Getting Started with Python, Anaconda, Jupyter, and Pandas

What are we even talking about?

- **Python** is a powerful and user-friendly programming language. Python's functionality is enhanced and extended by many useful *packages*. Python itself and almost all of its packages are free and open-source: anyone can download and modify them free of cost and without restrictions.
- **Anaconda** (specifically Miniconda) is the software you will download to install Python and the packages we will use. Anaconda contains `conda`, a *package manager* that makes it easy to create and manage *environments*, which are collections of Python packages.
- **Jupyter** provides a "notebook" interface for programming in Python. Working in Jupyter *notebooks* enables you to easily organize, document, and share your code. Jupyter can also provide similar functionality for a wide range of other programming languages.
- **Pandas** is an essential package for Python data analysis. It enables us to store and manipulate data in *DataFrames*, efficient and flexible structures that are like Excel sheets on steroids.

OK, how do I get these things?

First, you'll download Miniconda, then use `conda` to create a Python environment and install Jupyter and Pandas.

Visit https://docs.anaconda.com/miniconda/ and download Miniconda for your operating system. If you're on macOS, make sure to download the "pkg" version for your computer's system architecture (newer Macs have Apple M1 processors; older Macs have Intel x86 processors). Open the file you downloaded and follow the on-screen instructions. The default settings should generally work fine.

I have Anaconda installed. Now how do I get ready?

Next, we need to create a suitable Python *environment*. You only need to do this once; afterwards, you can skip to activating and using this environment. We will do so from the *command line*, a powerful yet bare-bones interface where you control your computer directly via typed text commands:

- In Windows, you can access the command line by launching "Anaconda Prompt (miniconda3)" from the Start menu.
- In macOS, you can access the command line by opening the Terminal app (in Applications > Utilities, or press Command-Space and start typing "Terminal").

In the command line, you type commands one at a time into a *prompt* (indicated by a `>` in Windows or a `%` in macOS), then press Enter to send each command. Your computer may respond with updates via text; when it's finished, it will display a new prompt. To learn more about working in a command line environment, check out http://tinyurl.com/commandline-windows or http://tinyurl.com/commandline-mac.

Now that you're in the Anaconda Prompt/Terminal, you will want to create a suitable Python *environment*. (You only need to do this once; afterwards, you can skip to activating and using this environment.) Type:

```
conda create -n data_science pandas jupyter conda-forge::census
```

Let's break down this command to understand it better:

- `conda create` tells conda (the *package manager*) to create a new environment.
- `-n data_science` tells conda to name the new environment `data_science`. You could instead specify whatever other name you like.
- `pandas jupyter conda-forge::census` tells conda to make sure that the `pandas`, `jupyter`, and `census` packages are included in the environment. conda will automatically install any other packages (and there are quite a few) that these packages depend on.
  - `conda-forge::census` means to look for the `census` package on the `conda-forge` *channel*. `conda-forge` is a community-maintained set of packages that includes some less prominent packages that aren't found in the main Anaconda channel (`defaults`).

OK, now that we understand what we're telling conda to do, press Enter. Conda will think for a while, then display a "package plan" listing which packages it will download and install. Type "y" at the prompt to proceed, and conda will finish creating the environment (this may take a few minutes).

Once your new environment is ready, type `conda activate data_science` (or whatever name you chose earlier) and press Enter. This command *activates* the environment, meaning that any conda commands you enter will affect this environment and any code you run will use the packages in this environment. You can tell which environment is active by checking the name in parentheses at the start of the command line prompt.

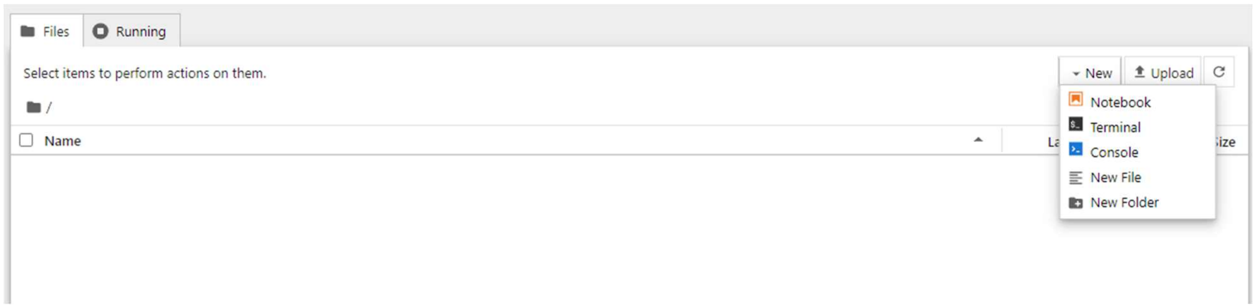
Enough preparations. Time to write code!

There are many ways to write and run Python programs. We will be doing so using Jupyter notebooks. To get started, we need to launch Jupyter from the command line, within the folder containing your data.

To load your project folder in the command line, first navigate to the directory that contains your project folder in File Explorer (Windows) or Finder (macOS). Now open the Anaconda Prompt (Windows) or Terminal (macOS), type `cd ` (including the space), then drag the desired folder from File Explorer/Finder into the terminal window, then hit Enter. Note that the latest command prompt now shows the path to your project folder before the `>` (Windows) or `%` (macOS).

While you're looking at the command prompt, this is a good moment to confirm that the correct Python environment is active: make sure your environment's name is shown in parentheses at the start of the prompt. If not, use `conda activate your_environment_name` to activate it.

Once you have a command line open in your working folder and your environment activated, type `jupyter notebook` and hit Enter. The Jupyter interface will appear in your web browser. In the upper right corner, click New > Notebook.



This will create and open a new Jupyter notebook. If this is the first notebook you've opened, you'll be prompted to "Select Kernel." Just check the box next to "Always start the preferred kernel" and click "Select."

Next to `[ ]:`, type:

```
print('Hello world!')
```

Then click Run > Run Selected Cell.[1] Congratulations: you've just written and run a program! Granted, it doesn't *do* much… Python just obeys your command to print out the string "Hello world!" You can see that sentence printed out below the editor cell you were just typing in. It's not much, but it's a start!

Is everything set up correctly?

To make sure the installation was fully successful, let's *import* the packages we'll use for the upcoming lab exercises.

Importing packages is easy. First, let's create a new notebook *cell*. Click the "+" icon in the Jupyter notebook toolbar to insert a new cell below the currently selected one (which is indicated by a thick bar on the left). In the new cell, type the following line:

[1] See Help > Show Keyboard Shortcuts for many time-saving key commands. Note that the Jupyter Notebook has a *modal* user interface, which means that the keyboard does different things depending on which mode the Notebook is in. There are two modes: edit mode and command mode. Much more useful info at https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html

```
import time
```

Then run the cell. You won't see any output, but you will now have access to the functionality in the `time` package, such as `time.sleep(5)`, which tells the program to wait for five seconds before continuing.

You can give a package a new name – an *alias* – when you import it. This can be a good way to save time by not having to type the full name. For example, in your code, you will frequently be referring to Pandas functionality, so an alias for `pandas` will save a lot of typing. By convention,[2] the `pandas` package is usually imported as `pd`:

```
import pandas as pd
```

You can also import a specific *module* from a given package. This lets you save typing by providing direct access to the functionality you care about. For instance, you can type:

```
from census import Census
```

Or you can import a module *and* give it an alias:

```
import matplotlib.pyplot as plt
```

Oh, whoops, a `ModuleNotFoundError`! I must have forgotten to tell you to install `matplotlib` earlier. If you need to install other packages into your environment, you can always use conda to do so. Open another Anaconda Prompt/Terminal (because the first one you opened is still busy running the Jupyter notebook), activate your environment, and type `conda install package_name`. You can install multiple packages at once by passing `conda install` a list of package names, separated by spaces: `conda install seaborn scikit-learn network`

Try installing `matplotlib` into your environment now, then *restart* the notebook's *kernel* to enable the notebook to use this newly installed package: click Kernel > Restart Kernel… > Restart. Then run all your cells again, because restarting the kernel discards all code and variables that were in memory.[3]

If all the cells in your notebook now run without throwing a `ModuleNotFoundError` or an `ImportError`, your Python installation is complete. (You might see some warnings about missing fonts while importing `matplotlib`, but you can safely ignore these.)

Get a Census Data API key

The last thing you'll need for the upcoming labs is a **Census Data API key**. An *application programming interface* (API) is a way for two programs to communicate with each other. For

[2] Other conventions exist: `import numpy as np`, for instance.
[3] It is good practice to use "Restart Kernel and Clear Outputs of All Cells…" instead of simply "Restart Kernel…", just as a visual reminder that all the cells you ran before restarting the kernel are no longer "in effect."

example, the US Census Bureau provides an API that lets us pull data from the decennial census and American Community Survey, directly into our Python programs. An *API key* is a code that lets you access the API, and lets the API know who is using it.

Go to https://api.census.gov/data/key_signup.html, fill out the extremely short form, and click "Request Key." In a few minutes, you'll receive an email with your new Census Data API key.

To test out your API key, type/paste the following lines into a Jupyter notebook cell:

```
from census import Census
c = Census("your_API_key_goes_here")
c.acs5.get(('NAME', 'B01001_001E'), {'for': 'state:06'})
```

Run the cell. This should return a response from the Census Data API containing the most recent ACS 5-year estimate of the total population of California. If you get a `JSONDecodeError` or an `APIKeyError`, double-check that you've pasted in your API key correctly; if you get a `SyntaxError`, make sure you've wrapped your API key in quotation marks.

All done! You can save your Jupyter notebook by clicking the "save" icon in the toolbar, clicking File > Save Notebook, or pressing Ctrl-S (Windows) or Command-S (macOS). You can also rename it by clicking the current name (default: "Untitled"), to the right of the Jupyter logo at top left.