INFO370 Problem Set: Python, data frames (100 pt)

January 11, 2023

Instructions

This is the first problem set. It is worth 100 "PS points", that is equivalent to 10 points of your 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 functions, lists and dicts. It also requires very basic data frame tools but more thorough data exploration will be left for the next PS.

Some background reading will be very helpful for those of you less familiar with Python: the Lubanovic (2014) book, chapters 2 (data types), 3 (lists, dicts, sets), 4 (code structures); and McKinney (2018), Ch 4 and 5. Alternatively, the basics are also explained in python notes `http://faculty.washington.edu/otoomet/machinelearning-py/python.html` in a much less thorough manner. But be aware that python notes are not a substitute to a thorough background readin (like Lubanovic), and google search is a substitute for neither (the opposite is also true). In order to know a coding language well you need to understand the basic concepts, in class we will use a very limited set of tools. Google is great when you know what to ask, but what to ask–this is what books are for.

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!

1 Base python (54pt)

Normally we ask you to write textual answers and explain the results. However, in this basic programming task this may not be necessary.

1.1 Computing (7pt)

Compute the following numbers, assign those to suitably named variables, and print:

1. (1pt) How many seconds are there in year?
2. (1pt) How many seconds is a typical human lifetime? Use the previously computed seconds in year.
3. (1pt) What is your age in seconds? Compute this based on your age.
4. (1pt) Create a logical variable that is true if you are more than 700M seconds old. Use logical operators, not if/else!
5. (3pt) Compute the following mathematical formulas:

$$p(1-p) \qquad \text{where } p = 0.4 \tag{1}$$

$$\frac{\alpha}{x_0}\left(1+\frac{x}{x_0}\right)^{-\alpha-1} \qquad \text{where } \alpha = 3,\ x_0 = 2, \text{ and } x = 2.5 \tag{2}$$

$$\frac{\alpha x_0^2}{(\alpha-1)^2(\alpha-2)} \qquad \text{where } \alpha = 3 \text{ and } x_0 = 2 \tag{3}$$

   (This is more about understanding math and less about coding.)

   Hint: the answer for (2) is approximately 0.0585, and (3) should give you a nice simple number.

1.2 Lists (8pt)

Consult python notes, ch 2.4.1 *Lists*.

1. (1pt) Create a list ‘movies‘ that contains the names of at least six movies you like
2. (2pt) Using indexing and **slicing**, 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
4. (3pt) Use *slicing* to list the movies in the opposite order

---

1.3 Loops (12pt)

1. (3pt) 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!)

   Hint: Consult python notes, ch 2.4.1 *Lists*.
2. (3pt) Use a loop to compute the mean value of the list: add the values to their sum in a loop, and at the end divide the sum by the number of the values. (Do not use `mean` function!)

3. (3pt) Use loop to compute sum of all integers from 1 till 100.

4. (3pt) 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. For example, you can output

   Adam: 33
   Ashin: 12
   ...

   Hint: loop over the integer range of the length of names (here from 0 to 4, remember–python indexing is 0-based and thus counting doesn't start at 1!), and use indexing to access the corresponding name and seat number.

1.4 Dicts and sets (13pt)

Consult python notes, ch 2.4.3 *Dicts* and 2.4.4 *Sets*.

1. (10pt) Create a word frequency table using dicts that take a looong string (feel free to copy-paste some text here), splits it into individual words, and counts the number of occurrencies of each word (and prints the result). Let's ignore punctuation. Your code should run over individual words and increase the counter for that word, stored in a list. It has to check if a word exists in the dictionary, and if not, take an appropriate action.

   Hint: you can do it as follows:

   (a) convert everything to lower case
   (b) use the split() method to get words: `"I have no clue".split() -> ['I', 'have', 'no', 'clue']`
   (c) use triple quotes `"""` to create long multi-line strings
   (d) create an empty dict of word counts
   (e) loop over words, and increase the count for each word. But you have to handle the words that are not yet in the dict somehow.

2. (3pt) Use *set* to find how many different words you have in your text above.

   Note: you can get this information from your word frequency dict too, but this task is about using set.

1.5 Comprehensions (6pt)

Consult Lubanovic *Comprehensions*, p 81 to learn about comprehensions.

1. (3pt) Use *list comprehension* to create a list of squares of numbers 1..10 (i.e. 1, 4, 9, ..., 100)

   Hint: Consult python notes, ch 2.4.1 *Lists*.

2. (3pt) Use *dict comprehension* to create a dict of numbers 1..10 and their squares (i.e. 1:1, 2:4, ..., 10:100).

1.6 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

---

2 Basic data frames (46pt)

Your second task is to analyze flights out of NYC airports in 2013.

Here explanations are a major part of the solution:

- Your results will only count if accompanied with sufficiently and clear explanatory text. Just plain output, with no explanation, will not count.
- Be sure that each visualization (graph or table) adds value to your written explanation; avoid redundancy – you do not need four different visualizations of the same pattern.
- Don't output irrelevant, or too much of relevant information. A few figures is helpful. A few thousand figures is useless. In particular, don't print all thousands of lines of data!

2.1 Setup (5pt)

In this problem set you will work with NYC flights data. The data is copied from the corresponding R package, you can read the documentation at e.g. RDocumentation. I guess the data originates from DOT Open Data Catalog, but seems like it has later been removed from that website.

Make sure you have read the background readings about pandas (see above).

1. (2pt) Load the data
2. (3 × 1pt) Do basic 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 that you always do similar checks every time you load data.

2.2 Basic data exploration (19pt)

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 as appropriate.

1. (5pt) How many NYC airports are included in this data? Which airports are these?
2. (4pt) Into how many airports did the airlines fly from NYC in 2013?

3. (5pt) How many flights were there from NYC to Seattle (airport code *SEA*)?

4. (5pt) Were there any flights from NYC to Spokane (GEG)?

2.3 Flight delays (32pt)

1. (4pt) Create a new data frame, a subset of the flights data that only contains two variables: departure delay and arrival delay.

   Use this data frame for the following questions.

   Hint: read python notes 3.3 *Indexing data frames and series*

2. (2pt) Print a small random sample of the delay data frame.

3. (3pt) What is the data type of the flight delay value?

   Hint: read python notes 5.1 *First steps: know your data*

4. (5pt) Filter to keep the "slow flights": cases where the plane leaves early but arrives late. How many cases do you have?

   Hint: about 35k

5. (5pt) What is the longest "extra flight time", i.e. difference between arrival delay and departure delay?

   Hint: you can create a new variable that is the extra delay, and find the max value of it.

   Hint2: read python notes 5.2 *What are the values?*

6. (4pt) Are there any implausible flight delay values?

7. (5pt) How many of flights were delayed (positive arrival delay)?

8. (5pt) What percentage of flights were delayed more than 15 minutes (at arrival)?

   Hint: slightly more than 23%.

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.

McKinney, W. (2018) *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, O'Reilly Media, 2nd edn.