INFO370 Problem Set 1: Python, data manipulations

March 30, 2022

Instructions

This is the first problem set. It is worth 8 points, or 8 percent 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 lists and dicts and functions. Some background reading will be quite helpful for those of you less familiar with Python: the Lubanovic (2014) book, chapters 2 (data types), 3 (lists, dicts, sets), 4 (code structures). The basics are 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 Basic Python (30 percent)

1.1 Lists

See Python Notes Ch 2.4.

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

---

1.2 Loops

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

1.3 Comprehensions

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

1.4 Functions and data transformations

1. 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 Numpy and Pandas (40 percent)

The following problems focus on the basic functionality of numpy and pandas. We recommend you consult the Python Notes, Ch 3 and Ch 4. *Do not* use loops below, except for 2.1.5.

2.1 Series

1. Use `np.arange` to create two arrays: integers $1, 2, \ldots, 10$ and squares of these integers.
2. Create a series where index is integers $1, 2, \ldots, 10$, and values are the corresponding squares.
3. Extract 3^2, 6^2 and 9^2 from the series using the index (not location!). Do this as a single operation that results in a single series.

   Hint: the result should look something like

   ```
   3     9
   6    36
   9    81
   dtype: int64
   ```

4. Now convert the result (the series of three squares) into a numpy array.
5. Use loop to fill a series with *square roots* of integers $1, 2, \ldots, 10$, where the index are the corresponding integers. First construct a series of arbitrary numbers with a given index; then loop over the index and fill the values with corresponding square root. *Use index, not location* when filling the value!

   Hint: use *np.sqrt* to compute square root (easier than *math.sqrt*).

   Note: in a task like this one you should use vectorized operations, not a loop. But doing this with loops is useful if the operations are more complex that just square root.

2.2 Data frames

1. Create a data frame with index being the integers $1, 2, \ldots, 10$, and two variables: squares and square roots of these numbers.
2. Add another column–logarithm of the number–to the data frame.

   Note: use natural logs (`np.log`), not decimal logs.
3. Extract only rows where root is at least 2.0 but log is less than 2.0 of the data frame.
4. Add a string variable, the integer number values in lower case words ("one", "two", ..., "ten").
5. Capitalize the words ("One", "Two", ...). Use the `.str` attribute to access vectorized string operations.

   Hint: see Python Notes 4.6 and Pandas' documentation.

---

3 Explore Data (30 percent)

The following questions ask you to use gapminder data. The data is complied from https://www.gapminder.org/data/. However, please use the dataset provided on canvas. Some explanations of the variables are as follows:

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)

1. Load the csv data file
2. 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.

3. Extract a subset of data which includes only Ukrainian entries, and the following variables: the 2-letter country code, population, cereal production, and time.

   How many cases do you get?

4. Rename "time" to "year" in the Ukrainian data subset.

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.