Practice: Python Scripting Infrastructure

Programming in Python - Summer I 2023 - Lyon College

Marcus Birkenkrahe

May 23, 2023

README

(You can find a PDF and a Org-mode version of this file on GitHub, and I will also print practice files for you if I find the time to do so.)

Objectives:

- Find Python on your computer

- Find Python for download to your PC on the web

- Starting and using Google Colab, IDLE, DataCamp and replit.com

- Understand the "literate programming concept"

- Understand when we're using Colab + IDLE + replit.com

- Options for creating and running Python scripts

- Creating and running our first ever Python script

Getting started with Window's CLI

1. Open the **terminal** (aka command line/shell) on your PC: on Windows, open the search bar and enter 'CMD' ("pin to taskbar" for later).

2. Old eyes? Blind? To change the font or the color scheme of this window, right-click on the top of the window and select `Properties`.

3. You can enter commands here at the right of the prompt (looks like `C:\Users\birkenkrahe>`), which is a PC location + a character (`>`).

4. The prompt contains an *absolute path* to your current location: you can get it in Windows with the command `cd`. Enter it. It mirrors what you already know (from the prompt).

5. When you type `cd\` you tell the PC to bring you back to the *root* directory (the top of your file hierarchy), `C:\`. To get back to where you were (your `HOME` directory), enter `cd Users\birkenkrahe` (with your user name, not mine).

6. Let's get on with it! At the **prompt** enter `python --version`. This is a command (`python`) followed by an option.

7. If you get any answer, you've got Python. If you don't, contact your nearest Python retailer for purchasing options [1]. Otherwise, at the prompt, enter `python` again, but this time without an option.

8. You are now on the "Python shell", also called the "console" or the "interactive shell", which sits on top of the Python *executable*.

9. Try to get out! Try the usual suspects: `exit`, `quit`, `q()`, `:q`, and `CTRL-c`. The shell will tell you what's what.

10. Close the terminal by entering `exit` at the command prompt.

Things to remember: terminal/command line interface (CLI)/shell, command prompt, absolute/relative path, root directory, Python shell/console, exiting Python, executable, `whoami`, `cd`, `python`.

Getting started with Python's `IDLE`

1. Open the Windows search bar again and type `IDLE`. You should see at least one app[lication] - pin it to the taskbar and open it.

2. This is the IDLE Shell (for your version of Python, e.g. 3.11.1): the prompt is indicated by `>>>`. Much like the Windows command line, it is meant to be used with the keyboard not the mouse.

3. The two most important commands on the shell are: `help()` to get help, and `CTRL + c` to interrupt any shell process. Enter `help()` first.

---

[1] Seriously: to install anything on college PCs you need admin rights, which you have if you're staff or faculty only. If you do, get the latest version of Python (3.11) from the Microsoft Store - it includes IDLE. If you don't, talk to me. For your personal computer, download the latest version of Python from python.org/downloads.

4. You're now on the `help` shell (which is a shell within a shell within a shell): at the `help>` prompt, type `keywords`. This is a list of all reserved words or commands. Type `symbols` to see a list of all the reserved symbols. This is Python's (basic) dictionary.

5. To try the *keyboard interrupt* command, enter `modules` at the `help>` prompt. On my PC, this leads to an infinite process (`infinity` is printed over and over again): type `CTRL + c` to stop it.

6. Click the `<F1>` function key: this should open the documentation for your version of Python in your default browser. (If it doesn't because `<F1>` is used elsewhere, open the `Help > Python Docs` tab in IDLE. Can you tell me where exactly this document is located?

7. In the documentation, enter `IDLE` in the `Quick Search` field at the top and click `Go`. This should open the IDLE documentation. At the top of the page find the term `modules` and click on it. These are all the already installed libraries with functions (like `help()` - the only function we've seen so far).

8. With your mouse, navigate at the top to the documentation start page. Interesting links here include: `FAQs` (that's what we did before AI), `The Python Standard Library` (all the most important functions that come with Python), and a complete `Tutorial`.

9. There's too much to take in here (now or ever): go back to the interactive shell, and let's look at creating and running a file.

10. To find out more in systematic way, check "Getting started with IDLE" (tutorial), and "Starting with Python IDLE" (video lessons).

Things to remember: IDLE, infinite loop, keyboard interrupt, Python documentation, keywords, symbols, modules, Python Standard Library, FAQ, `help()`.

Getting started with Pythons File Editor

1. In IDLE, to create a new file, type `CTRL + n` (or use the `File` tab if you're addicted to your mouse).

2. In the new window, called `*untitled*`, enter the following two lines - type slowly and notice stuff popping up as you write:

```
# print a greeting
print('Hello, world!')
```

3. What you should have noticed: as you finish typing `print(`, a pop-up says (unhelpfully)

```
(*args, sep=' ', end='\n', file=None, flush=False)
Prints the values to a stream, or to sys.stdout by default.
```

4. You also notice the effects of *syntax highlighting* - known elements of the language are highlighted in different colors: **comments** (`#`) in red, **keywords** (`print`) in purple, **symbols** in black, and **strings** (`'hello'`) in green. Tip: no highlighting means you made a **mistake**.

5. Time to run this little 'hello world' program: press `<F5>` to execute the script. You'll be prompted for a name to save the source code in: when the file explorer opens, you see where Windows wants to put your files - don't let it! Instead, navigate to your `HOME` directory (`C:\Users\birkenkrahe\`), create a `New folder` named `Python` and save the file there as `hello.py`.

6. After saving, the script will execute immediately, and the result should appear in the interactive shell as `hello, world` below the information that the shell had a `RESTART`.

7. What do you think it means that the shell was restarted?[2]

8. Experiment: enter `hello world`, `'hello world'` and `print('hello, world')` in the interactive shell.

9. Try to break the script: remove the closing parenthesis and run the script with `<F5>`. A pop-up will tell you what's wrong. Fix it.

10. Now insert a line with the word `printf` between the comment and the `print` command and run the script. This time, there's no pop-up, but in the interactive shell, a `Traceback` tells you what's wrong.

11. Fix the problem so that you have a pristine "hello, world" program. Save with `CTRL + s`. Then exit IDLE with `CTRL + q`.

Things to remember: comments (`#`), `print`, syntax highlighting, source code, machine code, `HOME` directory, `.py`, "hello world" program, traceback.

---

[2]It means that anything you did in the shell before is now wiped clean. To test that, enter `x=0` in the interactive shell, then enter `x` - the result is 0. Now run your `hello.py` script again. On the shell, enter `x` - this time the result is an error: `x` is not known.

Getting started with `Google Colaboratory`

- Log into your Lyon Google account `[yourname]@lyon.edu`

- Open the Google Colaboratory landing page colab.research.google.com.

- On the landing page, you find a lot of information about Colab - you can look at it later on your own. For now, enter `CTRL + ALT + n`

- A `Scratch cell` opens. This is a cell for code. Drag the cell over the text and/or close the Table of Contencts. Then enter this code:

```
print("Hello, world!")

Hello, world!
```

- Click on the play button on the left side of the cell:
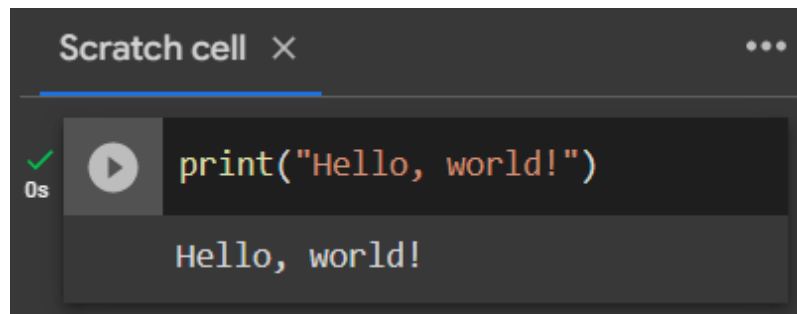
- You can also run a cell with `Shift + Enter` (`Tools> Keyboard shortcuts` is not working for me): delete the output (click on the `X` next to it) and run it with the keyboard shortcut again.

- Open the menu above the cell and select `Copy cell`.

- Open the `File` menu at the top of the pages and select `New notebook`.

- In the new notebook, enter `CTRL + v` and paste the cell into it.

- The notebook will save automatically to your Google Drive.

- In the title at the top, replace `Untitled1` by `hello_world`. The file that is saved in GDrive is now `hello_world.ipynb`.

- Open your GDrive to make sure this has happened. There will be a new directory `Colab Notebooks` where all notebooks are to be found.

- `.ipynb` is the file extension for IPython Jupyter notebooks, an shell environment for interactive literature Python programming.

- Colab/IPython has a lot more features. The most important ones:

  1. Tab-completion: In IPython, you can press the "Tab" key to autocomplete variable names, function names, and file names, making it easier to write code without making typos. (This is automatic in Colab - `Tab` instead inserts a tab).

  2. History: IPython keeps a history of all the commands you've typed, so you can easily access previous commands and reuse them.

  3. Magic commands: IPython provides a number of "magic" commands that allow you to do things like timing code execution, debugging, and profiling.

  4. Inline documentation: IPython provides inline documentation, which means that you can access the documentation for a function or module without leaving the interactive shell.

  5. Integration with other tools: IPython can be used with other tools like Matplotlib for data visualization, NumPy for numerical computing, and Pandas for data analysis.

-

"Literate" programming w/interactive notebooks

- The Google Colaboratory ('Colab') notebook is a browser-based tool for interactive (= real-time) authoring of documents, which combine text, mathematics, code and media output (= literate).

- Interactive notebooks are a standard way of working in data science, and languages like Python and R are particularly suited to it though any language can be used for literate programming.

- The notebook's computing is based on a console or shell program that runs in the background - in Colab, this shell is IPython, in Emacs, the default Python shell is used (and runs in a `*Python*` buffer).

Figure 1: Carl Spitzweg (1839) Der arme Poet (Neue Pinakothek, München)

- More specifically: Python is the programming language, Python 3 is the latest version of Python, IPython is an interactive shell for Python that provides extra features compared to the default Python shell, and Jupyter is a popular notebook (used in Colab).

- A Python example (using a *Python* console):

```
print("hello, world")

>>> hello, world
```

- An R example with graphics (using an *R* console):

```
plot(Nile)
```

- A C example (using the gcc compiler):

```
puts("Hello, world");
```

- A shell example (using the bash(1) shell):

```
echo "hello, world"
```

- A SQLite example (using the sqlite3 console):

```
SELECT "Hello, world";
```

- In the GNU Emacs environment that I'm using, you can see the different console applications as buffers that exist in the background:

- Unlike GNU Emacs' Org-mode, Google Colab (and Jupyter notebooks in general) are limited to either Python + SQL or R + SQL as programming languages.

- The popular VS Code editor has implemented some of Emacs' capabilities with extensions (similar to Emacs packages).

- The concept of "literate programming" was introduced by Donald Knuth in 1984 (GNU Emacs was launched first in 1985), which is when I entered university in Germany!

- If you want to get started with GNU Emacs, talk to me. I'm a fanatic supporter of FOSS in general, GNU and Emacs in particular.

Figure 2: GNU Emacs buffer list with shell/console applications

Exploring `Google Colab` **further**

- Return to your first Colab notebook `hello_world.ipynb`.

- Highlight the empty code cell above the copied + pasted cell and delete it.

- Add a text cell below the code cell with `CTRL + ALT + t` and enter a description of the code:
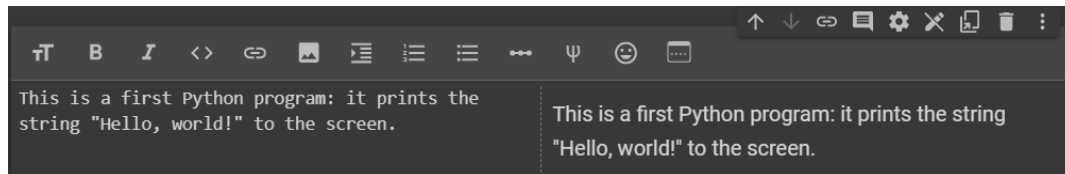

Figure 3: Colab text cell in edit mode

- Then move the text cell above the code cell using the up arrow in the menu right above the text cell:

- You can save copies of your file to GDrive or to GitHub in an existing repository or as a GitHub "gist" (code snippet):

- You can see (and access) all the files that you created in your Colab dashboard at colab.research.google.com:

9
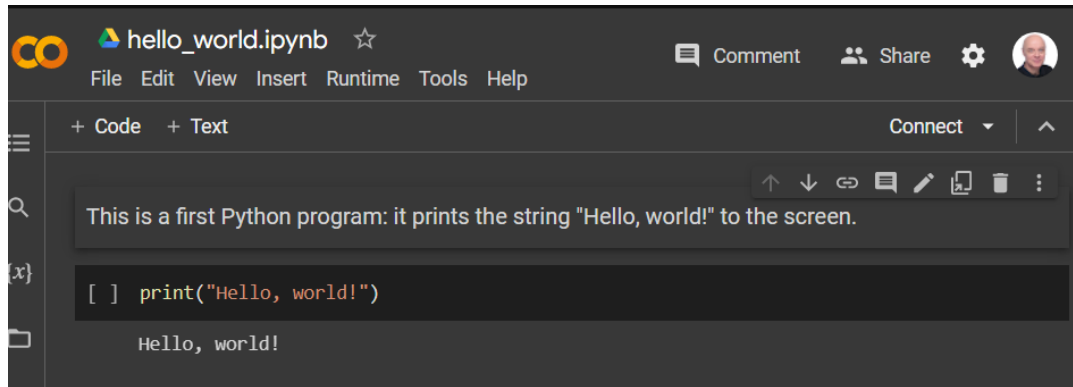
Figure 4: Colab code + text cell

Figure 5: GitHub gist with Colab notebook

Figure 6: Google colab recent file listing

Figure 7: Add new section in the Colab TOC

- Open the first tab in the left side bar (`Table of contents`), and add a new `section`, then enter `First Python Program` after the `#`:

- Move the headline above the text cell. Note that you can hide sections below headlines to make your document more compact:
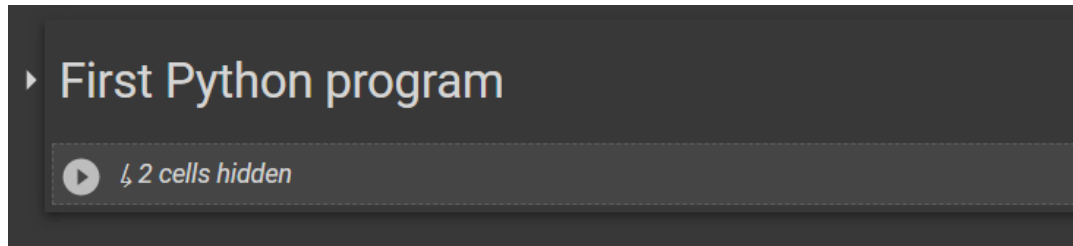

Figure 8: Section headings can hide cells in the section

- Open the search field below the TOC and search for `print`:
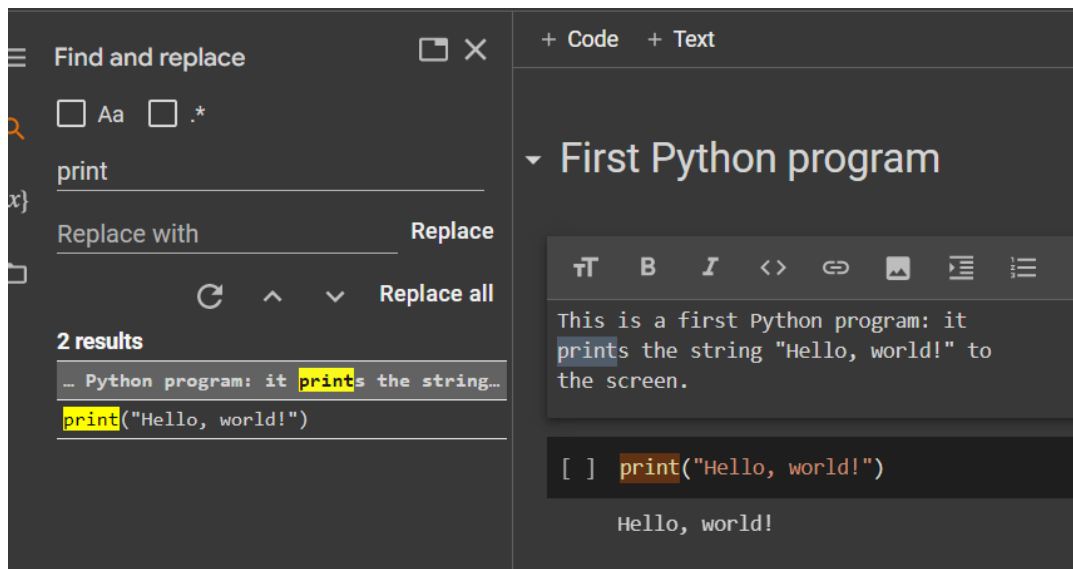
Figure 9: Find + replace menu

- The folder symbol hides the file tree: navigate one level up to see the Linux (container) file tree: to get access to the terminal/shell application, you need a Google Colab Pro license ($9.99/month).

12
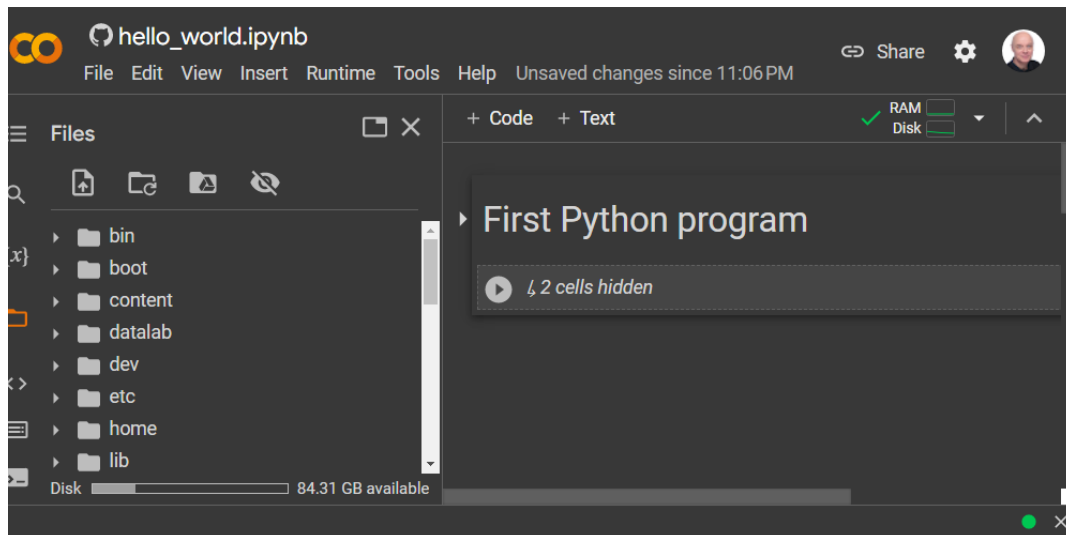
Figure 10: Find + replace menu

- Explore the rest of the functions on your own. You have access to an alternative (equivalent) notebook application via DataCamp including a terminal at workspace.datacamp.com, or at replit.com (more later).

- The workspace at DataCamp is actually a "Jupyter Lab". You can get that on your PC with `pip install jupyterlab` and start it with `jupyter-lab` - the app opens in your browser (locally hosted):
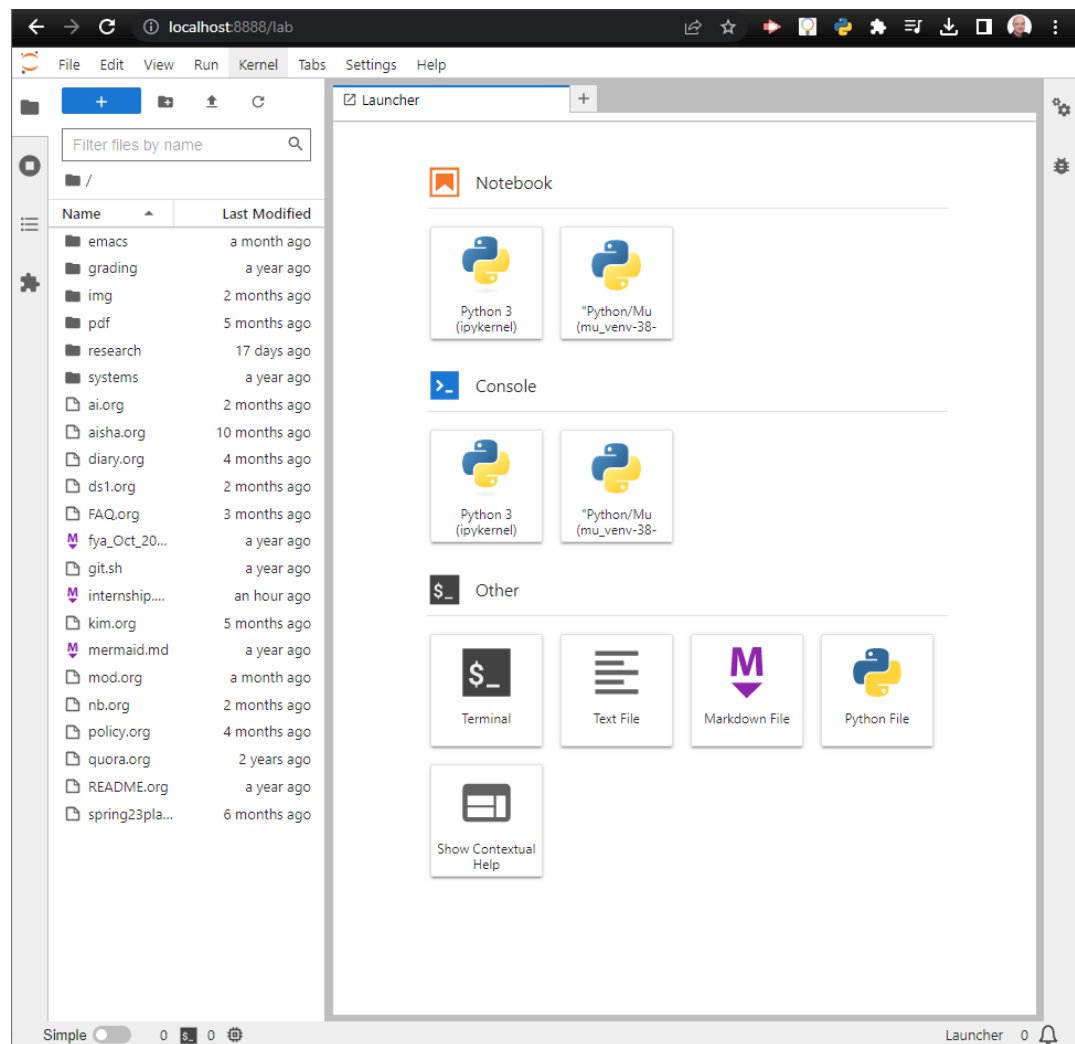
- Here is a short (8 min) video explaining the difference between notebook and lab and how to work with it (Lerner, 2022).

Hello world with `DataCamp workspace`

- With your DataCamp subscription comes free access to a `Jupyter Lab` installation, which includes the best of Colab and replit.com:

  1. a notebook (not available in Replit.com)
  2. a Python console

14

3. a terminal (available in Colab only with upgrade)
4. a file editor
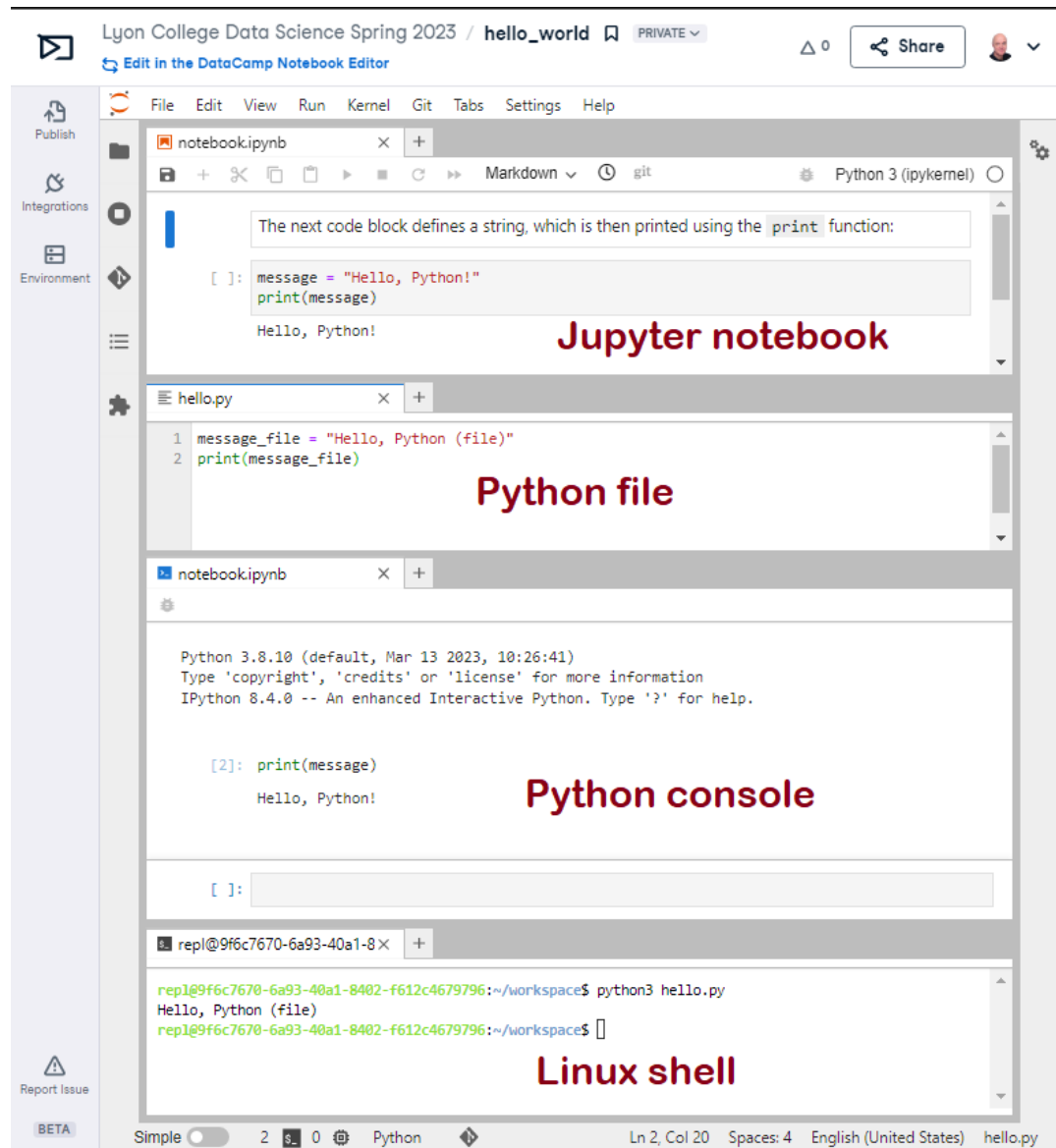
- Go to Canvas now and activate your free DataCamp subscription:
   1. register with DataCamp using your *lyon.edu* account

2. click on the invite link in `Canvas > Pages > Course links`
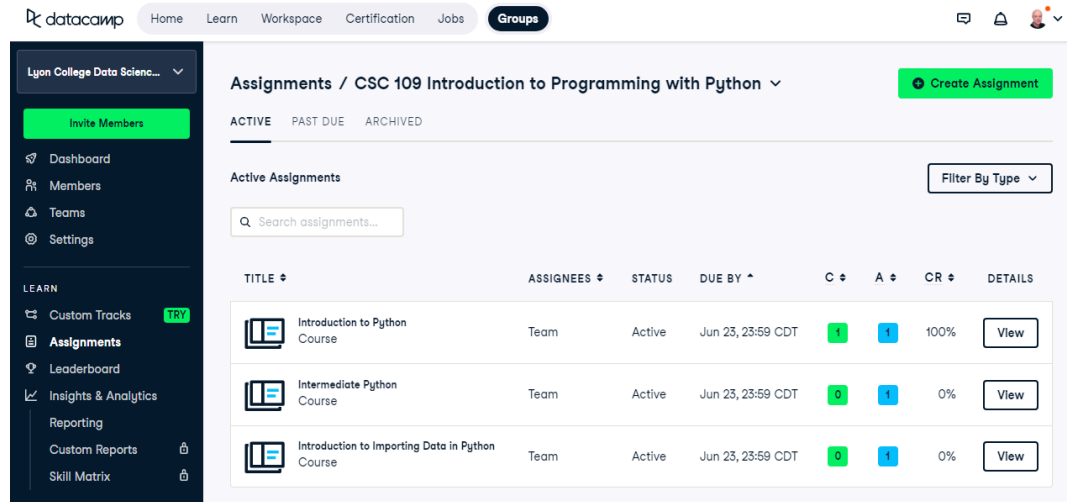3. check that you can see the `Assignments` for this course

Figure 11: DataCamp assignments for CSC 109 Summer I 2023

- Open DataCamp workspaces at: `workspace.datacamp.com`

- Click on `Create workspace` and create a `Python + SQL` workspace titled "Hello_World".

- Under `Files` you can upload your `.ipynb` notebooks.

- Under `Integrations` you get access to many sample databases (e.g. pre-loaded so that you can complete a project online).

- Under `Environment` you can see the multitude of pre-installed Python packages.

- Under `View` you can `Switch to JupyterLab`, which shows you the familiar IPython interface.

- Add a new tab to see the Selection "Python 3", "Terminal", etc.

- Open a `Python file` window. Enter the code to print "Hello, world!".

- Rename the file to `hello_world.py`.

- Add a new tab and pick `Terminal`.

- In the terminal, enter `ll`. This brings up all your files. One should be `hello_world.py`.

- Enter `which python3` to find the location of the Python 3 executable.

- Run the Python script with `python3 hello_world.py`.

- You can close the window. Everything will be saved automatically.

TODO REPL and Python scripts with replit.com

- We want the option of creating files and run them, not just notebooks, either directly or in a terminal.

- To do this online without having to install anything, we can use replit.com. This app also works really well on a smartphone.

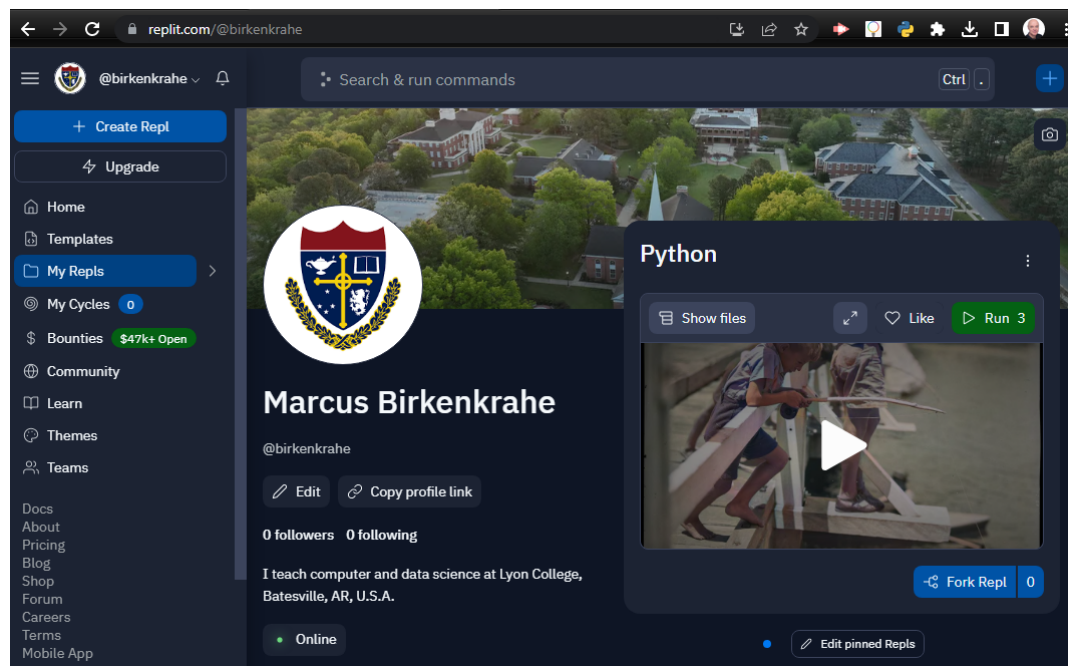- You will have to register at replit.com to write your own scripts:

Figure 12: Customizable replit.com profile page

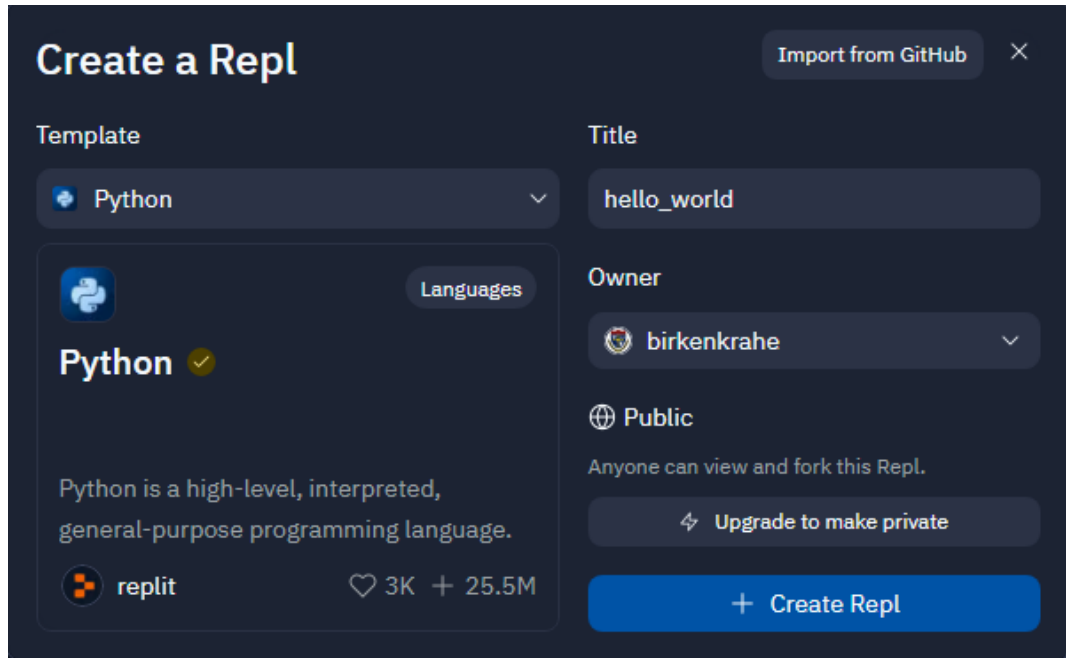- Create a new public REPL and name it `hello_world`:

Figure 13: Create new REPL and name it hello_world

- Once the REPL is booted (= installed for you), you have access to a file editor, a Python console, and a Linux (container) shell:

- Enter the Python code to print "Hello, world!" in the `main.py` partition after the line number 1:

- Disregard the information that appears for now, and run the program `main.py` with the green play button at the top: the string appears in the console.

- Recreate the windows structure shown in the next image, and enter `python --version` in the `Shell` window:

- The "Shell" is an interface to the operating system (Linux). The "Console" is a REPL, an interactive Python shell, and the script is a sequence of Python commands (here only one command) that are executed by the `python` program.
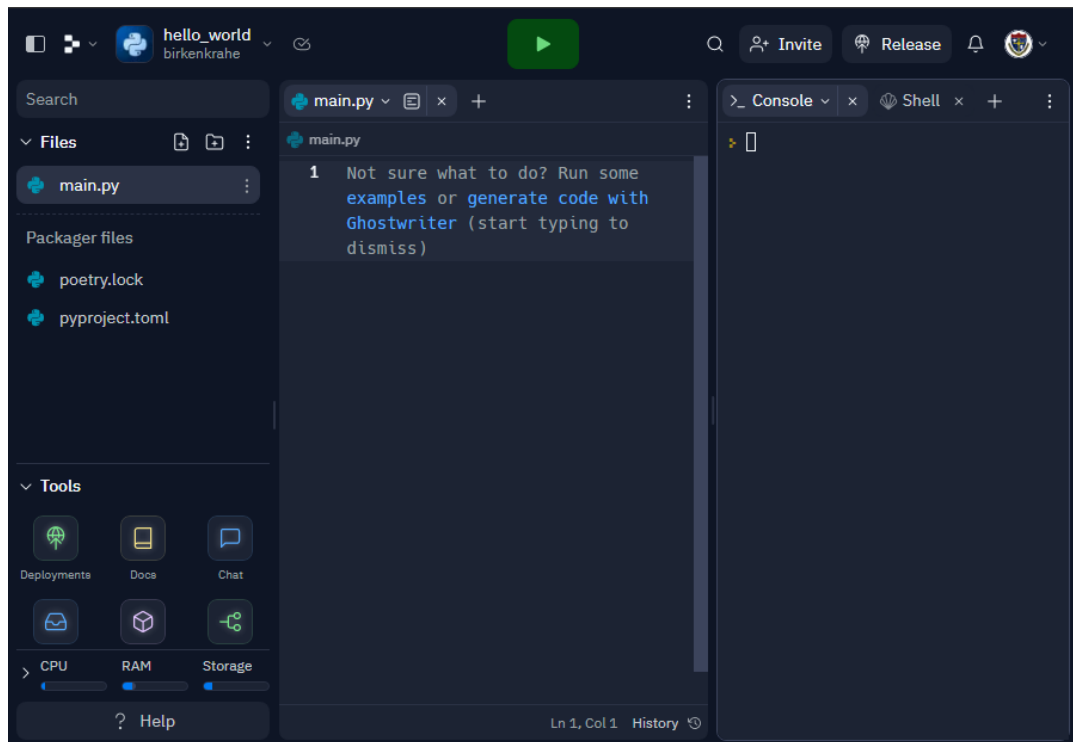
18

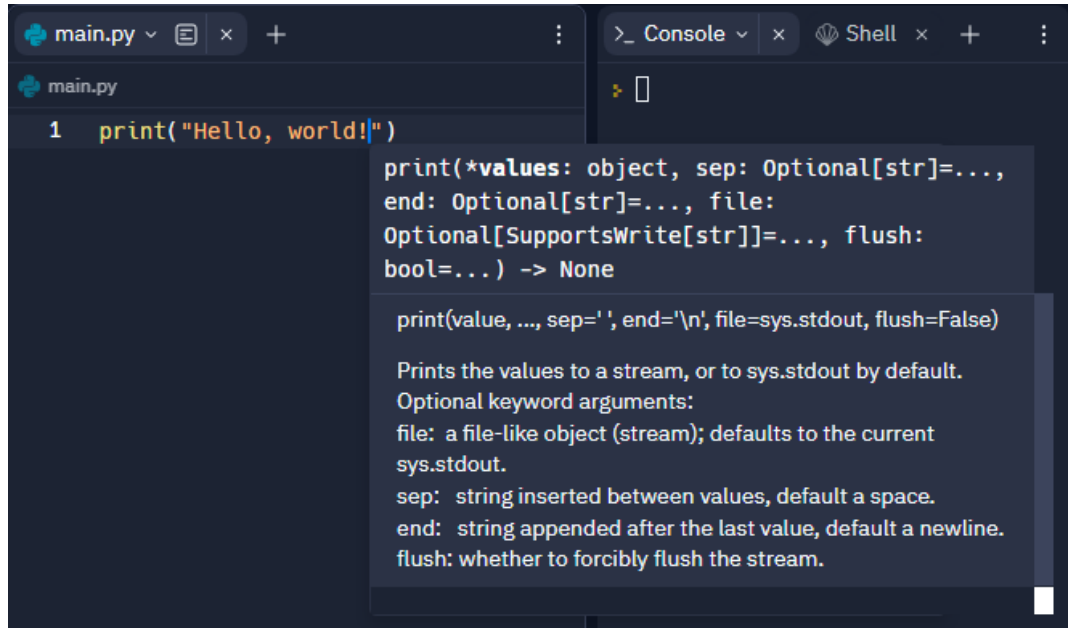Figure 14: REPL tools including file editor, console and shell

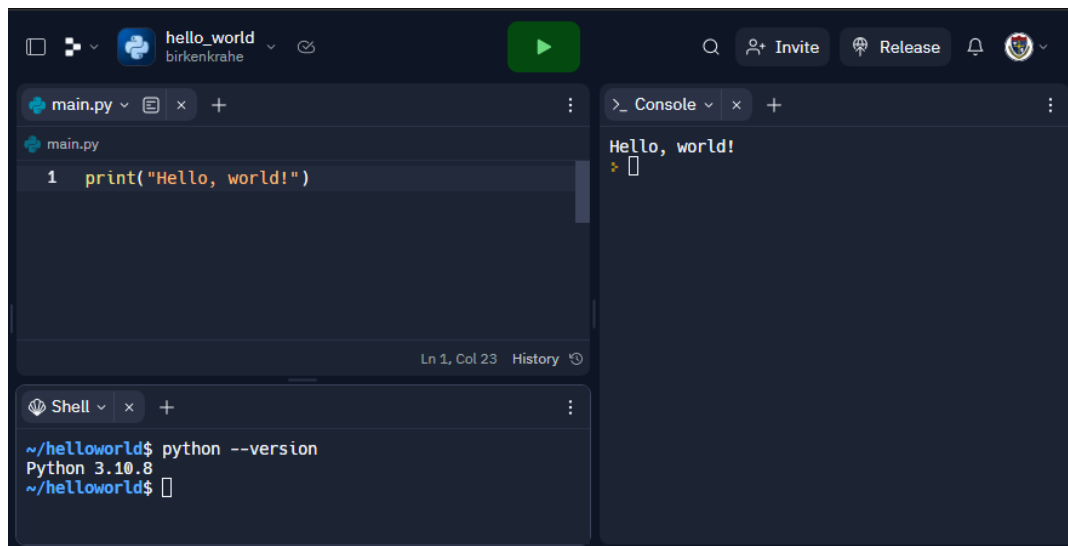Figure 15: Run the "hello world" program in the REPL

Figure 16: Python script, Console (REPL window) and Linux shell

- Run the Python script `main.py` on the Shell (hwere, `~/helloworld$` is the shell prompt consisting of location and prompt sign `$`):

```
~/helloworld$ python main.py
Hello, world!
```

- Like the Colab notebook, your REPL will be saved exactly in the topology in which you left it, including your files (only the command `history` will be deleted when you close the browser window).

TODO Summary