Assessed Exercise 2

May 10, 2019

1 Assessed Exercise 2

This exercise contains five tasks. You should attempt all of them, ensuring that you have completed every part of each task. You should write your solutions into this Jupyter notebook; you may add additional cells as necessary. Your code should run correctly when each cell of the notebook is executed in the order that they appear.

Your attempt must be submitted via Wattle by 17:00, Friday 31st May. To submit, you should: - Click `File > Save and Checkpoint` - Click 'File > Download as > Notebook (.ipynb) - Save the file to your computer - Upload this file via Wattle.

Remember to document your code using docstrings and comments. Where tasks require you to produce figures or other output, marks will be awarded for the quality of presentation: remember to include axis labels, titles, colorbars, etc. A full marking rubric is given below.

You may make use of (i.e., import) the following modules: - `numpy` - `matplotlib` - `pandas` - `cartopy` - `datetime` - `math` - `re`

No other modules may be used unless you have obtained permission from the course convenor (Dr Andrew Valentine). Note that you may not need to use all of the modules in the above list.

1.1 Drop-in sessions

There will be four drop-in sessions before the submission deadline. These are times when a course tutor will be available to answer any questions you may have about Python, any of the exercises, or about what is required for this Assessment (obviously, we cannot help you complete the tasks!). These will take place:

- Wednesday 15 May, 14:00-16:00
- Tuesday 21 May, 12:00-14:00
- Friday 24 May, 12:00-14:00
- Wednesday 29 May, 14:00-16:00

All will be in the TerraView Room (Jaeger 2, top floor). Attendance at these sessions is optional, and you need not come unless you have specific questions you wish to ask. However, this is an opportunity to get help, advice and feedback if you are finding anything difficult.

1.2 Oral examinations

When completing this assessed exercise, you are encouraged to make use of online resources and talk to your colleagues to help you find effective solutions, and fix problems. However, everything

you submit must be your own work, and you must be able to explain how your code works. To ensure this, everyone must attend a short oral examination. During this, we will ask you to explain aspects of your solution, and discuss why you chose to solve the task in that way. No particular preparation is required, although you might wish to read through your submitted work before attending the examination.

Oral examinations will take place on **Tuesday 4th June** in Andrew Valentine's office at RSES (Jaeger 2, Room 147a) as follows:

Time	Name
10:00	Darren Tsi Hwa Choo
10:30	Nini He
11:00	Sophaelia Hofseth-Ronsen
11:30	Ruoran Nie
12:00	Yujia Shao
14:00	Aditya Sharma
14:30	Callum Shaw
15:00	Xiaoyu Su
15:30	Tianyue Xing
16:00	Yi Zhao

If your allocated time is not convenient, you may swap with someone else, **provided that both people email Andrew Valentine (andrew.valentine@anu.edu.au) to confirm the change**. If you are unable to find someone to swap with, or are entirely unavailable on 4th June, please contact Andrew Valentine immediately to arrange an alternative time. **Failure to attend an oral examination will result in you receiving no credit for this assessment.**

1.3 Marking

High distinction

- Submitted code is a complete and elegant solution to the stated problem. No errors or omissions are present. Code is concise, clear and readily understood, displaying a level of sophistication and a full understanding of the Python language. Solutions may extend beyond the stated problem, or use aspects of Python not directly taught during the course.
- Code is fully and clearly documented, with comprehensive, well-structured explanation of function interfaces and program logic.
- Output (e.g. maps & figures) is of exemplary quality, conveying all required information in a visually-appealing manner, and of a standard suitable for submission in a top-quality journal.
- The student displays a comprehensive understanding of their code, and can fully justify the choices they made. They are able to discuss other potential solution strategies, and evaluate their pros and cons.

Distinction

- Submitted code fully addresses the stated problem, providing correct output(s) in all cases. No errors or omissions are present. Code is concise, well-structured and easy to understand. The full range of Python constructs are employed. Errors (e.g. bad user inputs) are handled appropriately.

- Code is fully-documented through docstrings and explanatory comments.
- Output conveys all required information without error or omission. Figures are suitable for inclusion in a journal publication, being generally well-presented and visually-appealing.
- The student displays a comprehensive understanding of their submission, and is able to discuss and justify the decisions and choices they made during implementation.

Credit - Submitted code fully addresses the stated problem, providing correct output(s) in all cases. No significant errors or omissions are present. Code is well-structured and clear, making use of a range of Python constructs as appropriate. Some attempt is made to check for and handle errors (e.g. bad user inputs) as appropriate. - Some attempt is made to document the code through docstrings and explanatory comments, but this is incomplete or lacking in detail. - Output conveys all required information without error or omission. Figures are suitable for inclusion in a journal publication but may lack polish and visual appeal. - Student displays a comprehensive understanding of their submitted code, and can explain how it works.

Pass - Submitted code largely addresses the stated problem. Some errors or omissions may be present, and outputs may not be correct in all cases. Code may be unsophisticated, with significant redundancy or inefficiency, and may use only a limited subset of the Python language. There may be little or no effort to handle common sources of error (e.g. incorrect user inputs). - Little or no attempt is made to document the code through docstrings and explanatory comments. - Output conveys the required information, perhaps with minor errors or omissions. Figure quality is well below the standard expected in a good journal. - Student displays basic familiarity with their submitted solution.

Fail - Submitted code does not provide a solution to the stated problem. - Output entirely fails to convey the required information. - Student is unable to explain how their code works, leading the examiners to believe that the submission does not represent the students own, independent efforts.

1.4 1. Letter Frequency Analysis

According to at least one source, the 5 most common letters in written English are E (on average, 13% of all letters are an E), T (9%), A (8%), O (8%), and I (7%).

Write a function that will analyse any piece of text and produce a table showing the frequency of each letter.

```
In [ ]:
```

1.5 2. Birthday Calculator

Write a function that asks the user to input their date of birth, and which then calculates and displays the number of days until the user's next birthday. Your function should handle incorrect inputs sensibly.

Hint: You will probably want to use the `datetime` module.

```
In [ ]:
```

1.6 3. The Preliminary Reference Earth Model

One of the most widely-used models describing the average seismic properties of the Earth is the Preliminary Reference Earth Model (PREM) developed by Dziewonski & Anderson (1981).

This provides estimates of density (ρ), P-wave velocity (V_p) and S-wave velocity (V_s) at any depth within the Earth.

In the upper mantle (i.e. in the uppermost 670 km), the model is specified by the following set of equations:

Depth range	Density (ρ/g cm^{-3})	P-wave velocity (V_p/km s^{-1})	S-wave velocity (V_s/km s^{-1})
0.0 - 3.0	1.020	1.450	0.0
3.0 - 15.0	2.600	5.800	3.200
15.0 - 24.4	2.900	6.800	3.900
24.4 - 220.0	$2.6910 + 0.6924x$	$4.1875 + 3.9382x$	$2.1519 + 2.3481x$
220.0 - 400.0	$7.1089 - 3.8045x$	$20.3926 - 12.2569x$	$8.9496 - 4.4597x$
400.0 - 600.0	$11.2494 - 8.0298x$	$39.7027 - 32.6166x$	$22.3512 - 18.5856x$
600.0 - 670.0	$5.3197 - 1.4836x$	$19.0957 - 9.8672x$	$9.9839 - 4.9324x$

In these equations, the variable x is related to depth, d by:

$$x = \frac{6371 - d}{6371}$$

i.e. x will always be in the range $0 \leq x \leq 1$. To find the properties at a given depth, we must first identify the relevant row of the table, then compute x and evaluate the equations in that row.

1. Write a function that can be called as `rho,vp,vs = evaluatePREM(depth)`, i.e. taking a single argument (the depth for which the properties should be computed) and returning three numbers, corresponding to the density, P-wave velocity, and S-wave velocity, respectively. Make sure your function is properly documented, and handles input depths outside the range 0-670 in a sensible manner.

2. Using this function, make a plot showing density, P-wave velocity, and S-wave velocity against depth in the upper 670 km. You may wish to consult Fig. 7 of Dziewonski & Anderson's paper to check your results; however, you are free to design your plot as you wish, and it does *not* need to be an exact replica of theirs. (Note that their figure contains various additional lines that are not required for this exercise.) Ensure that all axes, etc, are fully labelled; marks will be awarded for the quality of presentation.

3. Use your function to generate a text file containing four columns, listing the numerical values of depth, density, P-wave velocity and S-wave velocity every 1 km from the surface (depth=0 km) to the transition zone (depth=670 km). Your file should be nicely-formatted, with any non-data lines (e.g. column headings) beginning with the character `#`.

In [ ]:

1.7 4. R/V Investigator Cruise Data

The R/V Investigator is Australia's main ship for ocean-based research. You have been provided with a file `in2019_v07_uwy_data.csv` which contains minute-by-minute records from a recent research cruise, which sailed from Hobart in Tasmania, along the south-east coast of Victoria, and back to Hobart. You will notice that any 'missing' observations are represented by the value `-999.9`.

1. Plot a map showing the route taken by the ship (column names 'Longitude (degree_east)' and 'Latitude (degree_north)'). Mark and label the following:
    - The deepest and shallowest points encountered on the voyage (column name 'Depth (m)')
    - The hottest and coldest water encountered on the voyage (column name 'Water Temperature (degC)')
2. Make one figure containing histograms showing the weather encountered during the voyage, including columns 'Port True Wind Direction (degree)', 'Port True Wind Speed (knot)', 'Port Air Temperature (degC)' and 'Port Accumulated Hourly Rain (mm)'. (*'Port' here refers to the sensor on the port - i.e., left - side of the ship*)
3. To calculate the distance between two points expressed as (latitude,longitude), we can use the Haversine formula,

$$d = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

    Here, ϕ_1 and ϕ_2 represent the latitude of points 1 and 2, respectively, while λ_1 and λ_2 represent longitude. R is the radius of the Earth; you may assume $R = 6371.0$ km. The result, d, is the distance in km. Write a function implementing the haversine formula, and then use this to calculate the total distance covered by the ship during its cruise. *Hint: Remember, mathematical functions are available by importing NumPy. Also, the functions* `np.sin` *and* `np.cos` *expect angles in radians, not degrees - use* `np.deg2rad` *to convert degrees to radians.*

In [ ]:

1.8 5. Infrared Spectroscopy

The file `11octrll32bs1_2016-10-11T12-48-25.asp` is produced by an infrared spectrometer. Look at the file contents. It is formatted as follows:

- Line 1: Number of data points in file
- Line 2: Wavenumber (cm^{-1}) associated with first measurement in file
- Line 3: Wavenumber associated with last measurement in file
- Line 4-6: Miscellaneous information (not required for this exercise)
- Line 7-end: Reflectance measurements (%).

You can assume that the measurements are evenly-spaced in wavenumber.
The filename encodes some additional information:

```
11octrll32bs1_2016-10-11T12-48-25.asp
\___/\____/\| \________/ \______/
  A     B   C      D        E

A - Date of experiment (11 October)
B - Location of experiment ('rll32b')
C - Sample number (Sample 1) -- starting with 's'
D - Date of experiment (again; 11 October 2016)
E - Time of experiment (12:48:25)
```

Write a function that takes the filename as argument, and produces a high-quality figure, showing reflectance (y-axis) plotted against wavenumber (x-axis). The following information should also be displayed on the figure: - The date and time at which the experiment was run - The location, panel and sample numbers - The filename of the datafile - The wavenumber corresponding to the maximum amplitude in the dataset.

Your function should work for *any* file of this format. To help you test this, you have been provided with a second file from the same instrument - `12octrll61s1_2016-10-12T10-36-43.asp`.

In [ ]: