

SQL & Python for Analytics

USC Annenberg Digital Lounge
Week 4: Python Part 2

→ Get the guide at: digital-lounge-analytics.carrd.co

Get the guide at: digital-lounge-analytics.carrd.co

Today's session

This is going to be interactive, so please be ready to try out the exercises as we go!

We are going to be using a tool called Hex, which is a tool for using SQL and Python for doing data analysis and visualization.

Shout-out to Hex for giving us their Professional Plan for free!

Go to digital-lounge-analytics.carrd.co

Get set up

Go to digital-lounge-analytics.carrd.co and grab the link to Hex.

TODAY'S PROJECT

- Visit this link to sign up for Hex, the free data notebook tool we'll be using during this class: <https://app.hex.tech/link/1Hw6-74CMkrxy3hLjGgcV8a7>

If asked to select a workspace, choose Annenberg Digital Lounge:

USC Annenberg Digital Lounge

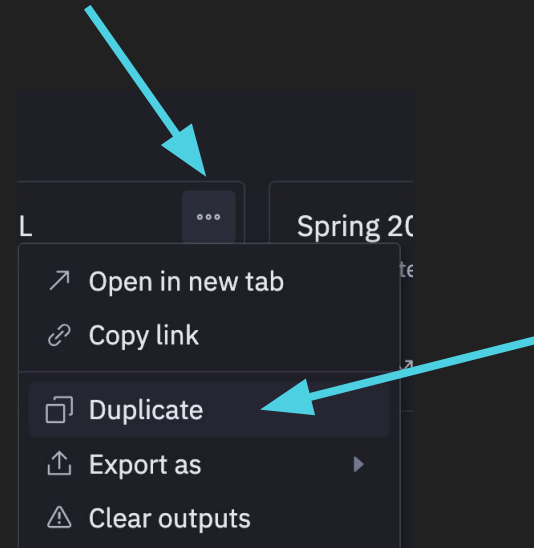
53 members | You are an Editor



Get the guide at: digital-lounge-analytics.carrd.co

Your workbook

In Hex, look for “**Fall 2025 Week 4**”, then click the “...” next to the name, then click **Duplicate**.



Last week

We discussed:

- What python can do for data analysis, and how to identify when to use it
- How to read python code and understand what it's doing
- How to write and edit simple python code for data analysis
- How to figure out what functions you need, when you need them

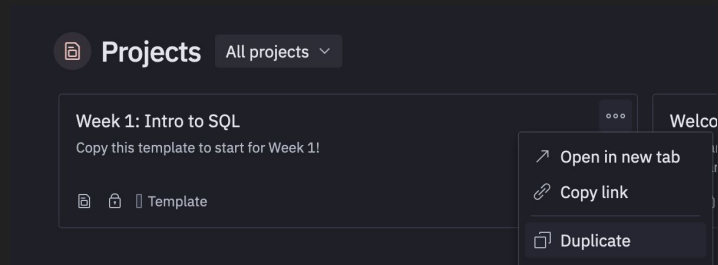
This week

You will learn:

- Refresher on navigating dataframes
- How to write your own functions
- How to use Hex Agent to assist with writing code
- Two ways to apply a function to a dataframe

Activity

1. Go to Projects, next to **Week 4** hit '...' then Duplicate
2. Load your datasets
 - a. Run the first cell to load your data!



```
Code 1

1 import pandas as pd
2 transcripts_data = pd.read_csv('https://query.data.world/s/2svp7yxiggyn4eui6wvklptslqctle?dws=00000')
3 ratings_data = pd.read_csv("SeinfeldRatingsData.xls")
```

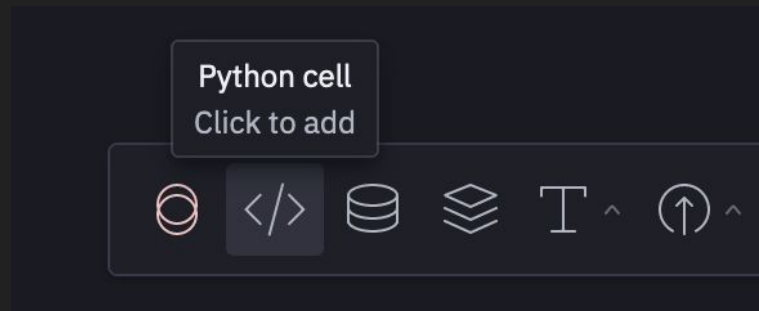
pd transcripts_data ratings_data

Terms to remember

- **Dataframe:** a “structure” you can access with code by rows and columns. There are many types of “structures”; broad term for them all is **Objects**.
- **Function:** a series of steps that you define, to manipulate inputs and produce an output
- **Library:** a bunch of functions somebody else already wrote, that you can just use!

Refresher: first bits of code

- Add a new Python cell to your Hex project



- Check out one of the columns of your dataset:

`ratings_data['Title']`

- To run your code, click Run on that cell, or hit Command-Enter

Add another new Python cell

- Check out one of the rows of your dataset:

`ratings_data.iloc[0]`

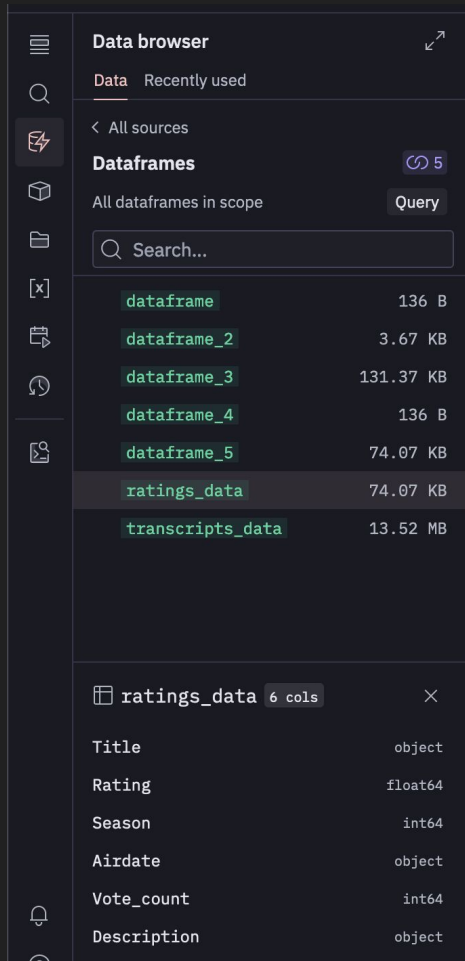
```
1 ratings_data['Title']
2
3 ratings_data.iloc[0]
```

Refresher: Navigating Hex

How do I know what columns are available to select from?

Go to Data Browser → choose a dataframe

See all the columns listed below



The screenshot shows the Hex Data Browser interface. On the left is a sidebar with navigation icons. The main panel is titled 'Data browser' and has tabs for 'Data' and 'Recently used'. Below the tabs, there's a section for 'Dataframes' with a link icon and the number '5'. It says 'All dataframes in scope' and has a 'Query' button. A search bar is present. Below the search bar is a list of dataframes:

DataFrame Name	Size
dataframe	136 B
dataframe_2	3.67 KB
dataframe_3	131.37 KB
dataframe_4	136 B
dataframe_5	74.07 KB
ratings_data	74.07 KB
transcripts_data	13.52 MB

Below the list, the 'ratings_data' dataframe is selected, showing its schema with 6 columns:

Column Name	Column Type
Title	object
Rating	float64
Season	int64
Airdate	object
Vote_count	int64
Description	object

Get the guide at: digital-lounge-analytics.carrd.co

Refresher: Saving info to a dataframe

Similar to creating a variable, you can create a new column in any dataframe to save any results or calculations.

Example:

```
1 ratings_data['a_new_column'] = ratings_data['Season'] + ratings_data['Rating']
```

Then go look in your data browser (left side) under ratings_data!

Variables vs. Functions

Variables allow you to save the results of a calculation to reuse. But what if you want to reuse the calculation itself later?

That's where **functions** come in handy.

We've already used some predefined functions: basic math!

What The Function...?

Here's an example of writing your own function:

```
1  def my_function(input1, input2):  
2      the_thing_i_want_to_output = []  
3      for item in input1:  
4          new_item = item + input2  
5          the_thing_i_want_to_output.append(new_item)  
6      return the_thing_i_want_to_output
```

Using a function

Here's an example of USING that function:

```
1 my_function(['a','b','c'], 'CATS')
```

What is this going to return?

Using a function on a column

What about running that same function on a column?

```
my_function(ratings_data['Title'], 'CATS')
```

This returns the same thing as simply:

```
ratings_data['Title'] + 'CATS'
```

How would I save these results to my dataframe?

Writing code to answer a question

What are the most common words in the philosophy statements of Olympic athletes?

Note that if we have other similar questions, like:

What about the most common words in their hobbies descriptions?

- This will be a good use case for a **function**!

Turning some quick code into a function

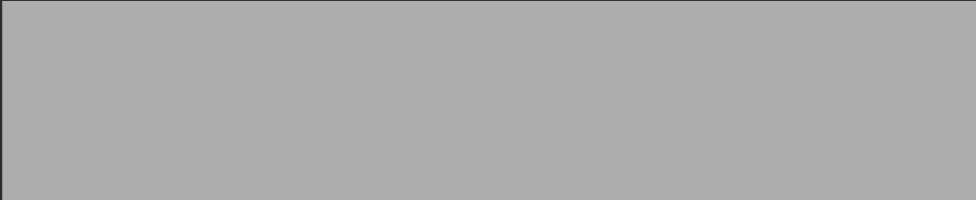
Here is how I'd turn some quick code written for one column, into a function that I could reuse on any other column:

```
1 from collections import Counter
2 Counter(ratings_data['Title']).most_common()
3
4 all_words = []
5
6 for title in ratings_data['Title']:
7     for word in title.split():
8         all_words.append(word)
9
10 print(all_words)
11
12 Counter(all_words).most_common()
13
```

```
def find_most_common_word(column):
    all_words = []
    for thing in column:
        for word in thing.split():
            all_words.append(word)
    return Counter(all_words).most_common()
```

Activity

1. Turn our wordcloud code into a function that takes the dataframe name, the column name, and prints out the word cloud!

```
1 def my_function(input1, input2):  
2       
3  
4  
5  
6     return the_thing_i_want_to_output
```

Activity

1. Test your function on some of the other columns in the athletes_data set:
 - a. Hobbies
 - b. Occupation
 - c. Ritual
 - d. Influences
 - e. etc.!

Using Hex Agent for an AI assist

One of the “new cell” options is Hex Agent, which is an AI assistant for creating SQL, python, or visualizations

Other AI tools offer similar abilities

The way I'd recommend using this is as a first draft, that you'll then revise to get the result you want

You'll still need to understand how SQL and Python work, to be able to use this effectively!

Here are two examples:

Using Hex Agent for an AI assist

The Catch-22

The problem with LLMs is that when it comes to writing code for real data science work (the kind of stuff you'll be doing in your advanced courses and in your career), LLMs are error-prone and require substantial supervision. Moreover, there is good reason to think that this is a problem that our current approach to developing LLMs will never be able to overcome. As a result, most practicing data scientists use LLMs like student research assistants—they let the LLMs write the first draft of code, then review and refine the code based on their own expertise. This is *extremely* useful, but it does mean that to use LLMs effectively, a data scientist still has to learn to program on their own.

And that's where the danger lies with LLMs: while LLMs are error-prone when it comes to real-world data science projects, they're shockingly good at the types of basic programming exercises that are the staple of introductory data science and programming courses. And that can create a temptation to use LLMs extensively in introductory classes, precluding the development of a strong understanding of the principles of programming and setting one up for problems down the road.

https://www.practicaldatascience.org/notebooks/PDS_not_yet_in_coursera/99_advice/llms.html

Get the guide at: digital-lounge-analytics.carrd.co

Hex Agent Example: Python

I gave this prompt:



write a python function to find the most common word in a dataframe column

Use @ for tables and dataframes. [Learn how Magic works](#)

And Magic wrote this function:

Common Word Finder

```
1 def find_most_common_word(df, column):
2     words = df[column].str.split().explode()
3     word_counts = Counter(words)
4     most_common_word = word_counts.most_common(1)[0][0]
5     return most_common_word
```



find_most_common_word

Kinda answers my question, but needs some tweaks

Hex Agent Example: Testing it out

Let's try out the function that AI wrote for us:

Code 24

```
1 find_most_common_word(ratings_data, 'Description')
```

```
'a'
```

Comparing to the results from the script I wrote, looks the same!

Literally gives us only the first word, though.

It also gave it the same name as my function, which overwrote it, so I'll rename it

Hex Agent Example: Next steps

What I would do in this case is:

- **Google** each of the functions or methods that are used, so that I understand what they do
- **Run it line by line**: copy just the first line or function into a new python cell, run it, and see what the results are. Then add the next line or function, and run it to see what the results are

```
1 ratings_data['Description'].str
```

```
1 ratings_data['Description'].str.split()
```

```
1 ratings_data['Description'].str.split().explode()
```

Get the guide at: digital-lounge-analytics.carrd.co

Hex Agent Example #2

Here's another prompt I gave it:

And here's what it returned:

Code 26

```
1 description_words = find_most_common_word(ratings_data['Description'])
```

↳ description_words



use python to create a word cloud of @description_words |

Use @ for tables and dataframes. [Learn how Magic works](#)

Word Cloud Generation

```
1 from wordcloud import WordCloud
2 import matplotlib.pyplot as plt
3
4 # Generate a word cloud
5 wordcloud = WordCloud(width=800, height=800, background_color="white").generate(
6 | " ".join(description_words)
7 )
8
9 # Plot the WordCloud image
10 plt.figure(figsize=(8, 8), facecolor=None)
11 plt.imshow(wordcloud)
12 plt.axis("off")
13 plt.tight_layout(pad=0)
14
15 plt.show()
```



```
-----
TypeError                                Traceback (most recent call last)
/tmp/ipykernel_12/2765158541.py in <cell line: 13>()
    12 # Generate a word cloud
    13 wordcloud = WordCloud(width=800, height=800, background_color="white").generate(
--> 14     " ".join(description_words)
    15 )
    16
```

TypeError: sequence item 0: expected str instance, tuple found

Get the guide at: digital-jungle-analytics.com/co

Hex Agent Example #2: Debugging

What's the error message saying? "Expected str instance"

What is description_words?

Word Cloud Generation

```
1 from wordcloud import WordCloud
2 import matplotlib.pyplot as plt
3
4 # Generate a word cloud
5 wordcloud = WordCloud(width=800, height=800, background_color="white").generate(
6 |     " ".join(description_words)
7 )
8
9 # Plot the WordCloud image
10 plt.figure(figsize=(8, 8), facecolor=None)
11 plt.imshow(wordcloud)
12 plt.axis("off")
13 plt.tight_layout(pad=0)
14
15 plt.show()
```

```
⚠ -----
TypeError                                Traceback (most recent call last)
/tmp/ipykernel_12/2765158541.py in <cell line: 13>()
    12 # Generate a word cloud
    13 wordcloud = WordCloud(width=800, height=800, background_color="white").generate(
--> 14     " ".join(description_words)
    15 )
    16

TypeError: sequence item 0: expected str instance, tuple found
```

Get the guide at: digital-jungle-analytics.com/co

Hex Agent Example #2: Debugging continued

OK, so how do we get a list of just strings?

Hmm, that sounds familiar, we did that earlier!

```
4 all_words = []
5
6 for title in ratings_data['Title']:
7     for word in title.split():
8         all_words.append(word)
9
10 print(all_words)
```

So let's swap description_words with all_words, and see what we get!

Hex Agent Example #2: Success!

Based on the debugging we did, we changed “description_words” to “all_words” and re-ran

Key points:

- Read the code and logic through what it does
- Read the error message
- Edit one individual piece at a time, and run it at every step so you can see the results

```
1 from wordcloud import WordCloud
2 import matplotlib.pyplot as plt
3
4 # Generate a word cloud
5 wordcloud = WordCloud(width=800, height=800, background_color="white").generate(
6     " ".join(all_words)
7 )
8
9 # Plot the WordCloud image
10 plt.figure(figsize=(8, 8), facecolor=None)
11 plt.imshow(wordcloud)
12 plt.axis("off")
13 plt.tight_layout(pad=0)
14
15 plt.show()
```




Get the guide at: digital-lounge-analytics.carrrd.co

Activity

1. Filter the athletes dataframe by a specific country, then run your function on your filtered dataframe

```
1 filtered_df = athletes_data[athletes_data['country'] == 'Ireland']
```

```
1 def my_function(input1, input2):  
2       
3  
4  
5  
6     return the_thing_i_want_to_output
```

Activity

1. Create a function that will filter your dataframe by your desired column value, and then create the same wordcloud as before!

```
def plot_a_column_filtered(dataframe, column_name, column_to_filter, filter_value):
```

If-Else Logic in Functions

We can create a function on just an individual value, and then later we'll apply it to the dataframe. Let's add an emoji based on episode ratings:

```
def rating_symbol(rating):  
    if rating > 9:  
        return '★'  
    if rating > 7:  
        return '🟢'  
    return '🔴'
```

Applying our function to a dataframe

Create a new column in our dataframe, then use **.apply** to run the function on each row of another column!

```
1 ratings_data['rating_emoji'] = ratings_data['Rating'].apply(rating_symbol)
```


Activity

1. Create a function that translates each medal description (e.g. 'Gold Metal') into your choice of emoji!
2. Apply that function to the medallists dataframe, saving the results to a new column 'medal_emoji'

When to use your toolkit

You might be wondering, when should you use SQL or Python?

- In general: start with the simplest tool that does the job
- This might be Excel or Google Sheets!
- When you start working with large datasets, or manipulating a lot of data, or repeating your tasks, try it in SQL
- When you're trying to do something more complicated or customized than SQL can handle, a good time to try it in python
- And with Hex, very easy to switch between SQL and python in each step!
We'll talk about that more next week

Keep learning!

Upcoming sessions:

- Analytics & Visualization with SQL and Python

More resources:

- learn.hex.tech
- Lots of python tutorials online – with Hex, you can skip any setup that requires you to install anything to your computer or use Jupyter, and just type the commands they're showing you into a Hex cell directly

Please share your feedback!

- Digital Lounge feedback: bit.ly/fall25feedback
- Email me: whaleyr@usc.edu