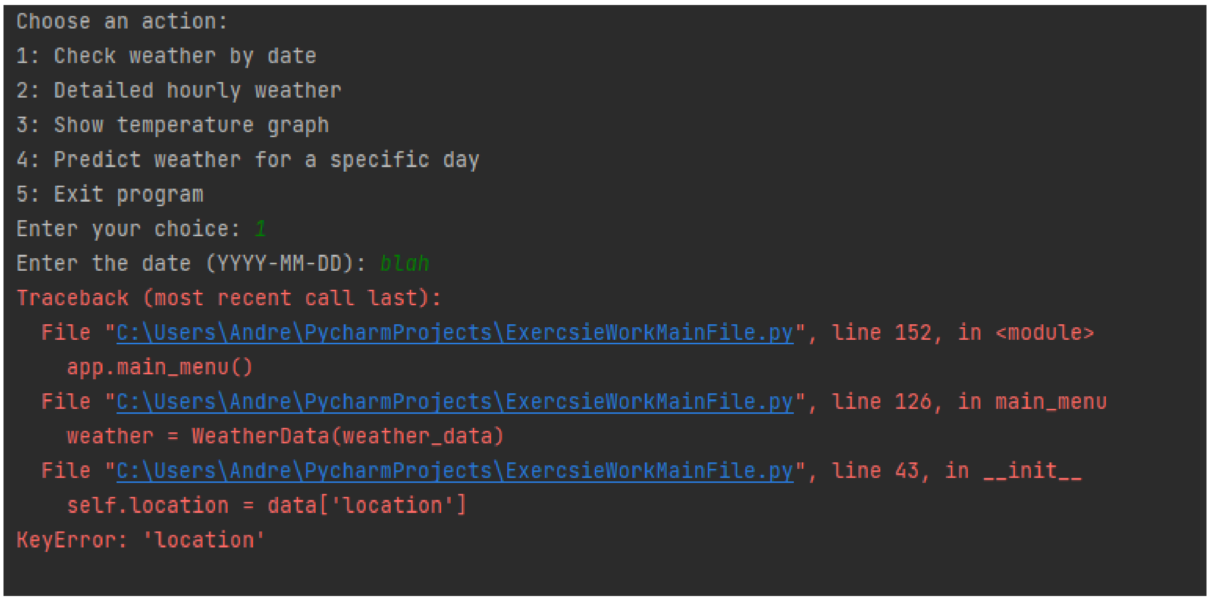
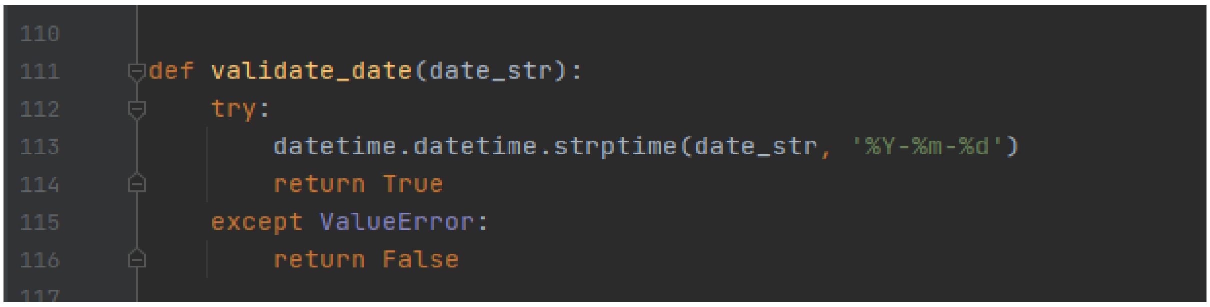
Testing report:

After the project was completed, the testing completed mostly of checking inputs in places that are not expected by the code and consequently implementing error handlers. The very first one that was attempted was

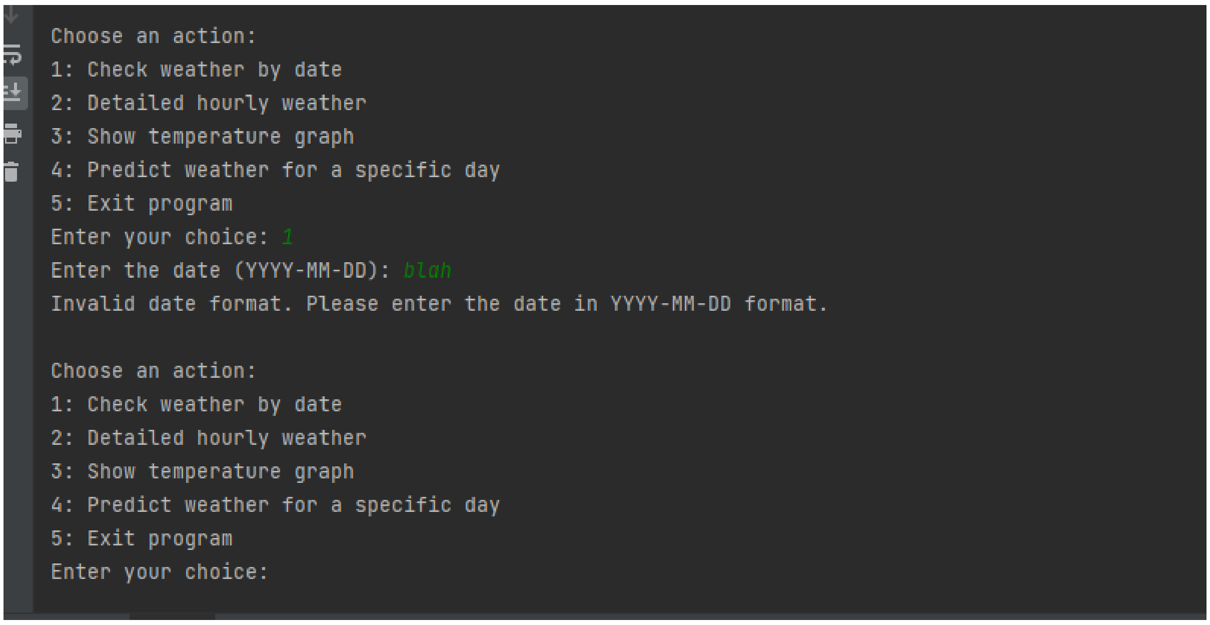


```
Choose an action:
1: Check weather by date
2: Detailed hourly weather
3: Show temperature graph
4: Predict weather for a specific day
5: Exit program
Enter your choice: 1
Enter the date (YYYY-MM-DD): blah
Traceback (most recent call last):
  File "C:\Users\Andre\PycharmProjects\ExercsieWorkMainFile.py", line 152, in <module>
    app.main_menu()
  File "C:\Users\Andre\PycharmProjects\ExercsieWorkMainFile.py", line 126, in main_menu
    weather = WeatherData(weather_data)
  File "C:\Users\Andre\PycharmProjects\ExercsieWorkMainFile.py", line 43, in __init__
    self.location = data['location']
KeyError: 'location'
```

An error that is received when the response I get from the API doesn't have the expected structure. This is however easily fixable, such as with a validation method that checks for correct input.



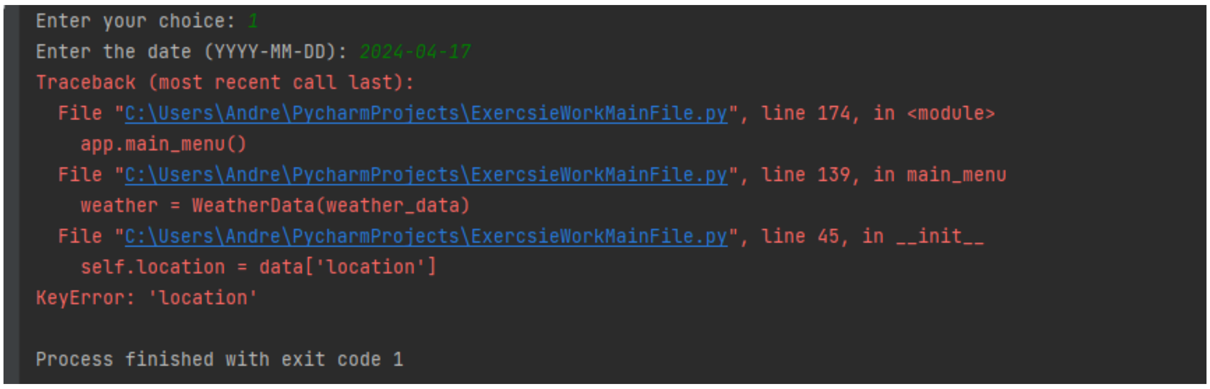
```
def validate_date(date_str):
    try:
        datetime.datetime.strptime(date_str, '%Y-%m-%d')
        return True
    except ValueError:
        return False
```



```
Choose an action:
1: Check weather by date
2: Detailed hourly weather
3: Show temperature graph
4: Predict weather for a specific day
5: Exit program
Enter your choice: 1
Enter the date (YYYY-MM-DD): blah
Invalid date format. Please enter the date in YYYY-MM-DD format.

Choose an action:
1: Check weather by date
2: Detailed hourly weather
3: Show temperature graph
4: Predict weather for a specific day
5: Exit program
Enter your choice:
```

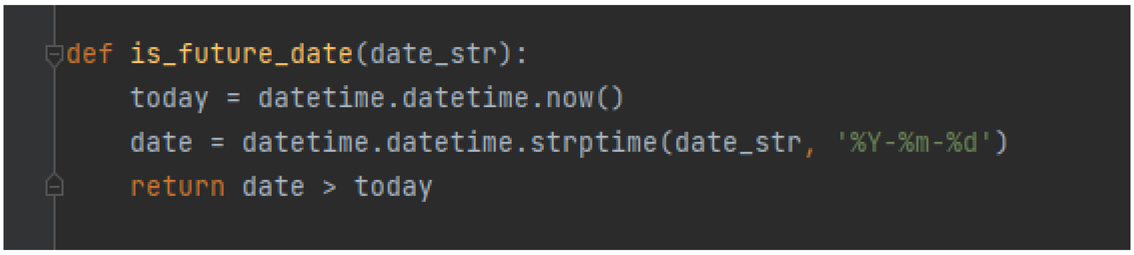
Next up was requesting the date for something that lies in the future:



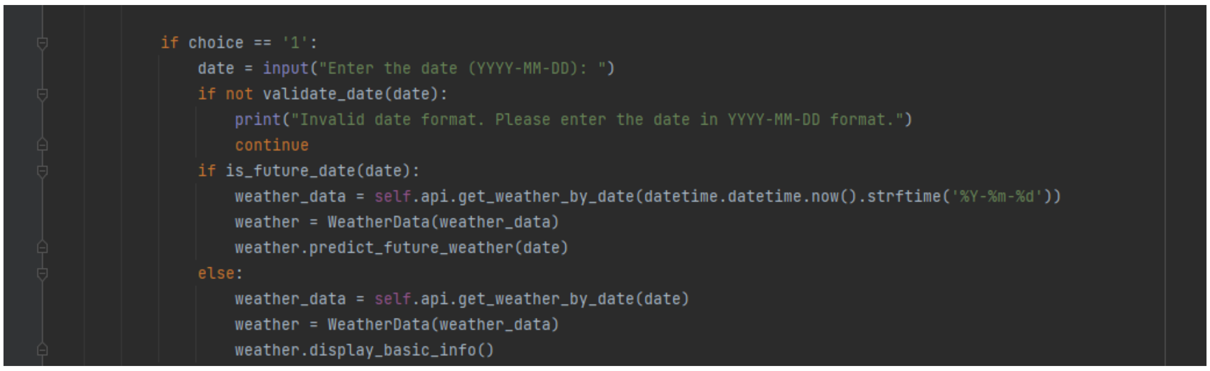
```
Enter your choice: 1
Enter the date (YYYY-MM-DD): 2024-04-17
Traceback (most recent call last):
  File "C:\Users\Andre\PycharmProjects\ExercsieWorkMainFile.py", line 174, in <module>
    app.main_menu()
  File "C:\Users\Andre\PycharmProjects\ExercsieWorkMainFile.py", line 139, in main_menu
    weather = WeatherData(weather_data)
  File "C:\Users\Andre\PycharmProjects\ExercsieWorkMainFile.py", line 45, in __init__
    self.location = data['location']
KeyError: 'location'

Process finished with exit code 1
```

I had initially tried to do this with a separate function for predicting the future (see option 4 from before), but this of course doesn't stop a user from trying to add a future date in the history method. Therefore, I simply merged the method so that it now makes predictions if you try to request a future date.



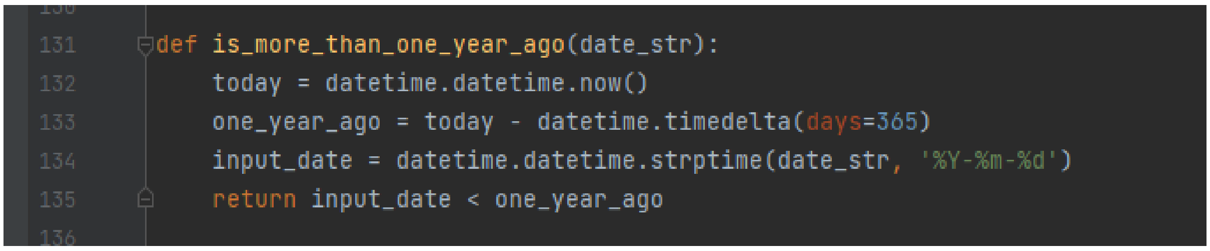
```python
def is_future_date(date_str):
    today = datetime.datetime.now()
    date = datetime.datetime.strptime(date_str, '%Y-%m-%d')
    return date > today
```



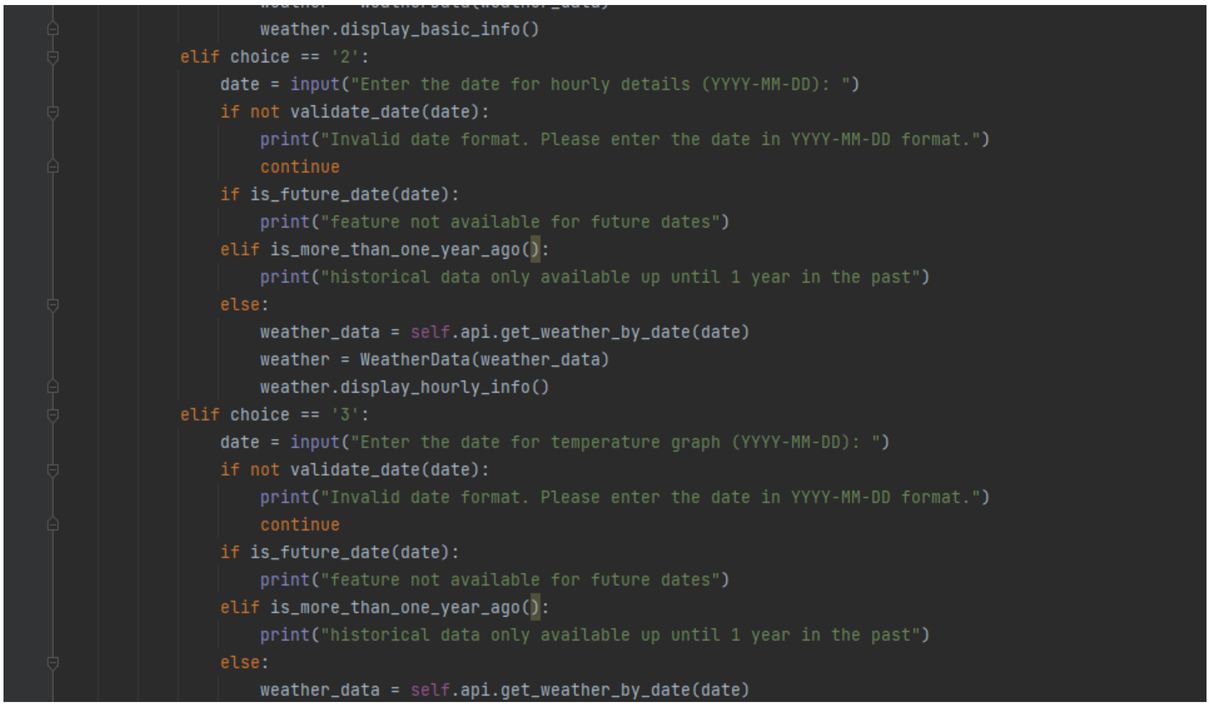
```python
        if choice == '1':
            date = input("Enter the date (YYYY-MM-DD): ")
            if not validate_date(date):
                print("Invalid date format. Please enter the date in YYYY-MM-DD format.")
                continue
            if is_future_date(date):
                weather_data = self.api.get_weather_by_date(datetime.datetime.now().strftime('%Y-%m-%d'))
                weather = WeatherData(weather_data)
                weather.predict_future_weather(date)
            else:
                weather_data = self.api.get_weather_by_date(date)
                weather = WeatherData(weather_data)
                weather.display_basic_info()
```

As for the other methods, I simply stated that it is not a supported feature

And because of the plan of my free API subscription, I can only call weather data until 1 year in the past. So I used



```python
def is_more_than_one_year_ago(date_str):
    today = datetime.datetime.now()
    one_year_ago = today - datetime.timedelta(days=365)
    input_date = datetime.datetime.strptime(date_str, '%Y-%m-%d')
    return input_date < one_year_ago
```



```python
        weather.display_basic_info()
elif choice == '2':
    date = input("Enter the date for hourly details (YYYY-MM-DD): ")
    if not validate_date(date):
        print("Invalid date format. Please enter the date in YYYY-MM-DD format.")
        continue
    if is_future_date(date):
        print("feature not available for future dates")
    elif is_more_than_one_year_ago():
        print("historical data only available up until 1 year in the past")
    else:
        weather_data = self.api.get_weather_by_date(date)
        weather = WeatherData(weather_data)
        weather.display_hourly_info()
elif choice == '3':
    date = input("Enter the date for temperature graph (YYYY-MM-DD): ")
    if not validate_date(date):
        print("Invalid date format. Please enter the date in YYYY-MM-DD format.")
        continue
    if is_future_date(date):
        print("feature not available for future dates")
    elif is_more_than_one_year_ago():
        print("historical data only available up until 1 year in the past")
    else:
        weather_data = self.api.get_weather_by_date(date)
```