labassignment1

January 19, 2023

1 Lab Assignment 1: How to Get Yourself Unstuck

1.1 DS 6001: Practice and Application of Data Science

Created: 01/18/23 by Tom Lever

Updated: 01/18/23 by Tom Lever

1.1.1 Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. Format your answers in a Jupyter notebook. To receive full credit, make sure you address every part of the problem, and make sure your document is formatted in a clean and professional way.

1.1.2 Problem 0

Import the following libraries:

```
[1]: import numpy as np
     import pandas as pd
     import os
     import math
```

1.1.3 Problem 1

Python is open-source, and that's beautiful: it means that Python is maintained by a world-wide community of volunteers, that Python develops at the same rate as advancements in science, and that Python is completely free of charge. But one downside of being open-source is that different people design many alternative ways to perform the same task in Python.

Read the following Stack Overflow post: https://stackoverflow.com/questions/11346283/renaming-columns-in-pandas/46912050. The question is simply how to rename the columns of a dataframe using Pandas. Count how many unique different solutions were proposed, and write this number in your lab report. (Hint: the number of solutions is not the number of answers to the posted question.)

Remember: your goal as a data scientist needs to be to process/clean/wrangle/manage data as quickly as possible while still doing it correctly. A big part of that job is knowing how to seek help to find the right answer quickly. Given the number of proposed solutions on this Stack Overflow page, what's the problem with developing a habit of using Google and Stack Overflow as your first source for seeking help? (2 points)

Answer There are suggestions to use documentation. There are suggestions to use `df.rename()`. There are suggestions to use `df.set_axis()`. There are suggestions to use `df.columns`. There are suggestions to use `pd.concat()`. There are suggestions to use `pd.DataFrame()`. There are suggestions to use `transpose` and `set_index()`.

A problem with developing a habit of using Google and Stack Overflow as my first sources for seeking help is that I would need to sift through many Google search results, Stack Overflow articles, Stack Overflow answers, and Stack Overflow proposed solutions, which have relevance secondary to docstrings and API documentation.

1.1.4 Problem 2

There are several functions implemented in Python to calculate a logarithm. Both the `numpy` and `math` libraries have a `log()` function. Your task in this problem is to calculate $\log_3(7)$ directly (without using the change-of-base formula). Note that this particular log has a base of 3, which is unusual. For this problem:

- Write code to display the docstrings for each function.
- Read the docstrings and explain, in words in your lab report, whether it is possible to use each function to calculate $\log_3(7)$ or not. Why did you come to this conclusion?

If possible, use one or both functions to calculate $\log_3(7)$ and display the output. (2 points)

Answer It is not possible to use `numpy.log` to calculate $log_3(7)$ directly. The description of `numpy.log` describes that `numpy.log` calculates a "Natural logarithm". The description of the parameters of `numpy.log` does not include a description of a base of a logarithm. dwitvliet, polvoazul, and their supporters in *NumPy: Logarithm with base n* (https://stackoverflow.com/questions/25169297/numpy-logarithm-with-base-n) imply that it is not possible to use `numpy.log` to calculate $log_3(7)$ directly.

```
[2]: import numpy as np
     np.log?
```

```
Call signature:   np.log(*args, **kwargs)
Type:             ufunc
String form:      <ufunc 'log'>
File:             c:
 ↪\users\tom\appdata\roaming\python\python39\site-packages\numpy\__init__.py
Docstring:
log(x, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None,␣
 ↪subok=True[, signature, extobj])

Natural logarithm, element-wise.

The natural logarithm `log` is the inverse of the exponential function,
so that `log(exp(x)) = x`. The natural logarithm is logarithm in base
```

```
`e`.

Parameters
----------
x : array_like
    Input value.
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it must have
    a shape that the inputs broadcast to. If not provided or None,
    a freshly-allocated array is returned. A tuple (possible only as a
    keyword argument) must have length equal to the number of outputs.
where : array_like, optional
    This condition is broadcast over the input. At locations where the
    condition is True, the `out` array will be set to the ufunc result.
    Elsewhere, the `out` array will retain its original value.
    Note that if an uninitialized `out` array is created via the default
    ``out=None``, locations within it where the condition is False will
    remain uninitialized.
**kwargs
    For other keyword-only arguments, see the
    :ref:`ufunc docs <ufuncs.kwargs>`.

Returns
-------
y : ndarray
    The natural logarithm of `x`, element-wise.
    This is a scalar if `x` is a scalar.

See Also
--------
log10, log2, log1p, emath.log

Notes
-----
Logarithm is a multivalued function: for each `x` there is an infinite
number of `z` such that `exp(z) = x`. The convention is to return the
`z` whose imaginary part lies in `[-pi, pi]`.

For real-valued input data types, `log` always returns real output. For
each value that cannot be expressed as a real number or infinity, it
yields ``nan`` and sets the `invalid` floating point error flag.

For complex-valued input, `log` is a complex analytical function that
has a branch cut `[-inf, 0]` and is continuous from above on it. `log`
handles the floating-point negative zero as an infinitesimal negative
number, conforming to the C99 standard.

References
```

```
----------
.. [1] M. Abramowitz and I.A. Stegun, "Handbook of Mathematical Functions",
       10th printing, 1964, pp. 67. http://www.math.sfu.ca/~cbm/aands/
.. [2] Wikipedia, "Logarithm". https://en.wikipedia.org/wiki/Logarithm

Examples
--------
>>> np.log([1, np.e, np.e**2, 0])
array([  0.,   1.,   2., -Inf])
Class docstring:
Functions that operate element by element on whole arrays.

To see the documentation for a specific ufunc, use `info`.  For
example, ``np.info(np.sin)``.  Because ufuncs are written in C
(for speed) and linked into Python with NumPy's ufunc facility,
Python's help() function finds this page whenever help() is called
on a ufunc.

A detailed explanation of ufuncs can be found in the docs for :ref:`ufuncs`.

**Calling ufuncs:** ``op(*x[, out], where=True, **kwargs)``

Apply `op` to the arguments `*x` elementwise, broadcasting the arguments.

The broadcasting rules are:

* Dimensions of length 1 may be prepended to either array.
* Arrays may be repeated along dimensions of length 1.

Parameters
----------
*x : array_like
    Input arrays.
out : ndarray, None, or tuple of ndarray and None, optional
    Alternate array object(s) in which to put the result; if provided, it
    must have a shape that the inputs broadcast to. A tuple of arrays
    (possible only as a keyword argument) must have length equal to the
    number of outputs; use None for uninitialized outputs to be
    allocated by the ufunc.
where : array_like, optional
    This condition is broadcast over the input. At locations where the
    condition is True, the `out` array will be set to the ufunc result.
    Elsewhere, the `out` array will retain its original value.
    Note that if an uninitialized `out` array is created via the default
    ``out=None``, locations within it where the condition is False will
    remain uninitialized.
**kwargs
    For other keyword-only arguments, see the :ref:`ufunc docs <ufuncs.kwargs>`.
```

```
Returns
-------
r : ndarray or tuple of ndarray
    `r` will have the shape that the arrays in `x` broadcast to; if `out` is
    provided, it will be returned. If not, `r` will be allocated and
    may contain uninitialized values. If the function has more than one
    output, then the result will be a tuple of arrays.
```

It is possible to use `math.log` to calculate $log_3(7)$ directly. The description of `math.log` describes that `math.log` returns "the logarithm of x to the given base". The signature of `math.log` includes a formal parameter representing the base of a logarithm.

[3]:
```
import math
math.log?
```

```
Docstring:
log(x, [base=math.e])
Return the logarithm of x to the given base.

If the base not specified, returns the natural logarithm (base e) of x.
Type:      builtin_function_or_method
```

$log_3(7)$ is calculated as follows.

[4]:
```
math.log(7, 3)
```

[4]:
```
1.7712437491614221
```

1.1.5 Problem 3

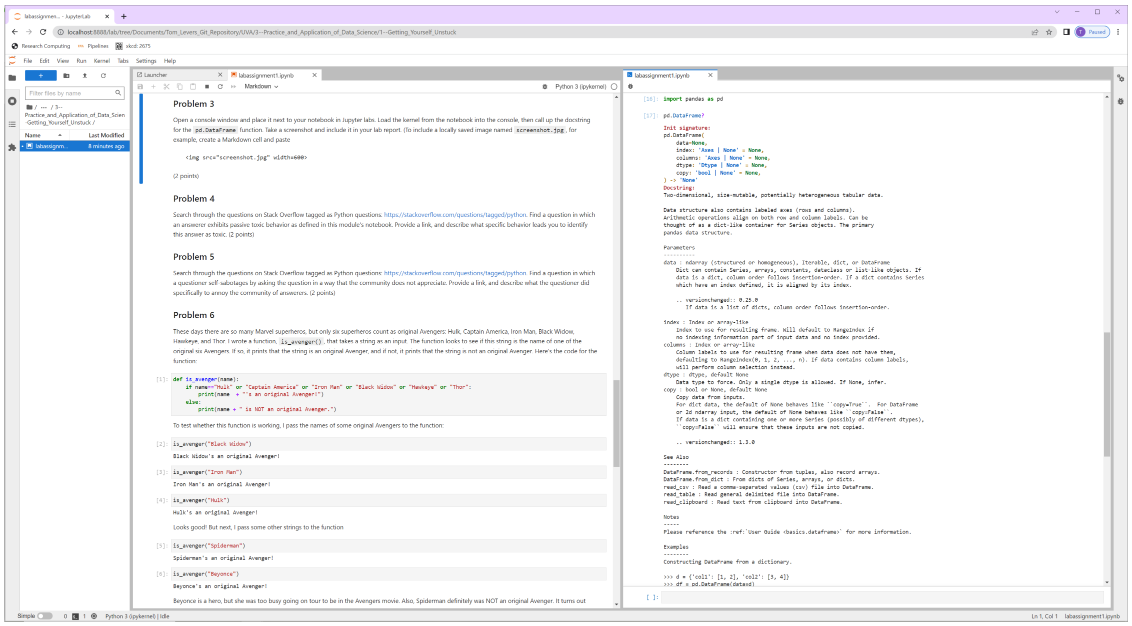
Open a console window and place it next to your notebook in Jupyter labs. Load the kernel from the notebook into the console, then call up the docstring for the `pd.DataFrame` function. Take a screenshot and include it in your lab report. (To include a locally saved image named `screenshot.jpg`, for example, create a Markdown cell and paste

```
<img src="screenshot.jpg" width=600>
```

(2 points)

[2]:

Answer



localhost:8888/lab/tree/Documents/Tom_Levers_Git_Repository/UVA/3--Practice_and_Application_of_Data_Science/1--Getting_Yourself_Unstuck

Problem 3

Open a console window and place it next to your notebook in Jupyter labs. Load the kernel from the notebook into the console, then call up the docstring for the `pd.DataFrame` function. Take a screenshot and include it in your lab report. (To include a locally saved image named `screenshot.jpg`, for example, create a Markdown cell and paste

```
<img src="screenshot.jpg" width=600>
```

(2 points)

Problem 4

Search through the questions on Stack Overflow tagged as Python questions: https://stackoverflow.com/questions/tagged/python. Find a question in which an answerer exhibits passive toxic behavior as defined in this module's notebook. Provide a link, and describe what specific behavior leads you to identify this answer as toxic. (2 points)

Problem 5

Search through the questions on Stack Overflow tagged as Python questions: https://stackoverflow.com/questions/tagged/python. Find a question in which a questioner self-sabotages by asking the question in a way that the community does not appreciate. Provide a link, and describe what the questioner did specifically to annoy the community of answerers. (2 points)

Problem 6

These days there are so many Marvel superheros, but only six superheros count as original Avengers: Hulk, Captain America, Iron Man, Black Widow, Hawkeye, and Thor. I wrote a function, `is_avenger()`, that takes a string as an input. The function looks to see if this string is the name of one of the original six Avengers. If so, it prints that the string is an original Avenger, and if not, it prints that the string is not an original Avenger. Here's the code for the function:

```
[1]: def is_avenger(name):
         if name=="Hulk" or "Captain America" or "Iron Man" or "Black Widow" or "Hawkeye" or "Thor":
             print(name  + "'s an original Avenger!")
         else:
             print(name + " is NOT an original Avenger.")
```

To test whether this function is working, I pass the names of some original Avengers to the function:

```
[2]: is_avenger("Black Widow")
Black Widow's an original Avenger!

[3]: is_avenger("Iron Man")
Iron Man's an original Avenger!

[4]: is_avenger("Hulk")
Hulk's an original Avenger!
```

Looks good! But next, I pass some other strings to the function

```
[5]: is_avenger("Spiderman")
Spiderman's an original Avenger!

[6]: is_avenger("Beyonce")
Beyonce's an original Avenger!
```

Beyonce is a hero, but she was too busy going on tour to be in the Avengers movie. Also, Spiderman definitely was NOT an original Avenger. It turns out

```
[16]: import pandas as pd

[17]: pd.DataFrame?

Init signature:
pd.DataFrame(
    data=None,
    index: 'Axes | None' = None,
    columns: 'Axes | None' = None,
    dtype: 'Dtype | None' = None,
    copy: 'bool | None' = None,
) -> 'None'
Docstring:
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
pandas data structure.

Parameters
----------
data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame
    Dict can contain Series, arrays, constants, dataclass or list-like objects. If
    data is a dict, column order follows insertion-order. If a dict contains Series
    which have an index defined, it is aligned by its index.

    .. versionchanged:: 0.25.0
       If data is a list of dicts, column order follows insertion-order.

index : Index or array-like
    Index to use for resulting frame. Will default to RangeIndex if
    no indexing information part of input data and no index provided.
columns : Index or array-like
    Column labels to use for resulting frame when data does not have them,
    defaulting to RangeIndex(0, 1, 2, ..., n). If data contains column labels,
    will perform column selection instead.
dtype : dtype, default None
    Data type to force. Only a single dtype is allowed. If None, infer.
copy : bool or None, default None
    Copy data from inputs.
    For dict data, the default of None behaves like ``copy=True``.  For DataFrame
    or 2d ndarray input, the default of None behaves like ``copy=False``.
    If data is a dict containing one or more Series (possibly of different dtypes),
    ``copy=False`` will ensure that these inputs are not copied.

    .. versionchanged:: 1.3.0

See Also
--------
DataFrame.from_records : Constructor from tuples, also record arrays.
DataFrame.from_dict : From dicts of Series, arrays, or dicts.
read_csv : Read a comma-separated values (csv) file into DataFrame.
read_table : Read general delimited file into DataFrame.
read_clipboard : Read text from clipboard into DataFrame.

Notes
-----
Please reference the :ref:`User Guide <basics.dataframe>` for more information.

Examples
--------
Constructing DataFrame from a dictionary.

>>> d = {'col1': [1, 2], 'col2': [3, 4]}
>>> df = pd.DataFrame(data=d)
```

1.1.6 Problem 4

Search through the questions on Stack Overflow tagged as Python questions: https://stackoverflow.com/questions/tagged/python. Find a question in which an answerer exhibits passive toxic behavior as defined in this module's notebook. Provide a link, and describe what specific behavior leads you to identify this answer as toxic. (2 points)

Answer The down-vote of the question at https://stackoverflow.com/questions/75167212/extract-empty-data-from-password-protected-website-using-selenium would be justified, more useful, and less toxic if the down-voter were to explain the criteria by which the down-voter deemed the question lacking in effort, clarity, and/or use.

1.1.7 Problem 5

Search through the questions on Stack Overflow tagged as Python questions: https://stackoverflow.com/questions/tagged/python. Find a question in which a questioner self-sabotages by asking the question in a way that the community does not appreciate. Provide a link, and describe what the questioner did specifically to annoy the community of answerers. (2 points)

Answer In writing the question at https://stackoverflow.com/questions/75167112/how-to-create-a-dictionary-from-list-with-single-for-loop, the questioner suggested that an interviewer might be wrong, which is akin to suggesting that Python documentation, high-caliber software developers, and/or volunteers are wrong. It may have been better to ask about creating dictionary-like data structures that allow duplicate keys.

1.1.8 Problem 6

These days there are so many Marvel superheros, but only six superheros count as original Avengers: Hulk, Captain America, Iron Man, Black Widow, Hawkeye, and Thor. I wrote a function, `is_avenger()`, that takes a string as an input. The function looks to see if this string is the name of one of the original six Avengers. If so, it prints that the string is an original Avenger, and if not, it prints that the string is not an original Avenger. Here's the code for the function:

```
[5]: def is_avenger(name):
         if name=="Hulk" or "Captain America" or "Iron Man" or "Black Widow" or␣
      ↪"Hawkeye" or "Thor":
             print(name  + "'s an original Avenger!")
         else:
             print(name + " is NOT an original Avenger.")
```

To test whether this function is working, I pass the names of some original Avengers to the function:

```
[6]: is_avenger("Black Widow")
```

```
Black Widow's an original Avenger!
```

```
[7]: is_avenger("Iron Man")
```

```
Iron Man's an original Avenger!
```

```
[8]: is_avenger("Hulk")
```

```
Hulk's an original Avenger!
```

Looks good! But next, I pass some other strings to the function

```
[9]: is_avenger("Spiderman")
```

```
Spiderman's an original Avenger!
```

```
[10]: is_avenger("Beyonce")
```

```
Beyonce's an original Avenger!
```

Beyonce is a hero, but she was too busy going on tour to be in the Avengers movie. Also, Spiderman definitely was NOT an original Avenger. It turns out that this function will display that any string we write here is an original Avenger, which is incorrect. To fix this function, let's turn to Stack Overflow.

Part a The first step to solving a problem using Stack Overflow is to do a comprehensive search of available resources to try to solve the problem. There is a post on Stack Overflow that very specifically solves our problem. Do a Google search and find this post. In your lab report, write the link to this Stack Overflow page, and the search terms you entered into Google to find this page.

Then apply the solution on this Stack Overflow page to fix the `is_avenger()` function, and test the function to confirm that it works as we expect. (2 points)

Answer I searched for `python string is one of`, and discovered https://stackoverflow.com/questions/17902492/python-testing-whether-a-string-is-one-of-a-certain-set-of-values. This article describes an even more relevant article: https://stackoverflow.com/questions/20002503/why-does-a-x-or-y-or-z-always-evaluate-to-true-how-can-i-compare-a-to-al.

Below, I fix and test `is_avenger` according to the latter article.

```
[11]: def is_avenger(name):
          if name in {"Hulk", "Captain America", "Iron Man", "Black Widow",␣
       ↪"Hawkeye", "Thor"}:
              print(name + "'s an original Avenger!")
          else:
              print(name + " is NOT an original Avenger.")
```

```
[12]: print(is_avenger("Black Widow"))
      print(is_avenger("Iron Man"))
      print(is_avenger("Hulk"))
      print(is_avenger("Spiderman"))
      print(is_avenger("Beyonce"))
```

```
Black Widow's an original Avenger!
None
Iron Man's an original Avenger!
None
Hulk's an original Avenger!
None
Spiderman is NOT an original Avenger.
None
Beyonce is NOT an original Avenger.
None
```

Part b Suppose that no Stack Overflow posts yet existed to help us solve this problem. It would be time to consider writing a post ourselves. In your lab report, write a good title for this post. Do NOT copy the title to the posts you found for part a. (Hint: for details on how to write a good title see the slides or https://stackoverflow.com/help/how-to-ask) (3 points)

Answer I would entitle my StackOverflow post to be specific, succint, and solutions-oriented: "How do I test that a string is one of a set of strings?"

Part c One characteristic of a Stack Overflow post that is likely to get good responses is a minimal working example. A minimal working example is code with the following properties:

1. It can be executed on anyone's local machine without needing a data file or a hard-to-get package or module
2. It always produces the problematic output
3. It using as few lines of code as possible, and is written in the simplest way to write that code

Write a minimal working example for this problem. (3 points)

Answer After writing a statement like, "I attempted to test that a string is one of a set of strings with the following code block. The condition evaluates to `true` regardless of the value of `name`. How do I test that a string is one of a set of strings?", I would provide a working example like the following.

```
[13]: name = "Beyonce"
      if name == "Hulk" or "Captain America" or "Iron Man" or "Black Widow" or␣
       ↪"Hawkeye" or "Thor":
          print(True)
      else:
          print(False)
```

```
True
```

1.1.9 Problem 7

Sign on to the PySlackers slack page and send me a private message in which you tell me which three channels on that Slack workspace look most interesting to you. (2 points)

Answer I'm especially interested in data_science, job_board, and help.

```
[ ]:
```