Getting Ready for your SW Carpentries Session

Felix Henninger, Sonja Grath, Reema Gupta, Malika Ihle and Luisa F. Jimenez-Soto

July 23, 2024

The Carpentries course

Dear Participant,

Welcome! We are happy you are interested in learning more about programming!

In general, Carpentries trainings, like this one, are designed to give you the basic tools and concepts for you to start to explore programming. You can use it for Data Analysis, programming, being lazy (let computers support your work!), and/or improving your work and research.

This course is called "Python flavored", because we will be teaching the basic parts: Unix and its shell, Git, and will focus on the programming language called python.

The structure of the course:

- Day 1: It contains two sections: Unix(Shell) and Git
- Day 2: Python: Programming and visualization

We will be doing mostly live coding sessions in order to give you an insight on how to work with them, but as any language: You need to practice it and if you do not understand, please ask. We will be at least 4 instructors and we are there to help you.

In order to make your experience easier, we have written this document for you to prepare and get the most of the training.

Please read it carefully, and if you have any questions feel free to ask us in the Help desk session scheduled for the day before the training.

Looking forward to meet you all,

your Instructors and helpers.

Getting to know your computer

Disclaimer

Your participation in the course requires that you download and install certain software and libraries maintained by an open community. If you decide to download and install them, we, the instructors, the OSC and the University (LMU) cannot be hold liable for any damages that can occur in your computer. To our best knowledge and limitations, these programs and libraries have proven safe in previous trainings. This is the reason we recommend them and use them. However, each computer system is different. Make sure you take an informed decision before using them.

To the point: Hardware and general software

It is very probable that you have worked with computers for at least a couple of years. Congratulations! However, for this course you will need to get to know your computer even better, specifically your keyboard.

Before classes start, we need you to **find out** the following tasks in your computer and keyboard

- **Your Terminal**: How to open your terminal (Mac OS and Linux OS) or PowerShell (Windows OS), (General information under https://swcarpentry.github.io/shell-novice/)

  If you are a Windows user, please follow the instructions under https://carpentries.github.io/workshop-template/install_instructions/#shell for Windows system,

  MacOS and Linux are both Unix systems, so they have by default a shell. Just check the instructions in the same url mentioned above for your systems to verify you have what you need, and

- how to write the following characters:
  - Caret → ∧
  - Back slash → \
  - Forward slash → /
  - Square brackets → [and]
  - Curly brackets → { and }
  - Round brackets → (and)
  - Backtick → `
  - Tilde → ∼
  - Pipe → |

- Pound or hash → #
- Ampersand → $
- Underscore → _
- Double quotes → " and "
- Single quote → ' and '
- Angle brackets → <and >
- Colon → :
- Semicolon → ;
- Comma → ,

All these characters will be used to communicate with your computer. The last thing you want is not being able to type the commands.

If you want more information about the types of keyboards used, here are some links and sources:

Keyboard layouts (most common)

- American QUERTY: https://en.wikipedia.org/wiki/QWERTY#/media/File:KB_United_States.svg
- QUERTZ German: https://en.wikipedia.org/wiki/QWERTZ#/media/File:German-Keyboard-La png
- AZERT (France and Turkey): https://en.wikipedia.org/wiki/AZERTY#/media/File:KB_France.svg

Articles regarding Keyboards

- Contains the map of the distribution of keyboards in Europe and info about QUERTZ: https://en.wikipedia.org/wiki/QWERTZ
- American QUERTY: https://en.wikipedia.org/wiki/QWERTY
- French AZERTY: https://en.wikipedia.org/wiki/AZERTY

Working with anaconda or miniconda

Conda is a package manager that allows us to use several versions of python in our computer without causing major conflicts.

In the instructions to the course, we gave you the option to install Anaconda or miniconda, two programs to use Conda. Here you have to choose: Anaconda is the easier way to interact with Conda, since it has the Anaconda Navigator, a Graphical User Interface (GUI) beginners friendly. However, *Anaconda is a very big program containing a lot of python libraries that might not be necessary for many people*. For Anaconda, follow these instructions.

Your other option is to use Miniconda, the lighter version with the minimum content to start working with python. However, miniconda does not have a (GUI) like Anaconda does, and it requires installation of the individual packages using the command line interface.If you choose to install miniconda, you will find the information in this section. This includes what you need for installing the miniconda and how to install the libraries we will use for the training: Matplotlib, Pandas, and Jupyter Lab.

For the use of Jupyter Lab using the GUI, you can go to Summary and Setup Lesson from Software Carpentry.

Installing miniconda

Depending on the operating system you have (MacOS, Windows, Linux), you need to download the program. Follow the instructions from the official conda site for the installation in your system.

For those of you who like it more visual, here are the some resources available in YouTube:

- For Windows 10: `https://youtu.be/oHHbsMfyNR4?si=WBbOzWVRgDbUynJh`
- For MacOS: `https://youtu.be/IRINDtw5xro?si=mfTCGE3-if7BCHOw`

Using your installed miniconda to install libraries

Once you have installed your miniconda, you need to verify that the installation is working in your computer.

In your command line terminal / PowerShell type the following

```
$ conda --version
```

You should get something like `conda 23.1.0`. If you do not have the same version, type after the prompt ($ or >) the following:

```
conda install conda
```

And accept the packages to be installed by typing `Y` or enter. You have now the most up-to-date version of conda.

For the course we need to use a couple of packages / libraries for data analysis. We are going to use the same method: interacting with conda using the command line, as we did for the update.

```
# For pandas and numpy
conda install anaconda::pandas
```

```
# for jupyter lab
conda install conda-forge::jupyterlab
```

```
# For matplotlib, for graphics
conda install conda-forge::matplotlib
```

Now you need to make sure you have everything you need for the course. For this you need to get the list of packages inside your conda environment called `base`. Just type after the prompt:

```
# Get the list of all packages inside your conda environment
conda list
```

Inside the list you should find the following (versions might differ a little bit):

- numpy (Version 1.26.4)
- pandas (Version 1.3.5)
- matplotlib (Version 3.9.0)
- jupyterlab (Version 4.2.3)

Recommendation 1: If you need help... we will be there for you!

If for some reason you get stuck or you do not feel confident enough about doing all this alone, do not despair! We have a Help desk (online) planned for **Wednesday, July 31st from 14:00 to 15:00** where we will try to answer any question you have about the installation process. Check your confirmation e-mail for the Zoom link.

Cheatsheet Python

The following pages contain the different commands we will be using in class, on the second day. We want you to be able to keep this information for your future work, and maybe use it in class if you get lost (can happen!). This snippets are based on the course Plotting and Programming with Python created by the Software Carpentry.

Python Fundamentals

Running and Quitting

```
print("hello world")
```

Variables to store values

```
# Create the variable name and assign a value
age = 42
first_name = 'Ahmed'

# Use the print() function to display the values
print(first_name, 'is', age, 'years old')

# Variables can be use for mathematical operations
age = age + 3
print('Age in three years:', age)
```

Access values by indexing

```
# Use the index to access one item in the variable
atom_name = 'helium'
print(atom_name[0])

# Slice to get a substring
atom_name = 'sodium'
print(atom_name[0:3])
```

Data types and type conversion

```
# Declare some variables (assign variables to data)
first_value = 1.0
second_value = "1"
third_value = "1.1"

# Check the type of data stored under the variable name
print(type(first_value))
print(type(second_value))
print(type(third_value))

# Transform the data
print(int(second_value) + float(third_value))
```

Built-in Functions and Help

```
# Obtain the maximum number in a series
max(1, 2, 3)

# Obtain the minimum
min(1, 2, 3)

# evaluate the length of the string "hello"
len("hello")

# Check the information about the function len()
help(len)
```

Lists

```
# Create a list
odds = [1, 3, 5, 7]

# print elements of the list using indexes
print('first element:', odds[0])
print('last element:', odds[-1])

# modify the list
odds.append(9)
print('odds after append:', odds)
odds.extend([11, 13, 15])
print('odds after extend:', odds)
del odds[0]
print('odds after delete:', odds)
```

Loops

```
    # create the list
    squares = [1, 4, 9, 16]

    #loop over each item of the list using index for selection
    for i in range(len(squares)):
        squares[i] = squares[i] + 1
        print(squares)
```

Conditionals

```
    num = 37

    # check if the value stored under num is greater than 100
    if num > 100:
    print("greater")
    else:
    print("not greater")
```

Writing Functions

```
    # declare and define the function to transform fahrenheit to celcius
    def fahr_to_celsius(temp):
    return ((temp - 32) * (5/9))

    # apply the function and print the value
    print(fahr_to_celsius(32))
    print(fahr_to_celsius(212))
```

Libraries

```
    # import the library called math
    import math

    # use the function from math to access the constant pi and print it
    print("pi is", math.pi)

    # Use the function cos() from math library to calculate the cosine of pi, and
        print the answer
    print("cos(pi) is", math.cos(math.pi))
```

Data Analysis using Pandas library

Reading Tabular Data into DataFrames

```
import pandas as pd

# use the fuction read_csv() from the library pandas (pd) to read the csv file
    with the data
data = pd.read_csv('gapminder_gdp_oceania.csv')

#print the dataframe resulting from using the pandas function read_csv()
print(data)
```

Pandas DataFrames

```
# print on the console the columns inside the dataframe under the variable data
print(data.columns)

# get an idea of the information stored in the DataFrame and its organization
print(data.describe())
```

Looping Over Data Sets

```
import pandas as pd
#Create a list containing the names of files you want to open and read in a
    dataframe
filenames = ['data/gapminder_gdp_africa.csv', 'data/gapminder_gdp_americas.csv']

#Create an empty list where all dataframes will be placed
dataframes = []

# Loop over the 'filenames' and read each in a dataframe
for f in filenames:
    dataframes.append(pd.read_csv(f))
    print(f, pd.read_csv(f).describe())
```

Plotting and MatPlotLib library

```
# import the package for ploting (matplotlib.pyplot)
import matplotlib.pyplot as plt

# create a barplot using the values inside the dataframe under "data" variable
data.plot(kind='bar')

# show the plot
plt.show()
```

Advanced extras

Errors and Exceptions

```
    def favorite_ice_cream():
        print('My favorite ice cream is', ice_cream)

    favorite_ice_cream()
```

Defensive Programming

```
    def normalize_rectangle(rect):
        if rect[0] * rect[1] < 0:
            raise ValueError('Invalid rectangle')
        if rect[0] < 0:
            rect[0] = -rect[0]
        if rect[1] < 0:
            rect[1] = -rect[1]
        return rect
```

Variable Scope

```
    pressure = 103.9

    def adjust(t):
        pressure = 0.9 * t
        return pressure

    print(adjust(pressure))

    print(pressure)
```

Programming Style

```
    # Good variable names
    count = 10
    name = "temp_data"

    # Avoid magic numbers
    MAX_TEMP = 100
    if temperature > MAX_TEMP:
    print("too hot")
```

Open Science and FAIR principles

Having a computationally reproducible workflow is part of making science more efficient, transparent, and reliable. Other research practices can enhance the credibility of science, and all of them are often packaged into what is called 'open research practices'. With the Carpentries, the tools your learn here will help you to follow the principles of Open science and FAIR data.

Open Science

How many of you have tried to reproduce an experiment you have read and it does not seem to work? Some of us have. And many times. These "reproduction of experiments" activity has wasted weeks of work and materials just to find out that the results and experiments described together with the protocols published do not bring the same results.

In 2005, a publication by John P.A. Ioannidis [1], discussed openly some of the first problems regarding reproducibility problems. This time it was for Behavioral Research (Psychology and co.), however, the same problems were later made visible in many other areas of science. A more mature view of the problem has lead to defining the reasons, "symptoms" (including P-Hacking) and possible "cure" for the reproducibility crisis [2]. The Open Science movement started.

If you want to read more about what Open Science means, there are excellent resources in the LMU Center for Open Science , teaching materials in the Open Science book, or very nice people willing to help in the OSIM (Open Science Initiative in Medicine)

FAIR principles

FAIR Part of the Open Science ideals is to be able to share the data. The standards for data and metadata (data about how data was obtained) under Open Science principles are defined under the FAIR acronym:

F Findable: The data has an unique identifier and the data complementing it (metadata) is descriptive enough that the identifier can be associated with the studies.

A Accessible: We can access the data with standard communication protocol.

I Interoperable: The data is in an standard format that can be read by all without the need to buy a software for it, and metadata have such standards (called ontologies) that the dataset can be merge with another.

R Reusable: The metadata contains all details about the data and its origin, so that it can be used by other people. The license has to allow this as well.

If you want more detailed information, and how to implement them go the publication from Mark D. Wilkinson et al, "The FAIR Guiding Principles for scientific data management and stewardship" [3]

Sources and Methods used for this guide

The code you find in this guide was extracted from the original material from the SW Carpentries training using ChatGPT 4o, and modified for understanding and simplicity.

The logos are from Open Sources available in Wikipedia under Creative commons or from the organizations themselves.

If you find an error, or a link is not working, please let us know so we can improve the manual. We are happy to get constructive feedback.

Bibliography

[1] J. P. Ioannidis, "Why Most Published Research Findings Are False," *PLOS Medicine*, vol. 2, p. e124, jul 2005.

[2] M. R. Munafò, B. A. Nosek, D. V. Bishop, K. S. Button, C. D. Chambers, N. Percie Du Sert, U. Simonsohn, E. J. Wagenmakers, J. J. Ware, and J. P. Ioannidis, "A manifesto for reproducible science," *Nature Human Behaviour 2017 1:1*, vol. 1, pp. 1–9, jan 2017.

[3] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J. W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S. A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. Van Der Lei, E. Van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons, "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific Data 2016 3:1*, vol. 3, pp. 1–9, mar 2016.