# Snooze It

T1A3 Terminal Application Assignment

# Table of Contents

# Feature 1 - User Input Prompts & Writing into CSV

```
def write_user_input(date, hours_of_sleep, quality_of_sleep, caffeine, journal, journal_entry):
  with open('user_information.csv', 'a', newline='') as file:
    headers = ['Log Date', 'Hours of Sleep', 'Quality of Sleep (1-10)', 'Caffeine Intake', 'Write Journal', 'Journal Entry']
    outputDictWriter = csv.DictWriter(file, headers)
    outputDictWriter.writerow({'Log Date': date, 'Hours of Sleep': hours_of_sleep, 'Quality of Sleep (1-10)': quality_of_sleep,
    print("Thank you for using Snooze It. Your data has been saved!")
```

- File provided with correct name & designated headers written
- Open file with append mode
  - Allows data to be written after any existing data already written in the file, instead of write, which overwrites existing data
  - Create Dictwriter object
  - Writerow - write user inputs into respective headers

# Feature 2 - Single Sleep Log Searches

```
def log_search_day():
    df = pd.read_csv("user_information.csv")
    df2 = pd.to_datetime(df['Log Date'], format='%Y-%m-%d') # converts data type into a datetime64[ns] object
    date = input("Please enter the date of the log (YYYY-MM-DD), then press <Enter>: ")
    search_date = dt.datetime.strptime(date, "%Y-%m-%d")
    current_date = dt.datetime.now()
    if search_date > current_date:
      raise InvalidDateError()
    filtered_df = df[df2 == search_date]
    if filtered_df.empty == True:
      raise EmptyDataError()
```

- Open CSV in read mode
- Convert Dates from Pandas dataframe into a datetime object for easier comparison to search date and current date
- Using dataframe to work with data in pandas dataframe

# Feature 2 - 7 Day Sleep Log Search

```
def log_search_week():
  df = pd.read_csv("user_information.csv")
  df2 = pd.to_datetime(df['Log Date'], format='%Y-%m-%d')
  start_date_raw = input("Please enter a date to backtrack from in (YYYY-MM-DD), then press <Enter> : ")
  start_date = dt.datetime.strptime(start_date_raw, '%Y-%m-%d')
  if start_date > dt.datetime.now():
    raise InvalidDateError()
  end_date = start_date - timedelta(days=7)
  mask = (df2 > end_date) & (df2 <= start_date)
  df3 = df.loc[mask]
  if df3.empty == True:
    raise EmptyDataError()
```

- Open CSV in read mode
- Convert Dates from Pandas dataframe into a datetime object for easier comparison to search date and current date
- Using dataframe to work with data in pandas dataframe
- Use timedelta object to automate 7 day week search and minimise user input errors
- Filter dates to less than start date and not greater or equal to end date calculation

# Feature 3 - Print Random Sleep Tip

```
def sleep_tip():
  with open("sleep_tips.csv", "r") as f:
    csv_reader = csv.reader(f)
    sleep_tips = list(csv_reader)[1:]
  print(f'\nSleep Tip: {random.choice(sleep_tips)}\n')
```

- Open file in read mode
- Create reader object
- Locate a sentence, skipping the header of the file
- Print a random sentence in the file

# Challenges in Development

- Challenges of writing a large amount of code for an application
- Understanding all of the concepts to implement into the project
- Trouble with translating created Pseudocode into python coding language
  - Improve Pseudocode
- Learning to use Modules
  - Pandas module
    - Working with Pandas Dataframe
      - Referring constantly to Documentation & Look at examples
    - Conversion of string objects and datetime objects for data/ input comparison
  - Datetime module
    - Confusion with datetime as a module or as a class
  - Utilising Terminal Menu

# Ethical Issues

- Personal Issues of Users
- Need to gain permission on storing user's private information & thoughts in the log

Solutions:

- Provide security in storage of user Information
- Establish data encryption, firewalls, protection of user information

# Future Developments of Application

- Add an Accounts / Login Feature

- Apply more Error handling in Feature 1
  - Avoid repetition of same date entries on record