FOR THE R WORKSHOP on March 8 PLEASE GO TO THE FOLLOWING LINK:

https://pad.carpentries.org/notre-dame-hesburgh-libraries-2021-03-08

Thursday, March 4, 2021

- **Instructor**(s): Ben Chiewphasa
- **TA**(s): Ben Tovar
- **Facilitator**: Julie Vecchio

Attendance & Introductions

Please fill in your name in the top right corner so that we can easily tell what you type. Feel free to choose your own color as well!

Thursday, March 4, 2021

- **Instructor**(s): Ben Chiewphasa
- **TA**(s): Ben Tovar
- **Facilitator**: Julie Vecchio

Attendance & Introductions

Please fill in your name in the top right corner so that we can easily tell what you type. Feel free to choose your own color as well!

- Sherryen
- Bo Wei Cynthia Chen
- Himani Patel
- Lan Dinh
- Meredith Lee
- Michael
- Michael Deike
- Rachel
- Sherryen
- Sunela

- **Anaconda**: platform that installs platforms such as Jupyter Notebook and packages that are integrated in working within python environment; like a woodshop with lots of tools within it

---

Notes & Exercises

- **Agenda**
  - 1st Hour underlying logic
  - 2nd Hour PANDAS library

PART 1

- **Vocabulary**
  - **Anaconda**: platform that installs platforms such as Jupyter Notebook and packages that are integrated in working within python environment; like a woodshop with lots of tools within it
  - **Jupyter**: open source web app allows you to share docs, do live coding, nice graphical user interface (GUI) so easier to work with Python than in the command line; coding workbench in Anaconda workshop
  - **Python**: a programming language that enables rapid development of data analytics, e.g.,; swiss army knife on the coding workbench in anaconda workshop
  - **pandas**: softwarea library written for Python language with goal of manipulating and analyzing data; like one of the individual tools on the swiss army knife
  - **IDE**: integrated development environment (to make programming easier to work with more streamlined, our IDE today is Jupyter); save code, cut, run sections of the code, etc.
  - **Spyder**: is a python IDE;
    - **Q**: Why choose one over the other?
      - Solely python while Jupyter incorporates other languages like R, Julia, e.g.
      - Mostly just python editor, with Jupyter there is a strong bias toward communication so you can construct documents for sharing and give to coworkers.
    - **Q**: Spyder is helpful for my work since I can see all the variables (sort of like maltab). Jupyter doesn't have the same function right?
      - It does not look that it does that out-of-the-box, but you can add extensions to it. There is an inspector extension that seems to do just that.
- **Why use Python?** Python is easier to learn than some other languages, which is great b/c more accessible for folks to use it to work with data without having to learn more complicated languages before being able to get started; versatile, reproducible (on other platforms, e.g.).
- **Open Anaconda**
  - Conda update jupyter
  - New Notebook Python 3 (should open new tab)
- ***Object/variable may be used interchangeably**
- a = 2 or b = 3.75 --doing this is assigning an integer value to that a or b object; if you run these lines of code is that a value is given a memory of 2 and b a memory of 3.75.
  - **In addition to run button can use keystrokes** (icon like tiny keyboard offers way to adjust key mapping--so you can assign a keyboard shortcut to perform and action you will perform a lot)

        - Command mode = the active spot in your notebook (color changes from green to blue around the "live" box, and the outline of the box changes shape); in this mode you can use keyboard shortcuts that are tagged "command mode".
            - **Q**: In previous cell can you call the contens of that "live" box (e.g., [1])?
                - No, for objects, but not lines of code in the "live" boxes you see.
                    - So does not class them automatically. No.
                    - Updated answer: It seems that you can! I did not know either, but it seems you can do, e.g., _i1 for the input of cell 1, or _1 for the otuput cell 1.
- Create new cell (live box) to try out **type** function
    - in[2] : type(a) control+enter
    - out[2] : int
- **Parenthesis holds an argument**
- Another function is **print**
    - In [3] : type(a)
    - print(type(a))
    - <class 'int'>
    - If change to b and re-run, class would be float type
        - what is different from **float and integer**?
            - float includes decimals, takes up more memory
    - If change a from 2 to 2.0, and check type again, it would change from int to float
- **Arithmetic operators**
    - a+b #addition
        - # is to comment in your code (explain how or why things are coded to work/document your code for future you, or collaborators!)
        - output will be 5.75 (2.0 + 3.75)
    - a * b #multiplication
        - output will be 7.5
    - why doesn't addition show up?
        - does output only show last thing to be computed? Correct!
    - a - b #subtraction
        - output -1.75
    - **what if do want more than just the last input result to show up? You can utilize the print function**
        - print(a + b)
        - print(a * b)
        - print(a - b)
        - outputs
            - 7.75
            - 7.5
            - -1.75
    - b ** a #exponent
        - outputs 14.0625
- sevenPointOne = "7.1"
    - **creating a new object**
    - **Python is case sensitive, be aware!**
    - print(sevenPointOne) outputs what's in the parentheses 7.1
- **Strings**
    - items in quotes (characters, not numeric values)

        - print(type(sevenPointOne)) outputs <class 'str'>
        - **single quotes and double quotes work just fine/the same**
- **Changing data types is possible but can open up a can of worms to be mindful of**
    - duplicate = 7.1 (since no quotes, it's a float b/c it's a number with a decimal point)
    - If want to change to integer, duplicate = 7.1
    - int(duplicate)
    - print(duplicate)
    - output is 7.1, why?
        - made object w/7.1 value, ran function to change from float to integer and still got 7.1 b/c line of code ran something but didn't save it to anything, so have to do this, instead:
            - duplicate = 7.1
            - ***duplicate = int(duplicate)***
            - print(duplicate)
            - output is 7
    - so bear in mind that it can be dangerous to change data type b/c could lose information (like the example above where 0.1 disappeared)
- **Q**: When assigning objects, you only need to use an equal sign, there are no other previous declaration needed?
    - Correct, you don't need to declare things first.
- **Example where may not make sense to change data type**
    - b = "Hello World" (this is a string, if want to change to an integer, won't happen because won't make sense to change a character/word to integer and you'll get an error message)
- **In programming, it's guaranteed that you will get error messages**
    - Can seem overwhelming, they're long and confusingly written, sometimes in scary colors, but that's OK! It happens to us all! :)
    - Take a step, breathe, and look at the error to see what information in there can help us to troubleshoot, look up online!
    - In python usually it will point you directly to the location of the issue to help you find trouble spots!
- **Boolean data type**
    - Boolean values (True and False)
    - bool_val_t = True
    - bool_val_t = False
    - print(bool_val_t)
    - Can get these outputs when you use comparison operators
        - == is a check for equal to
            - print('hello' == 'HELLO')
                - does hello equal HELLO
                    - output is False
        - != is not equal to
            - print(524345557 != 66)
                - output is True
- **Sometimes when troubleshooting, restarting can work, just like with a computer--turn it off and one again!**

PART 2 pandas

- **First, need to make sure python is pulling data from where we want it to + import pandas**
    - In a new cell of code, type **import os**, in second line **os.getcwd()**, and run that

    - Output should be a filepath with the working directory jupyter/python is using, something like C:\\Users\name\documents\etc.
    - Now please move all the data files downloaded from the setup page to wherever your output says the working directory is (like from the bullet above)
    - In a new cell, type/run: **Import pandas as pd**
        - this is telling python to import a python library called pandas and the as pd is interpreted as in my environment I'm calling this pandas library pd. pd can be anything! You could call it lolololol, or whatever you want--just make it something reasonable to type/remember, etc. pd is default best practice.
- **We are working w/pandas bc it's really a good environment/tool to use for data frames (tabular data types, analogous to working with Excel sheets, a data format with collumn names and plenty of rows and CSV files)**
- **Let's import a dataset to our python environment memory so we can work with it**
    - create an object, pull from the pandas library we called pd this read_csv function (this read_csv function is associated with pandas library), name is SN7577, and it's tab-separated
        - df_SN7577 = pd.read_csv("SN7577.tab", sep-'\t')
    - Troubleshooting
        - Check location, check spelling, slashes, etc.
    - Output should be dataset (the table view/rows and columns) for df_SN7577
- **Q**: instead of moving the files to the same directory, could you also put that different directory into the read csv function?
    - Yes, you can! e.g., actually I did: df_SN7577 = pd.read_csv("data/SN7577.tab")
- **Q**: can you use space separated instead of tab separated?
    - yes, you can set it with the sep=" " parameter, instead of \t.
- **Q**: what if it is just a text file and not a csv file?
    - you would read it with the default python statement (which we do not cover here). Otherwise you most likely would get really weird errors. But it would something like:
    - with open("my.txt.file") as f:
    - for line in f:
    - print(line)
- **Q**: Why do we need to wrap the file name SN7577 as a string: "SN7577"?
    - because otherwise python would look for a variable called SN7577.
    - e.g.:
    - x = "my.csv.file"
    - df = pd.read_csv(x)
    - so that x is the name of an object that just happens to contain the string "my.csv.file".
- **df stands for dataframe**
    - df_SN7577.head()
        - gives us the first 5 rows of our data frame
    - Using a method -- similar to functions except tied to an object or class
    - df_SN7577.size
    - print(df_SN7577.size) outputs 259772
- **Another way to investigate our data is what happens if we're working with a data frame and analyzing your data and only interested in reading the data with certain columns, like 3 or 4 of the dataset, not 200 columns?** 1st colulmn (0 cause index) and 174 and 175
    - df_SN7577_somCols = pd.read_csv("SN7577.tab", sep='\t', usecols = [0, 174, 175])
    - print(df_SN7577_somCols) outputs a table with the 1st, 174, 175 columns only
    - I subsetted using the index number under usecols but can subset by names of columns, as

well; so instead of using 0, you can put in ['Q1', 'age', 'agegroups'] (be careful of capitalization when typing column names)

- **Q**: would you use userow for selecting certain rows?
    - for rows, it is a little the opposite. You would use skiprows and a list or rows that you don't want.
- **Subsetting by rows**
    - Filter by choosing range, e.g., df_SN7577_someRows = df_SN7577[1:4] (this means I'm interested in rows 2-4, b/c python indexes)
        - **Brackets in python is doing work related to subsetting, so selecting certain components of your dataset**
        - We can set criteria, as well, if interested in data that has a certain value or falls within a certain parameter set, like don't want values with -1
            - df_SN7577 = pd.read_csv("files\\SN_7577.tab", sep='\t')
            - df_SN7577_YesNeg = df_SN7577[(df_SN7577.Q2 == -1)]
                - Telling python: look at this dataframe I want to subset according to the square brackets I'm going to look into column name Q2 and I want to look for all rows with values of -1
                - Running that, add another cell: df_SN7577_YesNeg outputs new dataframe with all 898 rows and 202 columns and you can see that we're only getting the negative 1 value returned
            - when working in python using same object names, be careful not to overwrite your objects, so good practice is to change up names or be very clear with commentary with # what you're doing
- **Q**: why do you not need to quote Q2 in the function when it has letters?
    - So Q2 is actually name of a column it's not a string alue type, it's embedded as part of the data frame
- **Add another cell and show you an example of another way to subset**
    - df_SN7577_complex = [(df_SN7577.Q2 == -1) & (df_SN7577.numage >60)]
    - df_SN7577_complex returns 292 rows and
    - Load in another dataset df_SAFI = pd.read_csv("SAFI_results.csv") we're not using the separator argument b/c not working with .tab file anymore
        - df_SAFI.describe()
            - give summary statistics for collumns associated with your dataframe
            - count, mean, std, min, 25%50%75%,max
    - can also get this for a single column by subsetting
        - df_SAFI['B_no_membrs'].describe()
- **Bear in mind: When working with methods (similar to functions but tied to specific object) the parentheses sometimes have parameters, but sometimes not, some don't take in parameters at all but still need to put empty parentheses there**
- Large datasets, a lot of times realistically we'll be dealing w/ missing data, and python handles **missing data well**
    - Completely remove rows w/missing data, or assign missing values an actual value, e.g.
        - print(df_SAFI.shape) outputs 131 rows 55 columns
        - df_SAFI = df_SAFI[(df_SAFI['E_no_group_count'].notnull())] (not null means not missing)
        - run above, then print(df_SAFI.shape) outputs 39, 55, so rows have been decreased
    - That 0 or missing value could have context w/in data analysis, so maybe if 0 value is in our data, we want to incorporate that in our analysis so can change missing value type to 0 in python

- How you handle missing data depends on your anlaysis needs, etc.

Feedback survey:

https://docs.google.com/forms/d/e/1FAIpQLSf16kISboQwEojMwudeej38VaH3mvCtMv3noJt1Z1I_nMonfw/viewform?usp=sf_link

Resources

- Stack Overflow
- Python mailing list: https://www.python.org/community/lists/
- Hitchhikkers Guide to Python: https://docs.python-guide.org/
- Library materials (e.g., ebooks): https://onesearch.library.nd.edu/primo-explore/search?query=sub,exact,Python%20%5BComputer%20program%20language%5D&tab=onesearch&search_scope=malc_blended&vid=NDU&offset=0