Causal 8607: Assignment 1

Sam Clark

2026-01-15

Format and due date

Prepare this assignment by creating a Quarto markdown document and rendering it to PDF in the Positron integrated development environment (IDE). Upload your PDF document to the class CarmenCanvas website 'Assignment 1' by **15:00 on Thursday January 22**.

It is essential that you complete this assignmetn *before* class on January 22.

Data science tools

File structure, paths, etc.

This is super basic information that you will need throughout this course, so in case you haven't dug into this before, please learn the basics of your computer's terminal and file structure, including how the *path* to a directory or file is specified and manipulated. Here are a couple helpful videos:

- Macos: macOS Terminal Basics: Files, Folders and the File System
- Windows: Navigating With Windows Terminal // Windows Fundamentals // EP 4

Positron

Positron is a relatively new IDE from Posit, the organization that created RSstudio. Positron is a fork of the open source version of the verry popular and widely used Visual Studio Code IDE (VS Code or VSC) that many developers use for general coding. Positron is VS Code adapted to support coding work using both R and Python. Learning how to use Positron well will set you up to work efficiently in both R and Python, and additionally, to easily transition to VS Code if you work in an environment that requires that in the future. We will focus on Python in this course, but feel free to explore Positron's support for R too.

There are a variety of good YouTube videos that introduce Positron and demonstrate how to get going with Python. Below are two required videos produced by Posit and a coouple optional ones if you want to go further. Next week in class, we will assume you have watched the two required videos; they are also super helpful to doing the rest of the homework.

- **Required**
  - Getting Started with Positron: A Quick Tour
  - Your First Python Project in Positron
- Optional
  - AI-Powered Data Science in Positron
  - Take Positron to Work with Positron Pro

Quarto

Google's AI summary of Quarto is a great description of what it is: "**Quarto** is an open-source technical publishing system that unifies code, narrative, and results from languages like Python, R, and Julia to create high-quality documents, presentations, websites, and books in formats such as HTML, PDF, and Word. Built on Pandoc, it's the next generation of R Markdown, offering a streamlined framework for reproducible data science by allowing you to write in Markdown, embed and execute code, and automatically generate formatted outputs."

A Quarto document is written as plain text using markdown and can incorporate code that is executed when the document is rendered one or all of a variety of formats, e.g. HTML, PDF, docx, etc. This allows you to combine narrative text, active code, the results of running the code, and a wide variety of formatting using a single, unified framework. It's a very powerful and flexible tool! Moreover, it was created by the same organization that created Positron so the two work nicely together.

Learnig Quarto requires two things: 1) learn basic markdown, and 2) learn how to create and render Quarto documents in Positron. The Quarto Guide website is a full guide to Quarto, and it contains a nice page describing markdown basics. Use this as a resource when you begin createing Quarto documents. To get started, watch these Youtube videos: Create a Quarto Document in Positron (Python Example), and Your First Python Project in Positron.

Use there resources to create the Quarto document that renders to the HTML that you turn in for this assignment.

Git and GitHub

We may get into this later. If you're curious now, here's a good instroduction video focusing on GitHub integration with VS Code: How To Use Git In VS Code Like A Pro!.

Python basics

There are thousands of 'Learn Python' courses, books, videos, etc. I searched around for quite a while to try to find Python learning resources that are comprehensive enough to be useful and efficient in terms of time invested. Eventually I found the following series of Youtube videos that get straight to the point and explain things clearly.

You are **required** to watch the series of videos below that cover the basics of Python. You will turn in a transcript of your work following along with the videos doing all the examples that they contain. Create a single Quarto document with sections for each video. Within each video's section create code blocks, and if you want, text to demonstrate and explain what you learned from the video. Render the final document to create an HTML document with your text and the results of running the code blocks. I want to see that you followed along, executed all the code, and understood what you were learning. You are free to organize your 'proof of learning' document as you wish. Remember that this video might be helpful as you get going on your Quarto document: Your First Python Project in Positron.

Required Python learning videos

- Python Tutorial for Beginners 2: Strings - Working with Textual Data. Time: 21:12.
- Python Tutorial for Beginners 3: Integers and Floats - Working with Numeric Data. Code snippets. Time: 11:54.
- Python Tutorial for Beginners 4: Lists, Tuples, and Sets. Code snippets. Time: 29:05.
- Python Tutorial for Beginners 5: Dictionaries - Working with Key-Value Pairs. Time: 9:59.
- Python Tutorial for Beginners 6: Conditionals and Booleans - If, Else, and Elif Statements. Code snippets. Time: 16:28.
- Python Tutorial for Beginners 7: Loops and Iterations - For/While Loops. Time: 10:14.

Here's an example of a some explanation and code (in code blocks) that I made while I watched the first video:

First piece of code - print “Hellow World”

```
print('Hello World')
```

```
Hello World
```

Store a string in a variable and index it (or slice it) to extract substrings.

```
my_message = "Sam's World"
print(my_message)
print(len(my_message))
print(my_message[0])
print(my_message[0:5])
print(my_message[6:])
print(my_message[:5])
# print(my_message[11]) # error, indexes 0-10, no 11th slot!
```

```
Sam's World
11
S
Sam's
World
Sam's
```

Figure 1: Markdown code blocks with explanatory text

And below is what it looks like rendered in PDF:

First piece of code - print “Hellow World”

```
print('Hello World')
```

```
Hello World
```

Store a string in a variable and index it (or slice it) to extract substrings.

```
my_message = "Sam's World"
print(my_message)
print(len(my_message))
print(my_message[0])
print(my_message[0:5])
print(my_message[6:])
print(my_message[:5])
# print(my_message[11]) # error, indexes 0-10, no 11th slot!
```

```
Sam's World
11
S
Sam's
World
Sam's
```