MPHYG001: Research Software Engineering with Python (2018)

Answer TWO questions from THREE

The provisional allocation of the maximum marks for each part of each question is indicated by the number in square brackets on the right-hand side.

For this MPHYG001 exam, the code is provided in Moodle. Download the code, and unzip the zip file to a folder of your choice.

Use of the Internet, and referring to on-line documentation is allowed. File sharing and messaging with fellow students is not allowed.

At the end of the exam you must upload the code to Moodle, as if it were an assignment.

QUESTION 1

The Python Language

In folder: `python_language_1`

The file `python_language_1_data.csv` contains daily rainfall measurements for a field station in the mountains of the Eastern USA between 1937 and 2012.

The code to answer this question should be saved in a file called `rainfall.py`.

(a) Write a python function to:

- Load this data into memory [1]
- Reformat the data into a dictionary keyed by year, using appropriate data types, an example of the expected output is provided [5]
- Write the dictionary to disk in JSON format [1]

(b) Create a function that takes the filename of the JSON file, a year, and an optional line colour as arguments and plots a timeseries of daily rainfall for that year. Save a formatted plot in `png` format showing the data for 1998. [6]

(c) Write a function that takes the filename of the JSON file and produces a plot showing the mean annual rainfall over a user specified time period. Save a formatted plot in `png` format showing the data from 1988 to 2000 (inclusive). [4]

QUESTION 1 - CONTINUED

(d) You discover that for the years 2005 to 2011 the rain gauge was malfunctioning. The published correction factor is

```
rainfall_value * 1.2 ^ root(2)
```

Write a function to apply this equation to a given value and **two** functions which apply this function to all of the data for a given year, returning the year's corrected results as a list.

- Function 1 should solve this using a for loop
- Function 2 should use a list comprehension [5]
- In the docstring for one of these functions, evaluate the pros and cons for these two approaches to the same task. [2]

(e) An additional mark will be awarded for well formatted and clearly commented code. [1]

CONTINUED

QUESTION 2

Testing

In folder: `testing_1`

An example python class, `simplemaths`, performs mathematical operations on an integer.

(a) Modify the constructor to ensure that an appropriate error is raised if the user attempts to pass anything other than an integer into the class. [5]

(b) Inside the provided file `test_simplemaths.py`, write a series of unit tests using `pytest` which test each of the methods in the class, ensuring that negative tests are included and that a range of inputs are tested. [16]

The mark breakdown is as follows:

- Constructor [5]
- ~~square and factorial [2]~~
- ~~power [2]~~
- ~~odd_or_even [3]~~
- square_root [4]

~~(c)~~ Write a short file, `testing.md` which explains to a novice user how to run the tests. [2]

~~(d)~~ An additional two marks are available for tests which are well written and self-documenting. [2]

TURN OVER

QUESTION 3

Refactoring

In folder: `refactoring_1`

The file `refactoring_1_bad_code.py` contains a poorly written implementation of a Julia set. The output of which can be seen in `refactoring_1_example.png`

(a) Refactor this code into one or more functions, contained within a file called `Julia.py`. [14]

(b) The example code displays the plotted Julia set within a `matplotlib` window, modify the code to instead save the image to disk in an appropriate format. [1]

(c) Configure this script so that it can be executed from the command line and identify meaningful parameters in the provided code that can be exposed to a user. Use `argparse` to document and handle these parameters. [10]

END OF PAPER