INFO370 Problem Set 1: Python, data manipulations (100 pt)

January 5, 2022

Instructions

This is the first problem set. These 100 points will give you 10 points of the final grade.

The goal of this problem set is to get you going with the basic python, such as creating and computing variables, and doing tricks with lists and dicts and functions. The good background reading is the Lubanovic (2014) book, chapters 2 (data types), 3 (lists, dicts, sets), 4 (code structures). The basics is also explained in python notes http://faculty.washington.edu/otoomet/machinelearning-py/python.html.

General requirements:

- All materials and resources that you use (with the exception of lecture slides) must be appropriately referenced within your assignment. In particular, note that Stack Overflow is licensed as Creative Commons (CC-BY-SA). This means you have to attribute any code you pick from SO (a link to the question/answer webpage will normally do).
- Be sure that each visualization (graph or table) adds value to your written explanation; avoid redundancy – you do no need four different visualizations of the same pattern.
- Don't output irrelevant information, or too much of relevant information. A few figures is helpful. A few thousand figures is useless. In particular, do not print thousands of lines of data!
- As the final submission, you should submit a) code; b) output; and c) explanations. If you are working with jupyter notebooks, all this can be easily included in the same notebook file but you still have to submit both your original file (so you grader can actually run the code), and an html version of it (which is much faster to check).
- Working together is fun and useful but you have to submit your own work. Discussing the solutions and problems with your classmates is all right but do not copy-paste their solution! First understand it, and thereafter create your own solution. Please list all your collaborators!

  1. ...
  2. ...

  ...
- Attempt each question and document your reasoning process even if you cannot quite get there! In this way you can get at least partial credit!

Normally we ask you to explain your answers in text. However, for the basic programming questions it is not really necessary.

1 Lists (6pt)

1. (2pt) Create a list 'movies' that contains the names of at least six movies you like
2. (2pt) Use **slicing**, and create a list of three first movies in the list
3. (2pt) Use slicing and list concatenation to create a list of the first two and the last two movies

2 Loops (6pt)

1. (2pt) Create a list 'numbers' that is the numbers 70 through 79 in the following manner: create an empty list and add numbers to it in a loop (do not use list comprehension or 'list' function here!)
2. (2pt) Use loop to compute sum of all integers from 1 till 100.
3. (2pt) Assign people to seats. Consider names *Adam*, *Ashin*, *Inukai*, *Tanaka*, and *Ikki*; and seats 33, 12, 45, 2 and 17. Print seat assignments by assigning each name to the corresponding seat. You coude should output

   Adam: 33
   Ashin: 12
   ...

   Hint: loop over the integer range of the length of names (here 0..4, remember–python indexing is 0-based), and use indexing to access the corresponding name and seat number.

3 Comprehensions (4pt)

1. (2pt) Use *list comprehension* to create a list of squares of numbers 1..10 (i.e. 1, 4, 9, ..., 100)
2. (2pt) Use *dict comprehension* to create a dict of numbers 1..10 and their squares (i.e. 1:1, 2:4, ..., 10:100).

Hint: Read Lubanovic *Comprehensions*, p 81

4 Functions and data transformations (8pt)

1. (8pt) Write a function that takes in time in the numeric form (i.e. not a string) of HHMM (hours-minutes), and returns it in the numeric form of HH.HH (hours + fractions of hours). For instance, 1015 → 10.25 (10 hrs 15 mins → 10.25 hrs). Demonstrate it works using values 1015 and 345.

   The function should *return* (not print) the result as a *number*.

   Hint: use modulo operator % and integer division operator //. Modulo of 100 gives you minutes and integer division by 100 gives you hours

The following questions ask you to use gapminder data. The data is complied from https://www.gapminder.org/data/ but use rather the ready-made file on canvas. The variables are the following:

name country name

iso3 3-letter country code

iso2 2-letter country code

region broad geographic region

sub-region more precise region intermediate-region

time year

totalPopulation total population

GDP_PC GDP per capita (constant 2010 US$)

accessElectricity Access to electricity (% of population)

agriculturalLand Agricultural land (sq. km)

agricultureTractors Agricultural machinery, tractors (count)

cerealProduction Cereal production (metric tons)

feritilizerHa Fertilizer consumption (kilograms per hectare of arable land)

fertilityRate total fertility rate (births per woman)

lifeExpectancy Life expectancy at birth, total (years)

childMortality Mortality rate, under-5 (per 1,000 live births)

youthFemaleLiteracy Literacy rate, youth female (% of females ages 15-24)

youthMaleLiteracy Literacy rate, youth male (% of males ages 15-24)

adultLiteracy Literacy rate, adult total (% of people ages 15 and above)

co2 CO2 emissions (kt)

greenhouseGases Total greenhouse gas emissions (kt of CO2 equivalent)

co2_PC CO2 emissions (metric tons per capita)

pm2.5_35 PM2.5 pollution, population exposed to levels exceeding WHO Interim Target-1 value 36ug/m3 (% of total)

battleDeaths Battle-related deaths (number of people)

5 Load data (8pt)

1. (2pt) Load the csv data file
2. (6pt) Do basic sanity checks:
   (a) How many variables (columns) is there in the data? Ensure you know the variables in the data. Keep the documentation nearby.
   (b) How many rows of data is there?
   (c) print the first few lines of data. Does it look reasonable?

   Through the course we expect you *always* to do a similar sanity check each time you load data.

6 Wealth (36pt)

First, let's do some data exploration. Answer the following questions: show the code, the computation results, and comment the results in the accompanying text. Remember: you have to explain your results as appropriate! Your task is not just to get correct answers but to convince us that you understand what you are doing.

1. (2pt) How many different countries are there in the data?
2. (1pt) What is the earliest and the most recent year in the dataset?

Now let's define wealth as GDP per capita and let's explore countries by average wealth.

3. (2pt) For which year do we have the most recent GDP data?

   Hint: You can remove all cases where GDP is missing.

4. (3pt) What is the average wealth on this planet as of 2019? Let's just compute average GDP across all countries in 2019 and ignore the fact that countries are of different size.

5. (6pt) But not all countries may have the same most recent year where GDP data is present. Which 5 countries countries have the largest number of the most recent years missing? (For instance, imagine there is GDP data for Funan for 2015, 2016, and 2017; but for Khmer Empire only for 2015. Hence Khmer Empire has two of the most recent years missing.) Till which year do these countries have data? What do you think, why do these countries have issues with more recent data?

   Hint: you may group by country and find max value for the year. In the resulting series, find the min/max. Check out the `nlargest` method.

   Hint2: two of these countries are Lichtenstein and Faroe Islands.

6. (6pt) Now let's compare the continents. We'll make it easy again and just compute the average wealth (i.e. GDP) for each continent in 2019, and we use *region* as continent. We disregard the fact that countries are of different size. Print the continents, and the corresponding GDP in a decreasing order. Do you think this order is reasonable?

   Remember to use only the most recent data!

   Hint: check out methods `groupby` and `sort_values`.

   Hint2: you should see value around 11,800 for Oceania.

7. (6pt) But this was just about the average numbers. Now for each continent let's also find the richest and the poorest country, the corresponding GDP, and population (2019).

   Note: While this gives a hint about inequality, we still ignore the intra-county inequality. Quite likely the rich in the poor countries earn more than the poor in the rich countries. Unfortunately we cannot tell based on these data.

   Hint: if you can find a good solution yourself, go for it! But here is a suggestion how you can do it, step-by-step.

   (a) Check out indexing with `.loc` and `.iloc` (McKinney, p 143)
   (b) Check out methods `.idxmax` and `.idxmin` (McKinney p 158-159)
   (c) Find the index of the largest GDP value in data (use `.idxmax`).
   (d) Extract the row of dataframe that corresponds to the index you computed in 7c
   (e) Now repeat the two previous steps either with `.groupby`, or if you cannot figure this out then with a for-loop over all continents.

Alternatively, you can extract the values using construct like `data.gdp == data.gdp.min()`. You can also just loop over continents, and for each continent find the richest and poorest country as of 2019 (check out methods `nlargest` and `nsmallest`).

8. (6pt) Finally, let's put all this into a single data frame. The data frame should contain continent name as index, and for each continent 6 variables: name, GDP and population for the poorest country, and the same for the richest country. Pick suitable variable names!

   The result might look something like:

```
         poorest   GDP         population  richest     GDP          population
region
Africa   Burundi   208.07473   11530580.0  Seychelles  15048.74693  97625.0
...
```

   Hint: consult Python notes 4.6: Combining data into data frames. You can use `.set_index` to convert a variable into index.

9. (4pt) Comment the list of poorest and richest countries. What do you think about these lists. Did you know that Bermuda is the richest country in Americas? Do you know why? Why do most of the rich countries have small population?

7 Health (32pt)

This is a more demanding problem. Health is a complex concept, but fortunately we can proxy health with life expectancy (LE). It is a natural index of health that has been measured rather well for decades, or even centuries.

1. (3pt) How many countries do not have LE data for year 1960? How many countries do not have this information for year 2019?

2. (4pt) What is the shortest and longest LE in data? Which years/countries does this correspond to?

   Hint: you can use methods `nlargest` to extract dataframe rows, and `idxmax` to extract index of row with the maximum value.

3. (4pt) If you did this correctly, you notice that the shortest LE is less than 20 years. What historical events does it correspond to? (You may consult Wikipedia).

4. (6pt) Find the country with longest and shortest LE for each continent.

   Hint: you can use the same approach as for GDP by continent.

5. (4pt) Now lets find, for each country, the first valid LE. Find the corresponding year, and LE value.

   Hint: you can use the same approach as when finding the largest GDP on each continent: find the index of minimum year of each country, and then extract the rows based on the index.

6. Now repeat this with the most recent valid LE.

7. (7pt) Finally, let's compute the growth rate. You can compute the growth rate (pct per year) as

$$g = 100\left[\left(\frac{LE_1}{LE_0}\right)^{\frac{1}{n}} - 1\right] \tag{1}$$

where LE_0 is the life expectancy at the beginning of the period, LE_1 is it at the end of the period, and $\mathfrak{n}$ is the length of the period in years.

Show the countries, drowth rates, and the corresponding life expectancies for 10 fastest and 10 slowest growers.

Hint: for each country, compute the first valid year of life expectancy, last valid year of life expectancy, and find their life expectancies for the corresponding years. You can do it in a fairly similar way as in the previous question. Now use this formula for each country. At the end just order the result.

8. (4pt) Do you see a pattern (or multiple patterns) here? Remember: you are looking at growth of life expectancy over an extended period.

How much time did you spend?

And finally-finally, tell us how much time (how many hours) did you spend on this PS!

References

Lubanovic, B. (2014) *Introducing Python: Modern Computing in Simple Packages*, O'Reilly Media.