Preprocessing Data in NumPy - FAQ

Preface

IMPORTANT – PLEASE DON'T SKIP THIS PART!

Here we will describe the process you need to go through whenever you encounter an error in programming. Please read the following two pages carefully, as they will surely save you a lot of time in the long run!

***A general note on problem-solving*:**

Even the best programmers out there find it nearly impossible to successfully create the complex code delivering the desired output the first time they write it. So, we can say coding is all about trying, stumbling upon an error, reading the obtained error message, finding a solution, and then applying it. Over and over again. In fact, this process is repeated infinitely.

Therefore, as a programmer, you need to develop the skill to quickly and efficiently identify and solve problems that occur while you are coding.

What should I do when I encounter a coding error?

With this question in mind, we will take you through the steps you should follow each time you encounter an error:

1) Read the error message carefully.

Every time you make a mistake, you will obtain an error message. Sometimes, this message will be quite specific; you will be able to immediately spot the mistake and correct it. Other times, though, it will sound more general. Then, it will be your job to find where the error stems from.

2) If you encounter an error while you are replicating code you've seen in a lecture, **please re-watch that video** or at least the parts about the query you are trying to execute. Remember that every symbol and letter could make a difference! So don't overlook the details – they might be the cause of this error!

3) If the error occurs while you are trying to solve an exercise, please carefully **check the notebook files** containing the corresponding solution.

4) In some situations, the 3 steps described above may not help. Or… you may still have a question, even though you've managed to solve the error. In that case, please **search among the existing threads in the Q&A section/hub**. If you can't find the answer to your question, you can post a new one there.

How can I speed up the process of solving the encountered problems?

1) While coding in Python, the order in which you execute the code cells matters. So, please double-check to make sure you have executed all cells in the order shown in the lectures or the notebook files.

2) Keep this FAQ sheet open, as it may contain the answer you are looking for. Use **Ctrl + F** to find a keyword related to the problem you are trying to solve quickly.

Questions related to Udemy and the smooth taking of the course

Apart from questions related to coding in Python, you may wonder how to load the notebooks located in the resources sections of the lectures, or how to obtain a certificate at the end of the course. We will deal with such questions in the next section.

Use the FAQ sheet, the Q&A, and good luck while taking the course!

General:

1. How can I open the exercise files and the *.ipynb files in general?

a. Keywords: **open, *.ipynb, exercise file, lecture file, solution file**

The notebook files (*.ipynb), located in the resources sections of the course lectures, cannot be opened by double-clicking on them. You need to start **Jupyter** (or any other environment we're using, like Spyder) and load the notebook files.

Then, you can apply this technique to any of the notebook files attached to the lectures.

2. Why is my quiz/video not loading/rendering?

Keywords**: quiz, video, not loading, not rendering**

Unfortunately, such issues occur from time to time.

You can try and refresh the page several times. If this does not solve your problem, you can wait for a while or contact our support team.

3. I have downloaded and installed Anaconda but I can't find and start Jupyter. What should I do?

Keywords**: install, Anaconda, launch, start, Jupyter**

There are three ways to start Jupyter.

1) Run *Jupyter Notebook* from the shortcut found in the Start menu (this is the option shown in the video).

2) Open *Anaconda Prompt* from the start menu and execute jupyter notebook

3) Open the *Anaconda Navigator* from the start menu and press the *Launch* button under the *Jupyter* icon

If 1) does not work, *Anaconda Prompt* will usually tell you what the error is, so you can look for a specific solution.

Please reinstall Anaconda in case none of these options works.

Mac users:

1) Open *Terminal*

2) Enter the startup folder by executing cd/folder_name

3) Execute jupyter notebook

4. Why do we stick to conventions in Python? (e.g. import pandas *as pd*)

Keywords: **conventions, Python, packages, pandas**

An example of a convention is calling the web data web or wb. In our course, we obviously stick to the second one.

But in terms of Python code, what does wb refer to? It refers to the data part of the pandas-datareader package, right? That's why we begin our code with

from pandas_datareader import data as wb.

Having executed this line of code in your session ("session" meaning the period in which you will keep your notebook and Jupyter open), Python will know that wb refers to the data part of the pandas-datareader package. This is nothing but a large chunk of code where certain functions and methods have been specified.

Therefore, wb.DataReader() is simply an indication where Python should search for a method called .DataReader().

Other conventions are plt for *pyplot matplotlib* code, np for *numpy*, and pd for *pandas*. It is a sign of good practice to stick to conventions such as np, plt, wb. That's why we use them in the course and advise you to use them, too. Additionally, if you're ever looking for some help in StackOverflow, most (if not all) solutions align with these conventions. Thus, the more you know and apply them, the better you'd understand somebody else's script (code).***Note:*** *Methods, on the other hand, have their fixed names, and you cannot really use an alias for them. You'd have to create a new function or method, based on the initial method, and give it a name you like. However, this is not something people normally do. Therefore, it is better to try to remember the method or function name you need.*

5. I opened the *Resources* link accompanying the lecture, but I can't make out what each file does. What files do I need for this lecture/section/exercise?

Keywords: **template, complete, notebook. .ipybn, .csv, exercise, solution, file**

In the *Resources* link accompanying this lecture/section, you'll find all the additional files you need to replicate the code in the video.

***-Template.ipynb** - These are "empty" Notebooks you can use to code along as we go through the lectures.

***-Complete.ipynb** - Filled-out versions of the code we typed out over several lectures (or a full section) which you can use to check what we did, how we did it and why we did it. There are additional comments to help you further understand how everything works.

***-Exercise.ipynb** - Notebooks which have some code filled out, but are mainly "empty" and only contain tasks/problems/exercises for you to replicate and test how well you understood the previous lectures.

***-Solution.ipynb** - Notebooks which contain one possible solution to the tasks assigned in the **-Exercise.ipynb* files.

***.csv** - Datasets which we will most likely import, manipulate and analyze in Jupyter.

Coding in Python:

1. I have all my Python notebooks and data files in the same directory but cannot load them. What should I do?

Keywords**: import file, CSV, directory, full path**

This is an issue that occurs due to your specific Anaconda installation. The easiest way to solve this is to reinstall Anaconda. However, we recommend that you use the **absolute path** of a file when loading the data.

Note: *We can use the full path, even if our notebooks (*ipynb) and data files (*csv) are in the same directory. In that case, even if we move the Python notebook to a different folder (directory) on our machine, the code should still run without any errors.*

Thus, you can write:

```
data = pd.read_csv('ABSOLUTE_PATH/real_esate_price_size.csv')
```

To me this looks like:

data = pd.read_csv('C:/Users/365/Desktop/The Data Science Course 2020 – All Resources/Part_4_Advanced_Statistical_Methods_(Machine_Learning)/S27_L142/real_estate_price_size.csv')

In your case, you can find that by opening the folder containing the files and copy-pasting the path.

Once you copy-paste this path, you should **CHANGE ALL SLASHES from backward to forward.** This means that if your path is C:\Users\YOUR_COMPUTER_NAME\Desktop\... , you should make it look like: C:/Users/YOUR_COMPUTER_NAME/Desktop/…

Note that, due to the standards for referencing paths, in Python, instead of a backward slash (\ ), you should use a forward slash (/).

2. What is the difference between a function and a method?

Keywords: **method, function**

Methods are called on specific instances of a given class (e.g. array_a.sort()), whereas functions don't. We call functions and provide the inputs within the parentheses (e.g. np.sort(array_a)).

Syntax-wise, functions usually appear at the start of the line, while methods show up at the end, and require a ".".

Since methods always ever refer to a specific object (e.g array_a), they sometimes store the output in place of the original variable. In other words, methods **can** delete the initial contents of the variable and store the output in their place. However, this varies from method to method (some even have *inplace* arguments which can be set to either True or False), so you it's a good idea to quickly check the documentation before using a method. In contrast, functions usually don't overwrite the original variable, so make sure to store their outputs if you plan on using them later on.

3. Why am I getting the error "object not callable"?

Keywords**: object not callable, attribute, method**

This occurs when we accidentally use *method syntax* (e.g. *variable_1.shape()*) to refer to an *attribute* (*variable_1.shape*).

When this happens, removing the parentheses should usually resolve the error.

4. What is the difference between "load" and "import"?

Keywords: **load, import**

Loading a dataset implies it's already properly formatted, and we can directly use it to compute mathematical operations. However, importing a dataset means we need to clean and preprocess it first.

Note: *We can still clean and preprocess a loaded dataset,* ***if*** *we wish to. However, we must always clean and preprocess a dataset we're importing.*

5. What is the difference between an error and a warning?

Keywords: **error, warning**

An error message means we're not able to run a specific line of code, so our program crashes. There are plenty of actions which can lead to an error, including: incorrect syntax, inappropriate type conversion (casting), mismatching shape/size of the input variables,

runtime errors caused by infinite loops/inefficient code, etc. These all stop the code from compiling and prematurely terminate the script.

In contrast, warning messages are just that – warnings. They don't prevent our code from running, but bring attention to a non-vital issue we might be overlooking. Examples of errors might be: overwriting columns of a series without using the *.loc()* or *.iloc()* methods, feeding empty lists/arrays/series to a function or method, using a function or method which will be removed in future versions of a package etc. The purpose of errors is to notify the user that there exist inefficiencies in the code. However, since they don't affect the output, we can even suppress them by writing the following:

6. I'm examining the results of a function/method, but the numbers look weird (e.g. 9.4529600e+03). **What does this mean and how do I read the results?**

Keywords**: scientific notation, e**

This is called scientific notation and we often see it among many programming languages. The purpose is to provide greater precision when we're working with very large/small numbers.

The true value of any such numbers equals the value to the left of "e", multiplied by 10, to the power of the value on the right side of "e".

Hence, 9.4529600e+03= 9.4529600 * 10^3 = 9.4529600 * 1,000 = 9,452.96.

If the number on the right side of "e" is negative (e.g. 5.67e-04), then we're diving by 10 to said power.

For example, 5.67e-04 = 5.67 * 10^(-4) = 5.67/(10^4) = 5.67/10,000 = 0.000567.

If we prefer to see the actual values on screen, rather than their corresponding scientific notation, we can suppress this type of notation. One way to achieve this is by calling NumPy's set_printoptions() function and set the *suppress* argument to *True*.

Note: *To use this function, we must also import numpy as np beforehand.*

NumPy:

1. What is the difference between an array and an ndarray?

Keywords**: array, ndarray**

There are different array types across various programming languages. Ndarray is the most well-known and widely used array class in Python. It is part of the NumPy package, works elementwise and extremely computationally stable, so we often use ndarrays to compute mathematical operations.

NumPy also contains character arrays, but they are only left for backwards compatibility with legacy code. If we ever need to store text data in a NumPy array, we'll do so by using an ndarray, where every individual element is a string.

2. Whenever I display larger arrays with more than 6 columns, the values of each 1-D array are displayed on several lines, so the first and 6-th column overlap. What can I do to make this more legible?

Keywords: **display large arrays, overlapping columns**

Whenever we're using NumPy and want to make our outputs easier to read and more coherent, we can use the *set_printoptions()* function.

By calling this *set_printoptions()* and setting the *linewidth* argument to a value like 100, 150, or even 200, we're increasing the number of elements we want Python to fit in a single line of outputs.

Now, for the remainder of this session (kernel), Python will fit 100 elements of the output on a line, before moving to the next.