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 [22]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
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)

Problem 1 Answer

At least 11 unique solutions were proposed in this post but there were many more responses that more or less proposed some variant on these types. The problem with developing a habit is that it can be overwhelming to pore through all the answers and see the same answers back to back with slight variations. It is important to recognize which solutions are the same, which are fastest, which are easiest, and which are unnecessary. For example, a solution that proposes writing a function to rename columns is unnecessary when the rename method is already built into Pandas.

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)

In [23]:
```
import numpy as np
import math as math
np.log??
```

```
Call signature:  np.log(*args, **kwargs)
Type:            ufunc
String form:     <ufunc 'log'>
File:            ~/anaconda3/lib/python3.11/site-packages/numpy/__init__.py
Docstring:
log(x, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True[, signatu
re, 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])
```

```
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 not possible to use numpy's log package to calculate log_3(7) because numpy's argument does not let you define a base, it only works with a natural log. However, it is possible to use math's log package to calculate log_3(7) because math's log package allows you to define a base.

In [24]:
```
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
```

In [25]:
```
math.log(7,3)
```

Out[25]: 1.7712437491614221

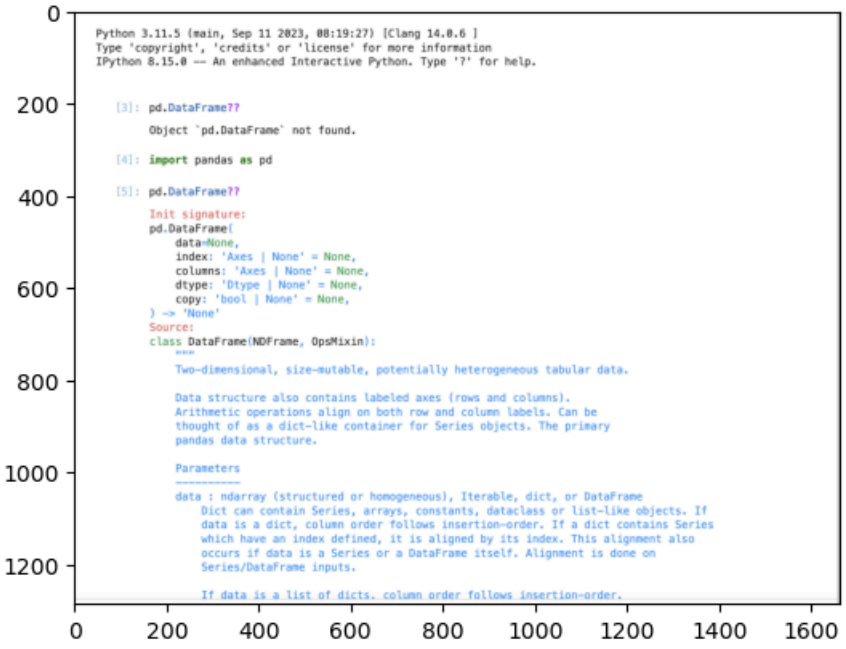
Problem 3

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)

In [31]:
```
img = mpimg.imread('Screenshot 2024-06-23 at 6.00.41 PM.png')
imgplot = plt.imshow(img)
plt.figure(figsize = (240,96))
plt.axis("off")
plt.show()
```



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)

Link: https://stackoverflow.com/questions/74618868/how-to-return-json-from-fastapi-backend-using-websockets-to-vue-frontend The first answer on here demonstrates passive toxic behavior in the sense that the commenter does not explain how to use the recommended package and tells the asker to "just use it" without giving any context or providing a clear answer to the asker's question.

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)

Link: https://stackoverflow.com/questions/20109391/how-to-make-good-reproducible-pandas-examples This user's question is unfortunately quite broad in its scope. Their question is an important one, but rather than providing a concise, specific problem, the user is asking about a class of problems that one can approach any which way. Some users very generously gave long-winded answers as to how they would approach the problem, but overall the question is too vague to be really helpful for most users.

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 [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:

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

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

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

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

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

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

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

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

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

In [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)

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) (2 points)

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. (2 points)

Link: https://stackoverflow.com/questions/20002503/why-does-a-x-or-y-or-z-always-evaluate-to-true-how-can-i-compare-a-to-al. I looked up many variations on this question and couldn't find a satisfactory one through just Google, so I asked chatGPT to help find me a post that asks about True/False statements like this not showing the alternative and ChatGPT found me a post that worked very well.

In [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.")
```

In [12]:
```
is_avenger("Spiderman")
```

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

Why does my python function accept any string input as True for my if/else/print conditions?

In [13]:
```
def func(x):
    if x == 1 or 2:
        print("This is a number.")
    else:
        return x
```

In [14]:
```
func("a")
```

```
This is a number.
```

Problem 7

This problem will test your ability to use a chatbot based on a large-language model, such as ChatGPT, to do data wrangling. Please sign-up for a free account to use ChatGPT if you have not already done so, and log on to the chat interface website for ChatGPT.

Part a

The following data comes from a Kaggle project on Jobs and Salaries in Data Science, compiled by Lucas Galanti (though I don't see an attribution to the original data source, so please take these numbers with a grain of salt). Load the data by running this cell:

In [15]:
```
jobs = pd.read_csv('jobs_in_data.csv')
jobs
```

Out[15]:

	work_year	job_title	job_category	salary_currency	salary	salary_in_usd	employee_residence	experie
0	2023	Data DevOps Engineer	Data Engineering	EUR	88000	95012	Germany	
1	2023	Data Architect	Data Architecture and Modeling	USD	186000	186000	United States	
2	2023	Data Architect	Data Architecture and Modeling	USD	81800	81800	United States	
3	2023	Data Scientist	Data Science and Research	USD	212000	212000	United States	
4	2023	Data Scientist	Data Science and Research	USD	93300	93300	United States	
...	...	...	...	...	...	...	...	
9350	2021	Data Specialist	Data Management and Strategy	USD	165000	165000	United States	
9351	2020	Data Scientist	Data Science and Research	USD	412000	412000	United States	
9352	2021	Principal Data Scientist	Data Science and Research	USD	151000	151000	United States	
9353	2020	Data Scientist	Data Science and Research	USD	105000	105000	United States	
9354	2020	Business Data Analyst	Data Analysis	USD	100000	100000	United States	

9355 rows × 12 columns

Our goal is to manipulate the `jobs` dataframe to create a table with four rows: one for each of the job titles Data Analyst, Data Engineer, Data Scientist, and Machine Learning Engineer; and two columns: one for the year 2022 and one for 2023. Inside each cell should be the average salary (`salary_in_usd`) for that job title and year, rounded to the nearest dollar. The resulting table should look something like this:

	2022	2023
Data Analyst	108658	110988
Data Engineer	139803	149945
Data Scientist	138529	163714
Machine Learning Engineer	151775	191026

Your task is to use ChatGPT -- and ONLY chatGPT -- to generate Python code that uses `pandas` that can generate a dataframe that looks like the above table. For this problem, use markdown cells in your notebook to display both your prompts and ChatGPT's responses. You will almost certainly need to issue several follow-up prompts to get to an answer, and you should list all of your prompts and the responses in your answer.

A few points to keep in mind:

- You will receive better responses by following the guidelines listed here: https://jkropko.github.io/surfing-the-data-pipeline/ch1.html#method-5-using-a-large-language-model-a-chatbot-to-generate-and-debug-code
- You are more likely to get an answer that works by chunking the task into discrete steps. Some of the steps that are needed here are:
  - Keep only the rows from 2022 and 2023
  - Keep only the rows with job titles Data Analyst, Data Engineer, Data Scientist, and Machine Learning Engineer
  - Collapse the data by taking the average `salary_in_usd` within each remaining year and job title combination
  - Reshape the data by moving the years to the columns
  - Round the average salaries to the nearest dollar

Whether you state these specific steps in your prompts or not, and the order in which you state them if you do, is up to you.

One last note: remember you are trying to generate code to generate the average salary table, not the table itself. If the code that ChatGPT generates yields using a small sample of the data calculates incorrect averages, that's OK as long as the code works properly for the full dataframe. (3 points)

I have a dataset named jobs that I want to manipulate with pandas. I have a column called work_year and I want to keep every row in that column with the year 2022 or 2023 in it

In [16]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('jobs_in_data.csv')

# Filter the rows where the work_year is either 2022 or 2023
filtered_df = df[df['work_year'].isin([2022, 2023])]

# Display the filtered dataset
filtered_df
```

Out[16]:

	work_year	job_title	job_category	salary_currency	salary	salary_in_usd	employee_residence	experie
0	2023	Data DevOps Engineer	Data Engineering	EUR	88000	95012	Germany	
1	2023	Data Architect	Data Architecture and Modeling	USD	186000	186000	United States	
2	2023	Data Architect	Data Architecture and Modeling	USD	81800	81800	United States	
3	2023	Data Scientist	Data Science and Research	USD	212000	212000	United States	
4	2023	Data Scientist	Data Science and Research	USD	93300	93300	United States	
...	...	...	...	...	...	...	...	
9124	2022	Financial Data Analyst	Data Analysis	USD	100000	100000	United States	E
9127	2022	Data Engineer	Data Engineering	USD	120000	120000	United States	E
9128	2022	Lead Machine Learning Engineer	Machine Learning and AI	EUR	80000	84053	Germany	
9130	2022	Computer Vision Engineer	Machine Learning and AI	USD	125000	125000	United States	E
9133	2022	ML Engineer	Machine Learning and AI	EUR	20000	21013	Portugal	E

9087 rows × 12 columns

Keep only the rows with job titles Data Analyst, Data Engineer, Data Scientist, and Machine Learning Engineer

In [17]:

```
job_titles = ["Data Analyst", "Data Engineer", "Data Scientist", "Machine Learning Engineer"]
filtered_df = filtered_df[filtered_df['job_title'].isin(job_titles)]
filtered_df
```

Out[17]:

	work_year	job_title	job_category	salary_currency	salary	salary_in_usd	employee_residence	experien
3	2023	Data Scientist	Data Science and Research	USD	212000	212000	United States	
4	2023	Data Scientist	Data Science and Research	USD	93300	93300	United States	
5	2023	Data Scientist	Data Science and Research	USD	130000	130000	United States	
6	2023	Data Scientist	Data Science and Research	USD	100000	100000	United States	
9	2023	Data Engineer	Data Engineering	USD	210000	210000	United States	E
...	...	...	...	...	...	...	...	
9111	2022	Data Engineer	Data Engineering	EUR	62000	65141	France	I
9112	2022	Data Scientist	Data Science and Research	USD	48000	48000	Russia	I
9117	2022	Data Scientist	Data Science and Research	USD	100000	100000	Canada	I
9119	2022	Data Scientist	Data Science and Research	USD	100000	100000	Algeria	Er
9127	2022	Data Engineer	Data Engineering	USD	120000	120000	United States	Er

6419 rows × 12 columns

Collapse the data by taking the average salary_in_usd within each remaining year and job title combination

In [18]:
```
collapsed_df = filtered_df.groupby(['work_year', 'job_title'])['salary_in_usd'].mean().reset_index(
collapsed_df
```

Out[18]:

	work_year	job_title	salary_in_usd
0	2022	Data Analyst	108658.286765
1	2022	Data Engineer	139803.400411
2	2022	Data Scientist	138529.095477
3	2022	Machine Learning Engineer	151775.392523
4	2023	Data Analyst	110988.393217
5	2023	Data Engineer	149944.588448
6	2023	Data Scientist	163713.765280
7	2023	Machine Learning Engineer	191026.424769

Reshape the data by moving the years to the columns

In [19]:
```
reshaped_df = collapsed_df.pivot_table(index='job_title', columns='work_year', values='salary_in_us
reshaped_df
```

Out[19]:

work_year	2022	2023
job_title		
Data Analyst	108658.286765	110988.393217
Data Engineer	139803.400411	149944.588448
Data Scientist	138529.095477	163713.765280
Machine Learning Engineer	151775.392523	191026.424769

Round the average salaries to the nearest dollar

In [20]:
```
reshaped_df = reshaped_df.round()
reshaped_df
```

Out[20]:

work_year	2022	2023
job_title		
Data Analyst	108658.0	110988.0
Data Engineer	139803.0	149945.0
Data Scientist	138529.0	163714.0
Machine Learning Engineer	151775.0	191026.0

Part b

Having worked through using ChatGPT for data wrangling, take a moment to reflect on when it makes sense and when it doesn't make sense to use ChatGPT for working with data prior to an analysis. Write a short paragraph here summarizing your thoughts. (1 point)

It makes sense to use ChatGPT when you have a specific goal in mind and you know the steps you want to take to get there. It helps to know how you want to order your thoughts and how you want to continue with the process. It doesn't help if you have a broad question that you don't know how you want to approach because ChatGPT can take the question anywhere. You really want to know what languages and packages you want to use and be able to break the question down into one step at a time. If you can do those things, ChatGPT can be a very helpful tool for breaking down the problem, especially when it provides hints as to why your code might not be working the way you want.

In [ ]:
```
```