Lab Assignment 1: How to Get Yourself Unstuck

DS 6001: Practice and Application of Data Science

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.

Problem 0

Import the following libraries:

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

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 1

df.rename(), df.columns(), and df.set_axis() are the three functions that I counted. However, for each function, there are multiple (many many many) numerous ways to utilise the functions to

arrive at the same answer. There were so many, it was hard to keep track of each unique solution.

A problem with relying too heavily on Google and Stack Overflow is that the problem that you're probably trying to solve has so many solutions, that it's going to take a long time to go through each solution and find the solution that works for you.

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 2

In [9]:
```
print(np.log.__doc__)
```

```
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.
       https://personal.math.ubc.ca/~cbm/aands/page_67.htm
.. [2] Wikipedia, "Logarithm". https://en.wikipedia.org/wiki/Logarithm
```

```
Examples
--------
>>> np.log([1, np.e, np.e**2, 0])
array([  0.,   1.,   2., -Inf])
```

In [12]:
```
np.log3(7)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[12], line 1
----> 1 np.log3(7)

File ~\anaconda3\lib\site-packages\numpy\__init__.py:311, in __getattr__(attr)
    308     from .testing import Tester
    309     return Tester
--> 311 raise AttributeError("module {!r} has no attribute "
    312                      "{!r}".format(__name__, attr))

AttributeError: module 'numpy' has no attribute 'log3'
```

I don't think we can use this log function to calculate log base-3 7 as the numpy log function only works with base-10. I did note that in the "see also" section of the doc string, there was log2 function, so I quickly tried log3, which didn't work. (There is a numpy function that can find it though: np.logn(x, base) but that's not THIS function soooo).

In [13]:
```
print(math.log.__doc__)
```

```
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.
```

In [14]:
```
math.log(7,3)
```

Out[14]:
```
1.7712437491614221
```

The math function can be used to calculate log base-3 7 (as shown above). The doc string mentions that we can specify the base (see: "to the given base" and "[base=math.e]".

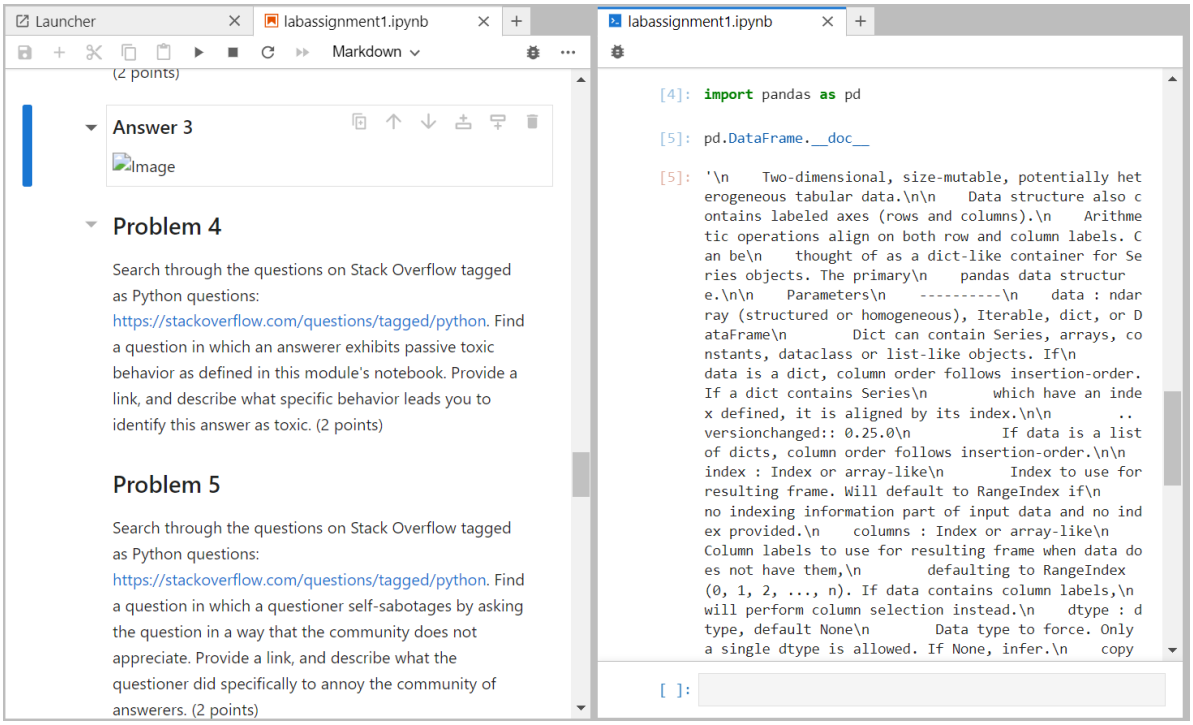
Problem 3

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)

Answer 3



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 4

I'm not sure how "toxic" this comment is, but at this link: https://stackoverflow.com/questions/38987/how-do-i-merge-two-dictionaries-in-a-single-expression-in-python?page=2&tab=scoredesc#tab-top

someone says "your implementation is weak. It is insane to use ... in list(y.keys()) instead of just ... in y".

I understand that they are trying to convey an easier/more efficient way to write code, but to call it "weak"... Why don't they just say "if I were you, I would do it this way since it would be more efficient" instead of insulting their skills. At least, it sounds like they're insulting their skills. Like... damn. Chill.

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 5

At this link: https://stackoverflow.com/questions/76534707/why-cant-i-put-a-while-loop-around-this-code

The poster asks about a while loop when there is no while loop. That means that the answerers have to figure out the intention of what the code needs to do, which (as seen in the comments) makes people confused. "How are we supposed to fix something that's not there?" type of questions. Luckily, it seems like someone was able to understand what the poster wanted, but still.

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:

In [1]:
```
def is_avenger(name):
    if name=="Hulk" or "Captain America" or "Iron Man" or "Black Widow" or "Hawkeye" c
        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:

In [2]:
```
is_avenger("Black Widow")
```

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

In [3]:
```
is_avenger("Iron Man")
```

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

In [4]:
```
is_avenger("Hulk")
```

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

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

In [5]:
```
is_avenger("Spiderman")
```

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

In [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 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 6a

link to solution: https://stackoverflow.com/questions/73518232/why-does-my-if-statement-always-evaluate-to-true My search words were "python if statement always return true using OR stackoverflow", but before that I tried: "python if statement always return true stackoverflow" "python if statement always return true with or condition" "python if statement always true problem" "if statement python not working with multiple conditions" "if else statement python multiple conditions stack overflow" (the solution implemented can be found at the bottom)

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 6b

My title: "Why does my IF ELSE statement always return TRUE for the IF statement when evaluating multiple OR conditions?" ***I wrote this title under the assumption that I would've checked to see if just one OR statement worked :)

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 6c

In [15]:
```
def num(x):
    if x==2 or 5 or 6:
        print("Yes")
    else:
        print("No")

num(1)
num(2)
num(3)
```

```
Yes
Yes
Yes
```

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 7

I sent an email under (jdu5sq@viriginia.edu) listing my three interesting channels on PySlackers.

In [16]:
```
def is_avenger(name):
    if name=="Hulk" or name=="Captain America" or name=="Iron Man" or name=="Black Wid
        print(name  + "'s an original Avenger!")
    else:
        print(name + " is NOT an original Avenger.")
```

In [17]:
```
is_avenger("Black Widow")
```

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

In [18]:
```
is_avenger("Iron Man")
```

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

In [19]:
```
is_avenger("Hulk")
```

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

In [20]:
```
is_avenger("Spiderman")
```

```
Spiderman is NOT an original Avenger.
```

In [21]:
```
is_avenger("Beyonce")
```

```
Beyonce is NOT an original Avenger.
```