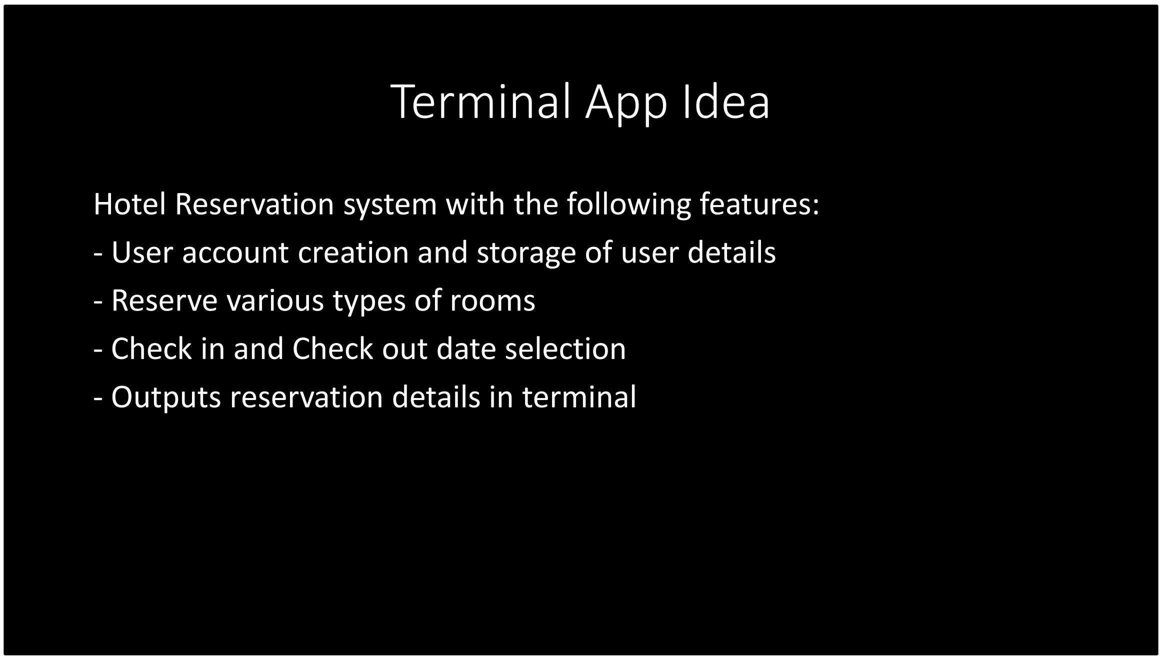
T1A3 Assessment

Terminal Application – Jonathan Ow

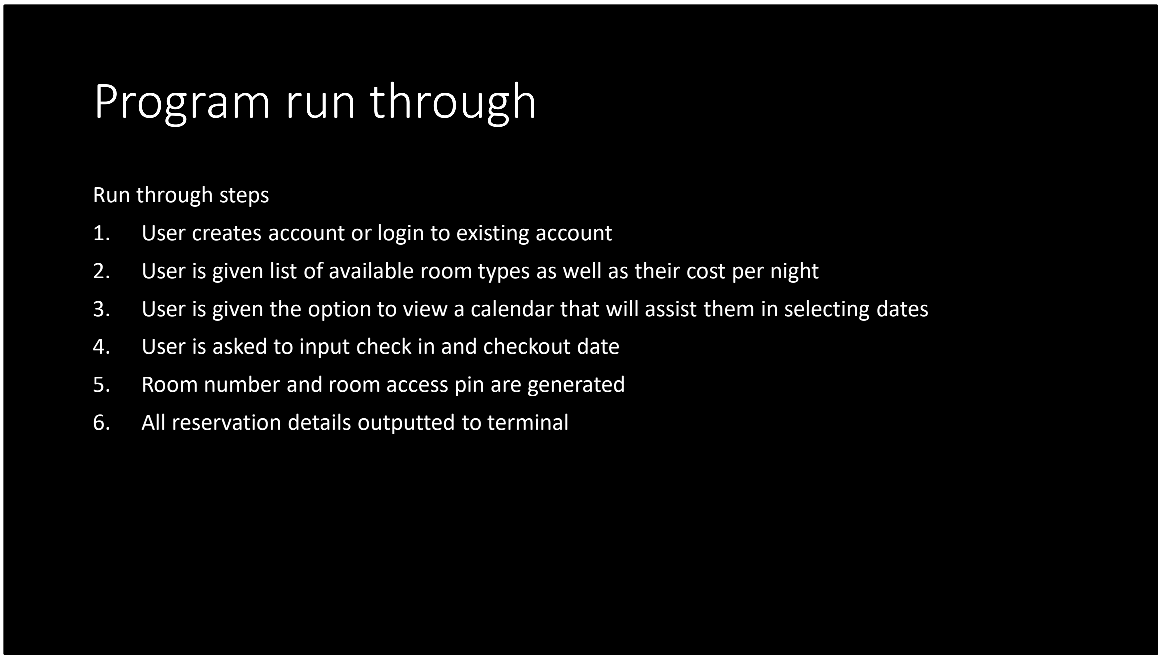


Terminal App Idea

Hotel Reservation system with the following features:
- User account creation and storage of user details
- Reserve various types of rooms
- Check in and Check out date selection
- Outputs reservation details in terminal

The Idea for the terminal application was a Hotel Reservation system which would enable users to book a hotel room of their choice. This (fictional) hotel, Atlas, provides guests with a variety of rooms which suit different needs and budgets. I have chosen this name as it is based off the mythological Greek titan Atlas which is said to be the one holding the earth on his shoulders. This fits with the philosophy of the hotel which aims to accommodate the needs of all individuals throughout the globe. The main features that the terminal app contains are firstly the user account creation system which ensures that users looking to reserve a room must first create or log in to an account ensuring that each reservation is assigned to a unique individual. Room types are varied, and the program then gives the user room options with varying prices depending on room type and then gives the user the ability to select how long they intend to stay in said room for by inputting a check in and a check out date. Once all required information is obtained the user receives the details of their booking via the output terminal.

To implement these features, I have chosen to separate them into functions which are name clearly ensuring that anyone who views my code will understand what its purpose is. Once a function is carried out it will call the function that proceeds and until the entire operation is complete.



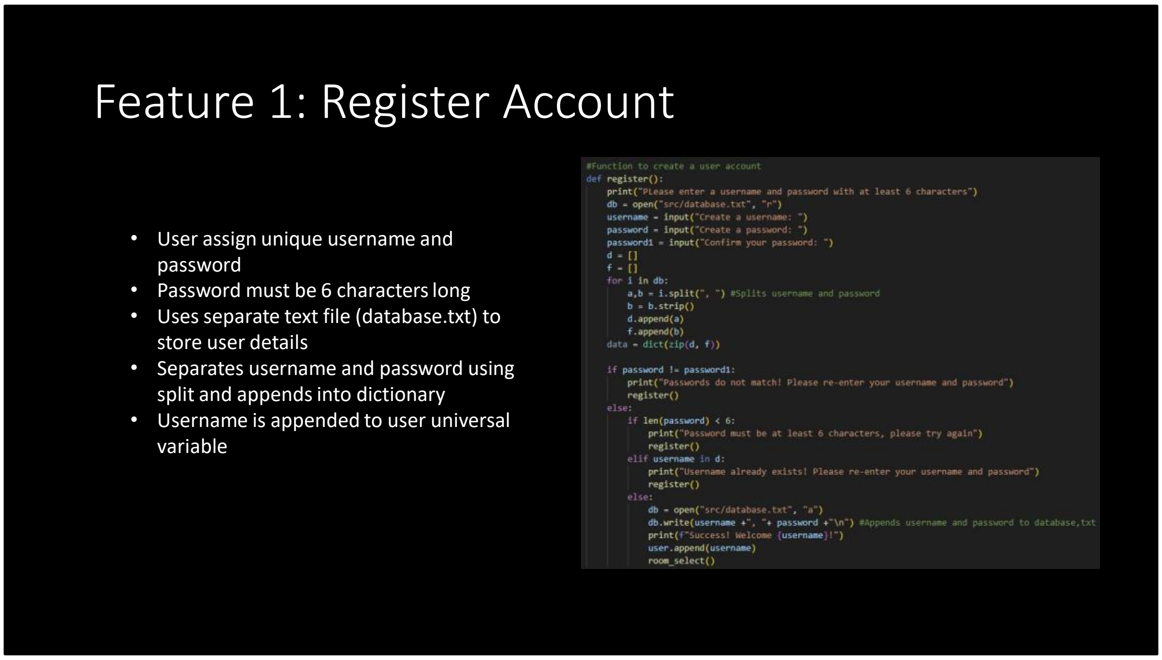
Program run through

Run through steps

1. User creates account or login to existing account
2. User is given list of available room types as well as their cost per night
3. User is given the option to view a calendar that will assist them in selecting dates
4. User is asked to input check in and checkout date
5. Room number and room access pin are generated
6. All reservation details outputted to terminal

The general run through of the terminal program goes as follows:

1. The user is first taken to the home function and given a prompt to either login or create an account. This feature is split into two functions which handles both options and enables the user to create their own username and password which is then verified
2. A list of rooms is then given to the user to select and gives details such as room type description and cost per night
3. An option to view a calendar is given to the user that will display a calendar for the month and year they want to view, this will assist the user in selecting their reservation dates
4. The user will then be prompted to enter a check-in and checkout date which is like the user account feature is split into two functions
5. Once the two dates have been confirmed, a 2-digit room number and 4-digit room access pin will be generated (invisibly)
6. The following reservation details will be outputted to the terminal: username, room type, room number, check-in date, checkout date and cost per night.

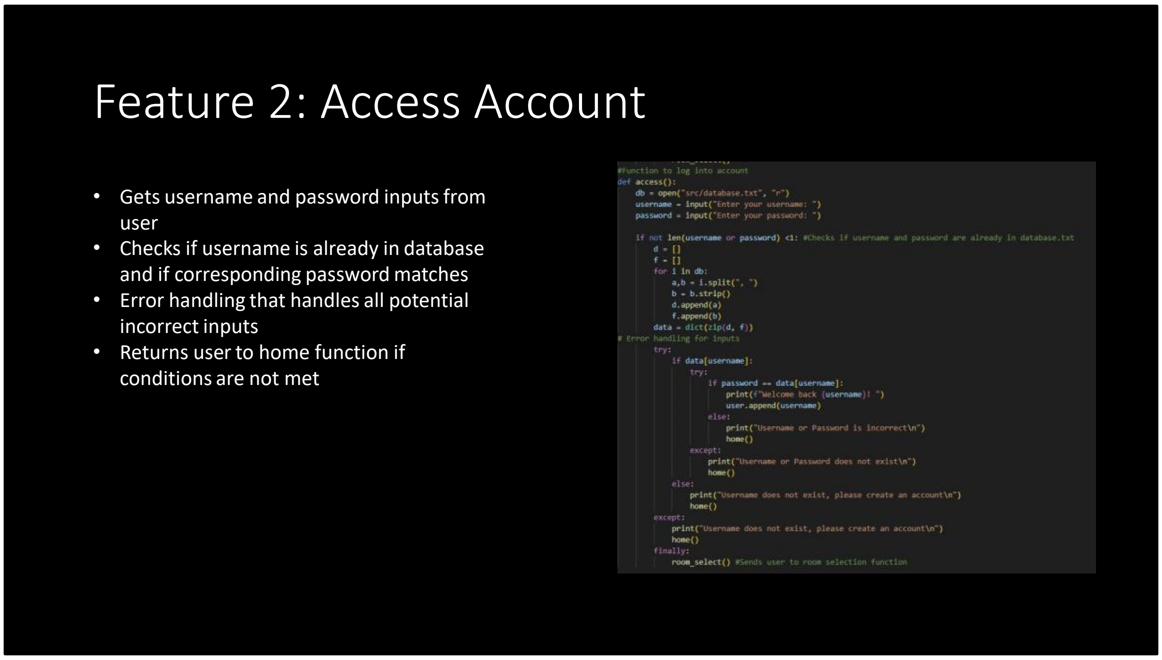


Feature 1: Register Account

- User assign unique username and password
- Password must be 6 characters long
- Uses separate text file (database.txt) to store user details
- Separates username and password using split and appends into dictionary
- Username is appended to user universal variable

```python
#Function to create a user account
def register():
    print("PLease enter a username and password with at least 6 characters")
    db = open("src/database.txt", "r")
    username = input("Create a username: ")
    password = input("Create a password: ")
    password1 = input("Confirm your password: ")
    d = []
    f = []
    for i in db:
        a,b = i.split(", ") #Splits username and password
        b = b.strip()
        d.append(a)
        f.append(b)
    data = dict(zip(d, f))

    if password != password1:
        print("Passwords do not match! Please re-enter your username and password")
        register()
    else:
        if len(password) < 6:
            print("Password must be at least 6 characters, please try again")
            register()
        elif username in d:
            print("Username already exists! Please re-enter your username and password")
            register()
        else:
            db = open("src/database.txt", "a")
            db.write(username +", "+ password +"\n") #Appends username and password to database,txt
            print(f"Success! Welcome {username}!")
            user.append(username)
            room_select()
```

In order to begin the reservation process, users must first create an account using the register function. The function takes 2 inputs, username and password which are then stored in the database.txt file. The split method is used to separate both inputs so that they can be accessed later. For security purposes, all passwords must be at least 6 characters long and an if statement ensures that if input does not match the requirements the user will be taken back to the Home function to restart the process. If the username the user has inputted already exists in the database, a prompt will appear telling the user and the process will restart. If both username and password meet the requirements, they will be appended to the database.txt file and the username will be appended to the user universal variable to be printed out at the end of the reservation process.

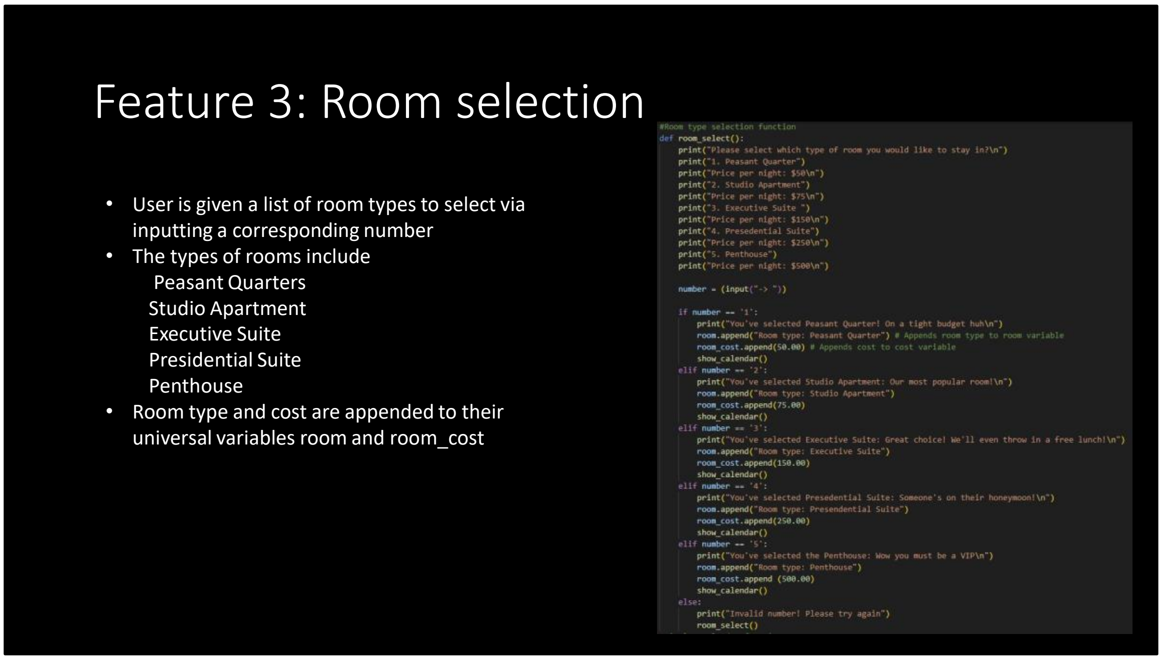


Feature 2: Access Account

- Gets username and password inputs from user
- Checks if username is already in database and if corresponding password matches
- Error handling that handles all potential incorrect inputs
- Returns user to home function if conditions are not met

```python
#Function to log into account
def access():
    db = open("src/database.txt", "r")
    username = input("Enter your username: ")
    password = input("Enter your password: ")

    if not len(username or password) <1: #Checks if username and password are already in database.txt
        d = []
        f = []
        for i in db:
            a,b = i.split(", ")
            b = b.strip()
            d.append(a)
            f.append(b)
        data = dict(zip(d, f))
# Error handling for inputs
        try:
            if data[username]:
                try:
                    if password == data[username]:
                        print(f"Welcome back {username}! ")
                        user.append(username)
                    else:
                        print("Username or Password is incorrect\n")
                        home()
                except:
                    print("Username or Password does not exist\n")
                    home()
            else:
                print("Username does not exist, please create an account\n")
                home()
        except:
            print("Username does not exist, please create an account\n")
            home()
        finally:
            room_select() #Sends user to room selection function
```

If the user has created an account in the past, they are able to re-access it by using the access function. This function accepts 2 inputs, one for username and one for password. The function will then search the database file and check whether or not the username exists and if the password that has been entered matches it. If not, the error handling statements account for all potential scenarios where either username or password is incorrect and will then prompt the user to either create an account or try again. I have chosen to separate this from the account creation function as it keep the overall code DRY and ensures separation of concerns is maintained

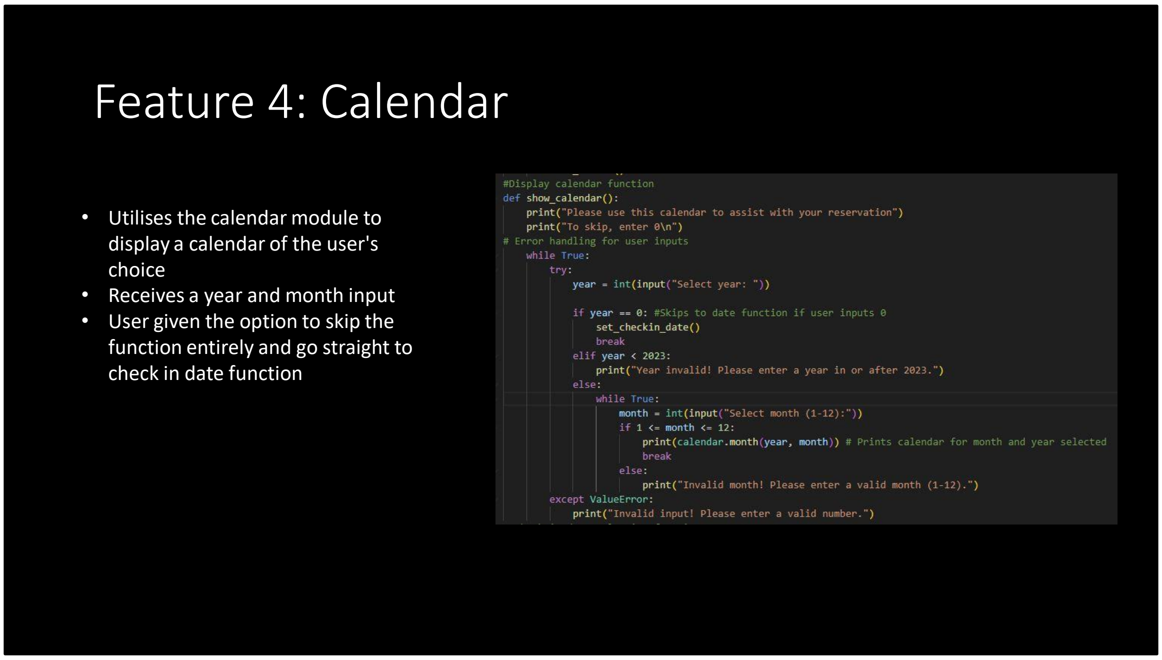


Feature 3: Room selection

- User is given a list of room types to select via inputting a corresponding number
- The types of rooms include
  - Peasant Quarters
  - Studio Apartment
  - Executive Suite
  - Presidential Suite
  - Penthouse
- Room type and cost are appended to their universal variables room and room_cost

```python
#Room type selection function
def room_select():
    print("Please select which type of room you would like to stay in?\n")
    print("1. Peasant Quarter")
    print("Price per night: $50\n")
    print("2. Studio Apartment")
    print("Price per night: $75\n")
    print("3. Executive Suite ")
    print("Price per night: $150\n")
    print("4. Presedential Suite")
    print("Price per night: $250\n")
    print("5. Penthouse")
    print("Price per night: $500\n")

    number = (input("-> "))

    if number == '1':
        print("You've selected Peasant Quarter! On a tight budget huh\n")
        room.append("Room type: Peasant Quarter") # Appends room type to room variable
        room_cost.append(50.00) # Appends cost to cost variable
        show_calendar()
    elif number == '2':
        print("You've selected Studio Apartment: Our most popular room!\n")
        room.append("Room type: Studio Apartment")
        room_cost.append(75.00)
        show_calendar()
    elif number == '3':
        print("You've selected Executive Suite: Great choice! We'll even throw in a free lunch!\n")
        room.append("Room type: Executive Suite")
        room_cost.append(150.00)
        show_calendar()
    elif number == '4':
        print("You've selected Presedential Suite: Someone's on their honeymoon!\n")
        room.append("Room type: Presendential Suite")
        room_cost.append(250.00)
        show_calendar()
    elif number == '5':
        print("You've selected the Penthouse: Wow you must be a VIP\n")
        room.append("Room type: Penthouse")
        room_cost.append (500.00)
        show_calendar()
    else:
        print("Invalid number! Please try again")
        room_select()
```

Once the program has confirmed the user has entered correct login details, the function room select will be called. The function displays the list of rooms that are available to be reserved by the user including their type and price per night. The user then is asked to input the corresponding number which will select the desired room. Once selected the program will append the room type to the variable room and the cost to the variable. Each condition statement will handle each input as well as incorrect inputs which will ask the user to re-enter a correct number.

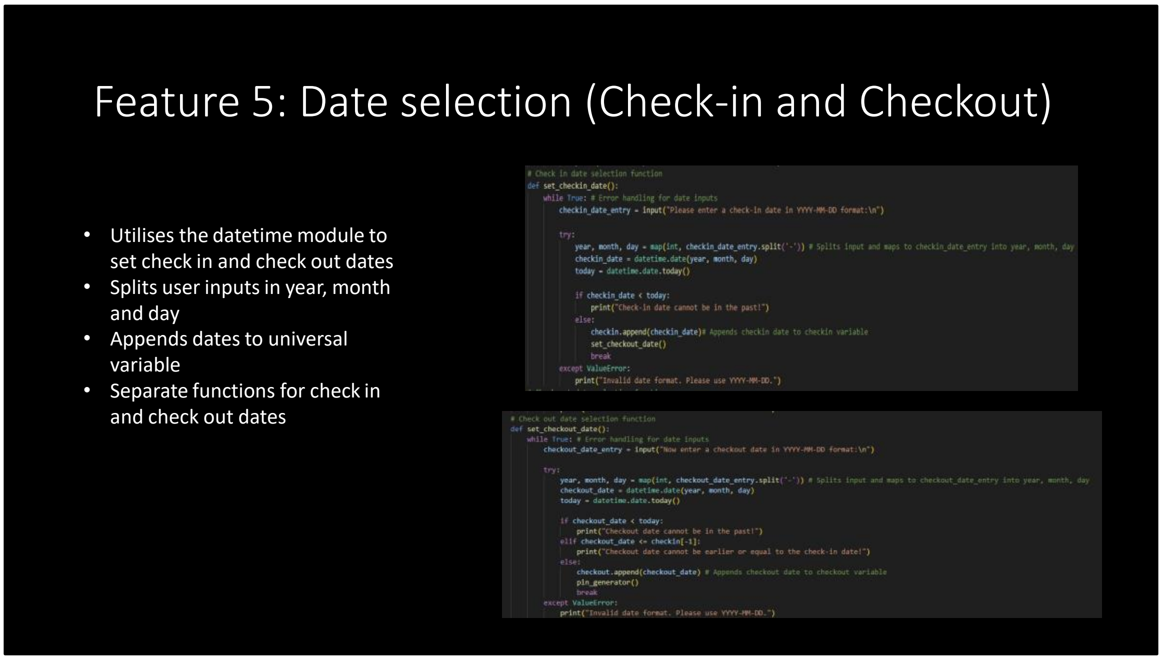


Feature 4: Calendar

- Utilises the calendar module to display a calendar of the user's choice
- Receives a year and month input
- User given the option to skip the function entirely and go straight to check in date function

```python
#Display calendar function
def show_calendar():
    print("Please use this calendar to assist with your reservation")
    print("To skip, enter 0\n")
# Error handling for user inputs
    while True:
        try:
            year = int(input("Select year: "))

            if year == 0: #Skips to date function if user inputs 0
                set_checkin_date()
                break
            elif year < 2023:
                print("Year invalid! Please enter a year in or after 2023.")
            else:
                while True:
                    month = int(input("Select month (1-12):"))
                    if 1 <= month <= 12:
                        print(calendar.month(year, month)) # Prints calendar for month and year selected
                        break
                    else:
                        print("Invalid month! Please enter a valid month (1-12).")
        except ValueError:
            print("Invalid input! Please enter a valid number.")
```

To assist the user with selecting their reservation dates, I decided to utilise the calendar module which outputs a calendar of the user's choice. The function takes in 2 inputs, a year and month. The main part of the function is the error handling which ensures that the year selected user from the current year onwards and that the month input is between1 (January) and 12 (December) If an incorrect value is entered the function will tell the user. Because this function is not an essential component of the reservation system, the user has the option to skip straight to the check-in function by entering 0 in the terminal.



Feature 5: Date selection (Check-in and Checkout)

- Utilises the datetime module to set check in and check out dates
- Splits user inputs in year, month and day
- Appends dates to universal variable
- Separate functions for check in and check out dates

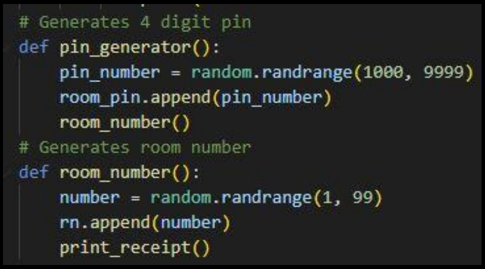
```python
# Check in date selection function
def set_checkin_date():
    while True: # Error handling for date inputs
        checkin_date_entry = input("Please enter a check-in date in YYYY-MM-DD format:\n")

        try:
            year, month, day = map(int, checkin_date_entry.split('-')) # Splits input and maps to checkin_date_entry into year, month, day
            checkin_date = datetime.date(year, month, day)
            today = datetime.date.today()

            if checkin_date < today:
                print("Check-in date cannot be in the past!")
            else:
                checkin.append(checkin_date)# Appends checkin date to checkin variable
                set_checkout_date()
                break
        except ValueError:
            print("Invalid date format. Please use YYYY-MM-DD.")
```

```python
# Check out date selection function
def set_checkout_date():
    while True: # Error handling for date inputs
        checkout_date_entry = input("Now enter a checkout date in YYYY-MM-DD format:\n")

        try:
            year, month, day = map(int, checkout_date_entry.split('-')) # Splits input and maps to checkout_date_entry into year, month, day
            checkout_date = datetime.date(year, month, day)
            today = datetime.date.today()

            if checkout_date < today:
                print("Checkout date cannot be in the past!")
            elif checkout_date <= checkin[-1]:
                print("Checkout date cannot be earlier or equal to the check-in date!")
            else:
                checkout.append(checkout_date) # Appends checkout date to checkout variable
                pin_generator()
                break
        except ValueError:
            print("Invalid date format. Please use YYYY-MM-DD.")
```

Using the imported datetime python module, I have implemented it in these two functions as it ensures that hard coding does not need to be done in order to specify length of years, month and days as the module already accounts for that. This ensures that the code block is optimised and does not require so many conditional statements. How the check in function specifically works goes as follows: It takes in the user input and maps to the year, month and day variable which is splits each by using the - symbol. These value are assigned to a new variable called checkin-date and is converted to datetime via calling the module. Datetime is also assigned to the today module, and this is used to ensure that the check-in date entered is not a past date and compares it to the user inputted date. If the format does not match, this will trigger the Value Error exception and ask the user to re-enter the desired date.

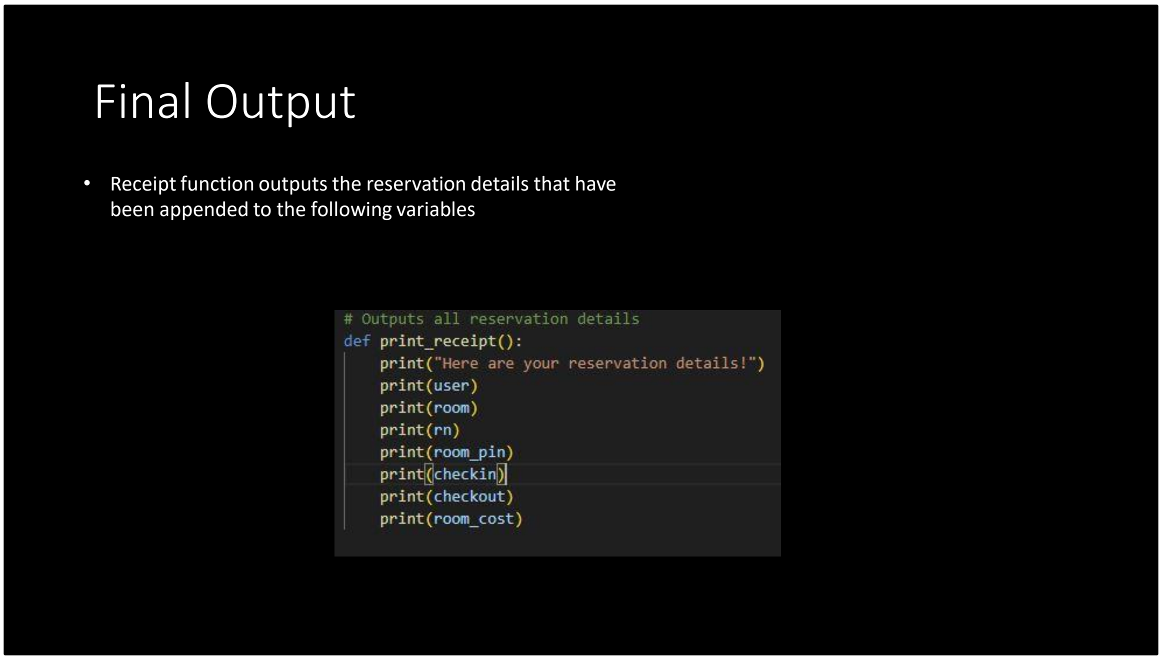
Feature 6: Pin and Room no. generator

- Generates random pin number and hotel room number
- Uses the random python module to generate numbers
- Room pin is 4 digits
- Room number is 2 digits



```python
# Generates 4 digit pin
def pin_generator():
    pin_number = random.randrange(1000, 9999)
    room_pin.append(pin_number)
    room_number()
# Generates room number
def room_number():
    number = random.randrange(1, 99)
    rn.append(number)
    print_receipt()
```

As with most hotel rooms, a room number is usually assigned as well as a pin number. The pin and room number generator functions simulate this reality and make use of the random python module to create a random pin that is outputted at the end.

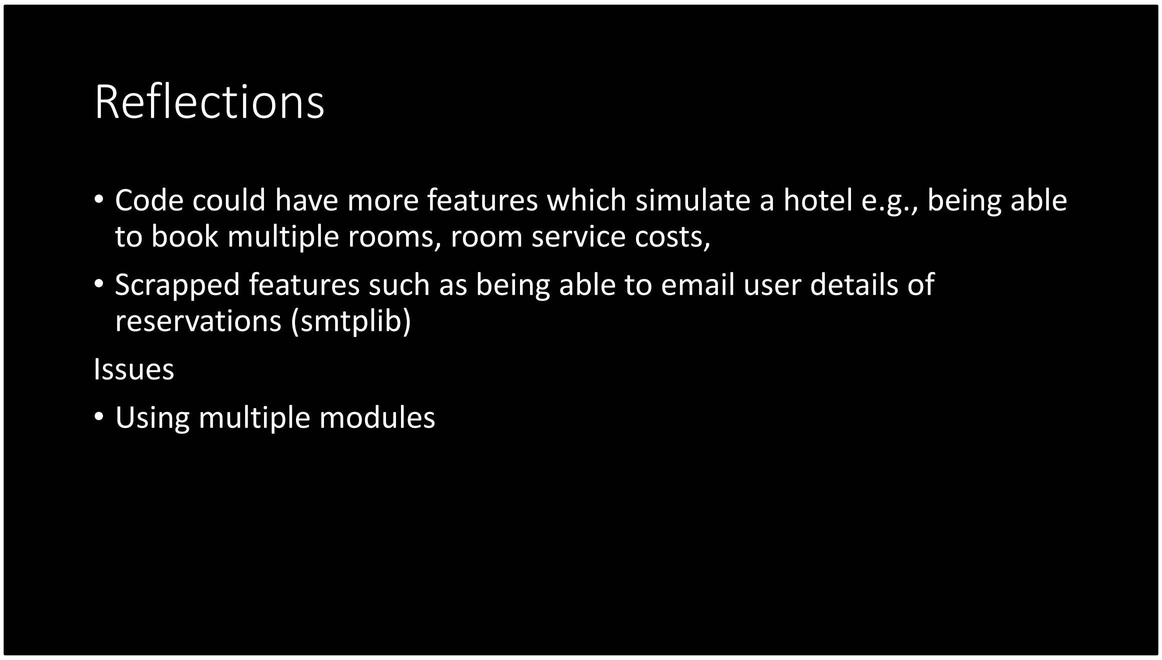


Final Output

- Receipt function outputs the reservation details that have been appended to the following variables

```
# Outputs all reservation details
def print_receipt():
    print("Here are your reservation details!")
    print(user)
    print(room)
    print(rn)
    print(room_pin)
    print(checkin)
    print(checkout)
    print(room_cost)
```

Finally, once all required details have been appended, the program will end by calling the receipt function which will print out each appended list in the terminal.



Reflections

- Code could have more features which simulate a hotel e.g., being able to book multiple rooms, room service costs,
- Scrapped features such as being able to email user details of reservations (smtplib)

Issues

- Using multiple modules

Overall, my knowledge of python has increased greatly when building my terminal program. However, I believe that my code could overall be more succinct and drier by utilising classes which I was unable to implement due to time constraints. Furthermore, features such as the ability to book multiple rooms and adding room service, features that are commonly provided by Hotel would improve user experience. A module that was unable to implement was the smtplib module which would have been able to send reservations to the user's email address of choice. However due to the complex nature of transfer protocol I decided not to pursue implementing it. Date time was also difficult module to use and assigning user inputs resulted in a number of errors which were ultimate resolved by error handling and mapping year time and day to a separate variable before feeding it into another one.