# Chapter 0 - Welcome to Python and Jupyter Notebooks

Objectives:

- Be introduced to Jupyter Notebooks and the Google Colab computing environment
- Run a simple "Hello World" code
- Learn about comments to document code

## 0.0 Introduction

This tutorial is a Jupyter Notebook running in Google Colab. Jupyter notebooks are a great way to mix executable Python code with rich contents (HTML, images, equations written in LaTeX). Colab allows us to run notebooks on the cloud for free without installing Python on your computer.

The purpose of creating an interactive Coding Tutorial is for you to play with the code and try things out. If you "break" the document, you can always go back and re-download the original, so please experiment. The most important thing is to have a curious mindset, try things out and have fun!

These tutorials are designed to be self-contained so you should be able to learn the basics of Python by reading through them and doing the exercises at your own pace.

### 0.1 Google Colab can be SLOW

To have an enjoyable google Colab experience, we recommend the following:

- Only have a couple tabs open in your browser
- Work on a relatively fast internet connection
- Don't multitask: try to have as few things running on your computer as possible
- If Colab is running slow, "Cycle the power", i.e. turn your computer off and back on.
- Try different browsers. Google Chrome often does pretty well.
- Be patient. Colab can often be "laggy" so when you click or type something wait a few seconds if you don't see an immediate response
- Even though Colab autosaves your work, we recommend periodically saving your work (File -> Save) or ctrl-S (command-S on Mac).
- Find a meditation practice that you like. Seriously. Coding an be creative and fun but also super frustrating. Meditation can help.

### 0.2 Code Cells and Text Cells

Everything in this document is written in either a Code Cell or a Text Cell.

- **Code Cells** are sections of the document where your can type Python commands and run your code without having to download Python onto your computer. You use "the cloud" (i.e. someone else's computer) to do the processing.
- **Text Cells** are sections (like this one) where text, equations, images, etc. are placed.

To create your own Code Cell or a Text Cell, do the following:

- slowly move your cursor up and down this notebook.
- When you come to the end of a cell, a horizontal line should appear. In the center of the line you should see two buttons labeled "+ Code" and "+ Text"
- Click whichever button you like to create a new cell.

- You can always delete a cell by clicking on it to reveal a menu bar that contains a trash can.

Note: Every section that contains multiple Text and Code cells will have an expand/collapse icon next to it. You can use these to hide (or reveal) the content in that section.

## 0.3 Running your first Python program

The document that you are reading is not a static web page, but an interactive environment called a notebook that lets you write and execute code. Notebooks consist of so-called code cells, which are blocks of one or more Python instructions. For example, here is a code cell that displays the message "Hello World" to the screen. The `print()` command is used to display messages or variables to the screen. Our message can be enclosed in either single or double quotes.

```
print("Hello World")
```

- Position your cursor over the code cell above
- A play button ▶ should appear on the left side of the code cell. Click the play button to execute the cell.
- The message in the print statement should appear below line and a green checkmark should appear to the left of the code box indicating the code has been executed.
- Alternatively, you can also execute the cell by pressing Ctrl + Enter if you are on Windows / Linux or Command + Return if you are on a Mac.

## 0.4 Skill Checks let you practice coding

Each tutorial has a set of Skill Checks (indicated by green check boxes ✅) that you will need to complete to receive credit for each chapter. Here's your first Skill Check:

### ✅ **Skill Check**

- Add a code cell below this cell: position your cursor right below this cell and above the green square. (The green square marks the end of the Skill Check). Two menu options should appear labeled "+ Code" and "+ Text". Click "+ Code".
- In the Code Cell cell, use a print statement to create your own message. When you are done, click the play button to display the new message.
- After you get it to work, create a Text Cell and type the message "I did it!"

🟩

### ✅ **Skill Check**

- Print one or more of your favorite emojis. You can insert an emoji using the following methods:
- On a mac, press ctrl-command-space, then scroll through the emojis
- On a PC, press the windows key and period (.)
- Make sure your emoji is inside quotes like this: `print("🐠🐠🐠")`

## 0.5 Table of Contents

This tutorial is divided into chapters, each in a separate Jupyter Notebook. You can view the contents of a given chapter by clicking on the Contents icon found at the top of the left-hand icon bar (it looks the "hamburger" icon ≡ with 3 dots in front of it). Click on it now to try it out!

## 0.6 Turn the Google AI assistant off

Google Colab documents have a built-in AI assistant (called Gemini at the time of this writing). I've noticed that as I've written these tutorials, Gemini will read my text and then automatically produce sample Python code once I create a Code Cell. Most of the time, Gemini does a pretty good job of "understanding" my intention and writing code that I would have written. It's actually pretty amazing.

In fact, AI tools have revolutionized how folks code. AI assistants like chatGPT, GitHub Copilot, Cursor, and many others can speed up code production immensely. Later in the course, we will explore some of these tools. Overall, we think AI is a good thing for coding, not a bad thing.

However, when you are first learning the basics of Python, and perhaps coding in general, the AI assistant can make the execises in these tutorials too easy. Psychologists who study learning talk about cognitive load theory and the idea of "desirable struggle":

- **Desireable struggle**: When a task is hard enough to force you to actively retrieve, reason, and problem-solve, but not so hard that you get stuck in frustration.
- **Overload:** If the challenge is too high (too much cognitive load), you can shut down, feel lost, or just copy without understanding.
- **Underload**: If the challenge is too low (everything done for you), you don’t engage deeply, so nothing sticks.

If you leave the Autocomplete setting in Gemini turned on, it will automatically produce many of the coding exercises for you and you can basically "turn off your brain." In other words, you won't learn to code.

For these reasons, we ask that you turn off Gemini autocomplete while you are going through these tutorials. Here's how to do it:

- In your Google Colab document, click "**Tools**" from the upper menu bar and select "**Settings**"
- In the Settings window, select "**Editor**"
- Scroll down and you should see a list of chceck boxes starting with "**Show context-powered code completions**"
- Make sure all of these check boxes are **unchecked**
- While you are there, I recommend setting the "**Indentation Width in Spaces**" to 4 (the default is 2). This will be important when we get to loops and if statements later in these tutorials.
- When you are done, don't forget to click **Save**

Once you save the settings, they will be used as the defaults when you open other Colab documents. If you get stuck and you want to use the AI assistant, you can always turn it back on by ckecking the appropriate boxes.

## ✅ Skill Check

Please sign the following statement by typing your name into the text box:

> I understand the importance of "desirable struggle" in learning and I agree to use AI sparingly and only when I feel I have exhausted other options.

- I have disabled the Google Colab autocomplete functions by following the instructions above.
- I agree to work through these tutorials without relying on AI tools.
- If I get stuck, I agree to reach out to the instructor or other students if possible to discuss ways of approaching the problem.
- If I do use AI to help solve a problem, I agree to disclose its use, i.e. "I asked chatGPT for ideas about how to approach this problem" or "I turned on Gemini autocomplete for this problem", etc.

Signature (type your name):

🟩

## 0.7 Comments

It is often a good idea to comment your code. A comment is a message to yourself (or others reading your code) to describe what your code is doing. Comments are created using the hash tag #. Anything placed after the hash tag on a given line will be interpreted as a comment (and not Python code). Comments may be placed on a line of their own or after Python code as shown in the following example:

```
# this is my first Python program
print('hi')     # this line prints the message "hi"
```

Play the above code block to execute it using one of the methods described above. You should see the message "hi" appear under the code block.

## Key Points

- This Python tutorial is written in Jupyter Notebooks and hosted on Google Colab.
- The tutorial is interactive, which allows you to try out your coding skills and run your code directly in the tutorial
- The table of contents icon found in the left-hand menu lets you navigate the sections of each tutorial
- Each tutorial has a set of Skill Checks (indicated by green check boxes ✅) that you will need to complete to receive credit for completing each chapter
- We ran our first Python program in this tutorials, a "hello world" print statement.
- Print statements allow your code to display text, variable values, etc. to the user
- We discussed the role of AI in learning to code and asked you to agree to turn off the AI autocomplete functions in Google Colab

This tutorial is a modified adaptation of "Python for Physicists" © Software Carpentry