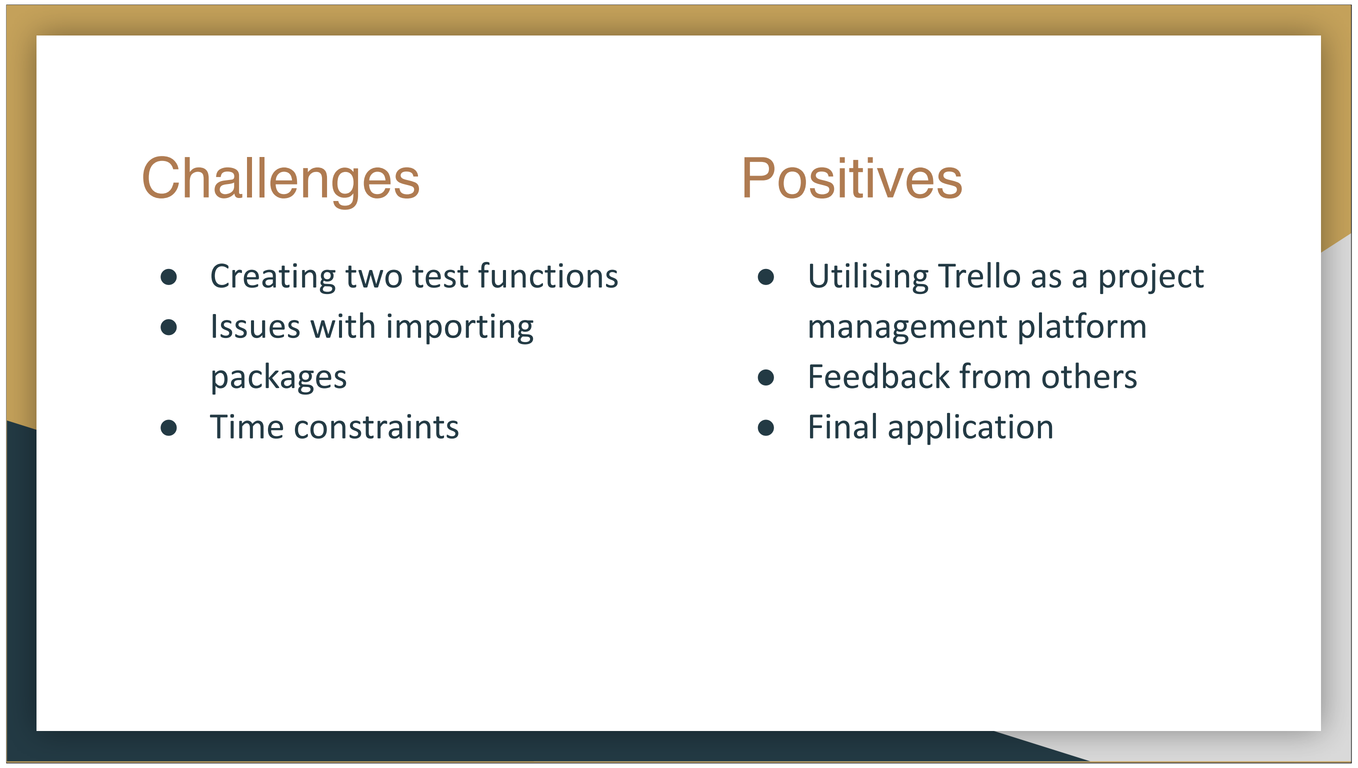
Error Handling

```python
def user_selection(self):
    while True:
        try:
            selection = input("\n:")
            choice = self.coffee_types[selection]
            return choice
        except (KeyError):
            print(f"\n{Fore.RED}Sorry [{selection}] is not a valid option")
            time.sleep(2)
            user_interface.clear()
            self.user_menu()
            continue
```

coffee_selection.py

```
 Please select which type of coffee you would li
[1] Latte
[2] Flat White
[3] Cappuccino
[4] Long Black
[5] Espresso
:f
Sorry [$f] is not a valid option
```

There are plenty of examples of error handling within the terminal application code. I have provided an examples on the screen as well as the output which occurs if the function encounters an error. I utilised a try-except block within the coffee_selection module. Within this function if the user enters a value which does not match a key within the dictionary then a KeyError is raised and the function handles it by printing an appropriate error message.



Challenges

- Creating two test functions
- Issues with importing packages
- Time constraints

Positives

- Utilising Trello as a project management platform
- Feedback from others
- Final application

One of the main challenges I faced during the development of the application was during the creation of the two test functions. I used PYTEST to complete the unit testing and it was a very new concept to me so it involved a lot of research and practice, but I do feel like I now have a better understanding of unit testing and it's importance. The second issue I had was importing packages. There were a lot of packages I installed from PyPi that didn't seem be recognised when I attempted to import them. So I found myself doing a lot of research into packages other people where using for certain python features and using more recently updated packages. I am still unsure if it was something to do with my virtual environment or not but the packages I did utilised ended up being sufficient for the project. Lastly, a final challenge I faced was time constraints. Some functions took a greater amount of time that anticipated, so some features I wish I could have implemented simply didn't get done. For example, one of the educators suggested that if time permitted I could add a feature that allows setting of the cost of each type of coffee and a sale price, then on demand, have the app show a balance sheet where it shows how many sales of each there were, and calculates the total profit for each. This would have been a great feature but unfortunately time did not permit, but it is ok because I feel like the app still works as intended.

A few positives from the project was getting to use a project management platform for the first time. I utilised Trello to create a project board and I found it extremely helpful for time management. Also I allowed some friends to test out the application and their feedback was great, it was very rewarding to see how far I have come for only having started to learn python a few weeks ago.

THANK YOU