ECM1400 Programming
ECM1400 Continuous Assessment

Date set: 1st November, 2021
Hand-in date: **11:59am 10th December, 2021**

This continuous assessment (CA) comprises 100% of the module assessment.

Note that electronic submission is required through eBart.

This CA is designed to test the programming concepts that we will cover in the entire module. This assignment will require the application of fundamental programming constructs, control flow, data structures and design patterns to develop and deliver a Python application to solve a real-world problem. Note that some of the content is only taught within 2 weeks of the deadline, however, this should not stop you making a start and implementing a lot of the functionality well in advance of the deadline.

Specification

Since the outbreak of COVID-19 the day-to-day routine for many people has been disrupted and keeping up-to-date with the local and national infection rates and news on government guidelines has become a daily challenge.

In this assignment you will implement a simple personalised covid dashboard. Dashboards are automated systems that help visualise input data streams. Your dashboard application will coordinate information about the COVID infection rates from the Public Health England API and news stories about Covid from a given news API.

The implementation of an automated data dashboard is not like a typical Python script. The dashboard will need to run continuously and respond to events. These events can either be scheduled by the developer or triggered by user inputs and require an event-driven software architecture. Your personalised dashboard must use (and not edit) the provided front end web template for the user interface and you will implement the dashboard application backend in Python following the guidance below.

- **Covid data updates**

  All the Covid data handling functionality specified in this section of the specification should be contained in a module called `covid_data_handler.py`.

  Read data from a file: Write a function called `parse_csv_data` that takes an argument called `csv_filename` and returns a list of strings for the rows in the file. Test this function with the data file, nation_2021-10-28.csv. A sample example test is given below to check the function is identified correctly.

  Link to download example csv file available in the assessment tile on ELE: `https://vle.exeter.ac.uk/course/view.php?id=10422#section-3`

```
from covid_data_handler import parse_csv_data
def test_parse_csv_data():
  data = parse_csv_data('nation_2021-10-28.csv')
  assert len(data) == 639
```

  Data processing: Next write a function called `process_covid_csv_data` that takes a list of data from an argument called `covid_csv_data`, as would be returned from the function above, and returns three variables; the number of cases in the last 7 days, the current number of hospital cases and the cumulative number of deaths, as contained in the given csv file. The cases in the last 7 days should be calculated by summing the new cases by specimen date for the last 7 days ignoring the first entry as the data is incomplete for that day.

```
from covid_data_handler import process_covid_csv_data
def test_process_covid_csv_data():
  last7days_cases, current_hospital_cases, total_deaths =
    process_covid_csv_data( parse_csv_data( 'nation_2021-10-28.csv' ) )
  assert last7days_cases == 240_299
  assert current_hospital_cases == 7_019
  assert total_deaths == 141_544
```

  Live data access: Write a function called `covid_API_request` that takes two arguments called `location` and `location_type` with the default values of `"Exeter"` and `"ltla"` and returns up-to-date Covid data as a dictionary. You should access current COVID-19 data using the `uk-covid-19` module provided by Public Health England. The documentation on how to use the uk-covid-19 module is found here, `https://publichealthengland.github.io/coronavirus-dashboard-api-python-sdk/`. You may also need to process the JSON file returned from the API using the `json` module (`import json`). Note you might also want to implement further functions to process this data as it is formatted differently to the csv data.

  Automated updates Write a function called `schedule_covid_updates` that takes two arguments called `update_interval` and `update_name`. Use the `sched` module to schedule updates to the covid data at the given time interval.

  [10 marks]

- **Covid news**

  The other part of your dashboard will handle and display information about news articles with headlines that mention the terms "Covid", "COVID-19" or "coronavirus" (not case sensitive). Your system will achieve this by pulling information from a news API and updating a list

of headlines on the dashboard. This functionality should be implemented in a module called `covid_news_handling`.

Write a function called `news_API_request` that takes an argument called `covid_terms` with a default value of `"Covid COVID-19 coronavirus"`. Access current news data using the `requests` module and the News API (https://newsapi.org/).

Write a function called `update_news` that uses the `news_API_request` within the function and is able to update a data structure containing news articles. Integrate this with the scheduled updates. Note, news should be updated at a different interval than the covid data. Also note, there is a way to 'remove' articles that you have seen from the interface. These removed articles should not reappear when the news is updated.

[10 marks]

- **User interface**

  All smart systems are designed with automation in mind so there is an emphasis on minimal user input, however, the user needs to be able to set and cancel updates and remove news article widgets. A template will be provided for the html interface although you will need to map the interface with functionality. The interface should use the `flask` module to enable the user to manually trigger news and covid data updates, schedule and cancel future updates and remove news articles.

  Link to download template html file available on the assessment tile on ELE: `https://vle.exeter.ac.uk/draftfile.php/1357171/user/draft/904436418/index.html`

  [10 marks]

- **Project delivery**

  This specification is for a real-world application. This means there are more detailed consideration about how the project is delivered. The project needs documentation so new users can deployment and customise their app, as well as monitoring and regularly testing the functionality.

  – Configuration file: Sensitive information, such as api keys and user credentials, and configuration information, such as location update intervals and filepaths for application resources, should be stored in a configuration file rather than in the source code so they can be easily updated. Create a json file called `config.json` that contains all the information for updating the dashboard settings and is used to run and personalise the dashboard. **[5 marks]**

  – Logging: A central functionality in any event-driven architecture is tracking and handling events automatically. This means the system should be executing tasks remotely, while you arent watching. Logging is therefore an important feature that will make the program accountable and make it possible to trace behaviour and recover state. You should log all events that happen in your dashboard and categorise different types of event that may be treated in different ways, such as errors if the web services don't respond. **[5 marks]**

  – Testing: As your software is designed to depend on third party services, like the PHE Covid-19 API, and is designed to run continuously there is a good chance errors will occur while the program is deployed. As part of your code you should include unit testing for each of your functions and include a deployment test cycle as well as scheduling tests to

regularly check the functionality of your program. Some sample tests are provided. You should extend these tests and test your code using the `pytest` module. **[5 marks]**

– Distribution and Documentation: The software you develop may be useful for more than just one user and it may inspire others to extend the code. Host your code publicly on github or PyPI so the Python community can extend or import your code. As the code is designed for others to use, documentation is foremost and should consider both a user and a developer. A must know how to deploy and use the system and a developer must know how the code is structured, what it does and how to extend it. **[5 marks]**

You should carefully follow the functionality described. Note the requirements intentionally do not include some details and you should make design decisions carefully.

Submitting your work

The CA requires electronic submission to the BART online submission platform.

Electronic You should submit your Python code via the electronic submission system at `https://bart.exeter.ac.uk/` under CEMPS and Harrison. Use the category containing ECM1400 and 2021 Continuous Assessment. Upload a compressed version of your files as a single file using the `zip` compression format.