Contact Lens Tracker

T1A3 Terminal Application

By Alicia Han

Purpose

- Tool for optometric practices to log and track contact lens orders
- This process can vary significantly from clinic to clinic
  - Often this includes paper forms, different trays for said paperwork, and orders across multiple websites
- Complicated for staff and can be inconvenient for patients
- This tool is designed to add and organise orders in a single programme
  - Aim is to improve efficiency and organisation within a busy clinic

Feature 1:

- Prescription converter
- Function in separate file (‘contactrx.py’), then imported to ‘main.py’
- Inputs:
  - Spectacle Prescription
  - Back Vertex Distance (typically 11-13mm)
- Output:
  - Contact Lens Prescription
  - Rounded to the nearest 0.25 dioptre
- Error handling

Feature 1:

- contactrx.py

src > contactrx.py > clrx

```python
# Calculate contact lens prescription from known clinical measurements

def clrx():

    # Get user input for spectacle prescription (Rx) and back vertex distance (BVD)
    rx = float(input('Enter spherical Rx (DS): '))
    d = float(input('Enter back vertex distance (mm): '))

    # Use formula to convert spectacle prescription to contact lens prescription
    # Round result to nearest 0.25 dioptre steps as per optometry convention
    return round((rx / (1 - ((d/1000) * rx))) * 4)/4
```

Feature 1:

- main.py

```python
import contactrx
```

```python
# Get contact lens prescription and assign to variable
while True:
    try:
        clrx = contactrx.clrx()

        # Print results to terminal, including + or - signs, as per optometry convention
        if clrx >= 0:
            print(f"The contact lens prescription required is +{clrx:.2f}DS.")
            break
        else:
            print(f"The contact lens prescription required is {clrx:.2f}DS.")
            break

    # Error handling for input parameters
    except ValueError:
        print('Please enter a valid input in numeric format.')
        continue
    except Exception:
        exception()
```

Feature 2:

- User selects lens modality
  - Daily
  - Fortnightly
  - Monthly
- Each option returns a corresponding .txt file
  - This means as lenses are discontinued or added, we only need to modify the .txt file as opposed to the main code
- User enters lens selection
  - No restrictions on this input as there are different specialist customised lenses available that won’t be in the list from the .txt file

Feature 2:

- main.py

```python
# Selecting modality, or wearing schedule, of contact lens
mod = ['daily', 'fortnightly', 'monthly']

print('Please enter the desired contact lens modality:')
for index, item in enumerate(mod):
    print(f"{index+1}. {item}")

# While loop until user enters a valid selection
while True:
        modality = input('Your choice: ')

        if modality in mod:
            break

        else:
            print("Invalid selection, please choose from 'daily', 'fortnightly', or 'monthly'.")
```

Feature 2:

- Separate .txt files

src > daily.txt

```
Coopervision Myday
Alcon Dailies Total 1
Alcon Precision 1
Acuvue Oasys 1 Day
Acuvue Moist 1 Day
BioTrue
Ultra One Day
```

src > fortnightly.txt

```
Acuvue Oasys
Coopervision cAir
```

src > monthly.txt

```
Coopervision Biofinity
Alcon AirOptix
Ultra
PureVision 2
Acuvue Vita
```

Feature 2:

- main.py

```python
# Opening selection of contact lenses from external txt file, depending on the modality
def open_list():
    print('Please choose from the following lenses: ')

    if modality.lower() == 'daily':
        f = open('daily.txt')

    elif modality.lower() == 'fortnightly':
        f = open('fortnightly.txt')

    elif modality.lower() == 'monthly':
        f = open('monthly.txt')

    # Printing list of available contact lenses to terminal
    for line in f:
            print(f'{line.strip()}')

    # Close txt file
    f. close()

open_list()

# User input chosen lens, leaving options open in case choice isn't in the list
lens = input('Enter your choice, or specify another lens: ')
```

Feature 3:

- User input patient ID
  - e.g. ID number in database, surname, initials
- User input amount needed
  - Only positive integers accepted
- Output
  - Confirmation message containing order details printed to terminal
    - ‘y’ to proceed
    - ‘n’ to end programme and start again
    - >5 invalid inputs will end programme

Feature 3:

- main.py

```python
# User input patient ID, format would be clinic-dependent, e.g. by system ID number, surname, initials, etc.
id = input('Enter patient ID: ')

# User input amount of lenses needed
while True:
    try:
        amount = int(input('Enter pairs of lenses needed: '))

        if amount >= 1:
            break

        else:
            print("Amount can't be less than 1!")

    except ValueError:
        print("Value entered wasn't an integer.")
        continue

    except Exception:
        exception()
```

Feature 3:

- main.py

```python
for retry in range(5):
    # Confirm with user that the details are correct
    confirm = input(f"Your order is: {amount} pair(s) of {clrx:.2f}DS {lens} lenses, for patient #{id}. Y/N: ")

    if confirm.lower() == 'y':
```

```python
    elif confirm.lower() == 'n':
        print('Please start again.')
        sys.exit(1)

    else:
        inputfunctions.y_or_n()

else:
    inputfunctions.invalid_choices()
```

Feature 4:

- Add order details to csv file
- User prompted to confirm whether lenses have been ordered
- Import datetime
- If yes, order details will be appended to ‘ordered.csv’
- If no, order details will be appended to ‘pending.csv’

Feature 4:

- main.py

```python
import csv
from datetime import date
```

```python
        today = date.today().strftime('%d-%m-%Y')
        order = {'Date': today, 'Patient ID': id, 'Modality': modality, 'Lens': lens, 'CL Rx': clrx, 'Amount (pairs)': amount}
        fields = ['Date', 'Patient ID', 'Modality', 'Lens', 'CL Rx', 'Amount (pairs)']

        for retry in range(5):
            # Has this order been placed
            if_ordered = input('Has this order already been placed? Y/N: ')

            # If yes, print to ordered.csv file with today's date
            if if_ordered.lower() == 'y':
                with open('ordered.csv', 'a') as ordered:
                    writer = csv.DictWriter(ordered, fieldnames = fields)
                    writer.writerow(order)
                break

            # else, print to pending.csv file with today's date
            elif if_ordered.lower() == 'n':
                with open('pending.csv', 'a') as ordered:
                    writer = csv.DictWriter(ordered, fieldnames = fields)
                    writer.writerow(order)
                break

            else:
                inputfunctions.y_or_n()

        else:
            inputfunctions.invalid_choices()
```

Feature 4:

src > ordered.csv

```csv
Date, ID, Modality, Lens, CL Rx, Amount (pairs)
14-05-2023,123,daily,Sample Lens,-7.0,3
14-05-2023,456,monthly,AnotherSample,11.5,5
```

src > pending.csv

```csv
Date, ID, Modality, Lens, CL Rx, Amount (pairs)
14-05-2023,789,fortnightly,ExampleLens,-9.25,10
14-05-2023,345,daily,OneMoreExample,10.75,6
```

Feature 5:

- Display order history
- User will be prompted to enter whether they would like to view order history
- If yes:
  - ‘ordered.csv’ and ‘pending.csv’ are displayed in terminal in dict
- If no:
  - Programme ends with ‘thank you’ message

Feature 5:

- main.py

```python
# Read csv files and print as dictionary

def read_ordered_lenses():
    print('The following orders have been placed:')
    with open('ordered.csv') as f:
        reader = csv.DictReader(f)
        for row in reader:
            print(row)

def read_pending_lenses():
    print('The following orders are pending:')
    with open('pending.csv') as f:
        reader = csv.DictReader(f)
        for row in reader:
            print(row)
```

Feature 5:

```
Would you like to see the history of order details? Y/N: y
The following orders have been placed:
{'Date': '14-05-2023', ' ID': '123', ' Modality': 'daily', ' Lens': 'Sample Lens', ' CL Rx': '-7.0', ' Amount (pairs)': '3'}
{'Date': '14-05-2023', ' ID': '456', ' Modality': 'monthly', ' Lens': 'AnotherSample', ' CL Rx': '11.5', ' Amount (pairs)': '5'}
The following orders are pending:
{'Date': '14-05-2023', ' ID': '789', ' Modality': 'fortnightly', ' Lens': 'ExampleLens', ' CL Rx': '-9.25', ' Amount (pairs)': '10'}
{'Date': '14-05-2023', ' ID': '345', ' Modality': 'daily', ' Lens': 'OneMoreExample', ' CL Rx': '10.75', ' Amount (pairs)': '6'}

Thank you for using this programme!
```

Feature 5:

- main.py

```python
# Get history of orders placed and pending orders

while True:
    history = input('Would you like to see the history of order details? Y/N: ')

    if history.lower() == 'y':
        read_ordered_lenses()
        read_pending_lenses()
        break

    elif history.lower() == 'n':
        break

    else:
        inputfunctions.y_or_n()

print('\nThank you for using this programme!\n')
```

Challenges

- Not knowing if my ideas were implementable
- Bugs bugs bugs!

What I Enjoyed…

- Brainstorming ideas that will be beneficial in real life scenarios
- Using Trello as a checklist
- Overcoming bugs in the code
- Understanding Python better as I progressed through this assignment

Ethical Issues

- Privacy concerns when patient ID is involved
  - Especially when patient ID includes any identifying information such as surnames
- Which lenses to prioritise in the programme
  - Are lenses listed at the top more likely to be seen and chosen, thereby unintentionally increasing the number of patients being fitted with said lenses?\
- How will this tie in with existing software in optometric practice?
  - Is this something that existing clinic software can implement into their own programme?

Further ideas

- Move orders from ‘pending.csv’ to ‘ordered.csv’ when order has been placed
- Ability to calculate astigmatism in the prescription converter
- Ability for user to organise csv files, e.g. by patient ID or name of lens
- Set upper limits to prescriptions available so only valid contact lenses are listed depending on the prescription
- Ability to send order details to suppliers via email

Thank you.