



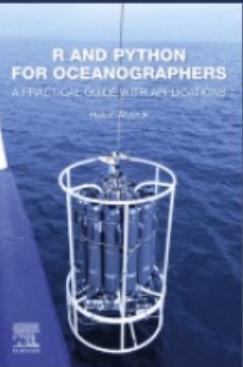
# Intro to Python & HW4

# Read R & Python for Oceanographers, Ch 1 (Python part only)

## R and Python for Oceanographers

*A Practical Guide with Applications*

Book • 2019



Authors:  
Hakan Alyuruk

↓ About the book

### Browse this book

↓ By table of contents

#### Book description

R and Python for Oceanographers: A Practical Guide with Applications presents the uses of scientific Python packages and R in including both script code and applications. It covers the basic concepts of scientific computing with R and Python, and provides practical examples of how to use them in oceanographic research.

including both script code and applications ... [read full description](#)

#### 1.7 Introduction to Python

Python is an interpreted, general-purpose, object-oriented and high-level programming language. It was conceived by Guido van Rossum in the late 1980s, and its first implementation started in 1989. Python is developed as a successor of ABC language at Centrum Wiskunde & Informatica (CWI) in the Netherlands. In the development of Python, van Rossum also influenced from Modula-3. As a result, Python has been evolved as a new language that includes improvements over ABC and Modula-3 [3, 4].

12 R and Python for Oceanographers

Currently, the development of Python language is organized and supported by Python Software Foundation (PSF).

Some advantages of Python as a data science tool are:

- It is simple, readable, and easy to learn.
- Python interpreter can run on multiple platforms: Windows, Linux, and Mac OSX.
- It is possible to express functions with fewer codes compared to other languages.
- It is an open source language and can be freely used and distributed.
- There are many libraries for scientific computing.
- It is possible to use Python in combination with other languages such as C or C++.

# What is Python and why would we use it?

- Did anyone have frustrations with the last two Excel exercises (other than just downloading Argo data)?

# What is Python and why would we use it?

- Did anyone have frustrations with the last two Excel exercises (other than just downloading Argo data)?
- Python is a scripted language (we write code to make it do stuff)
- It's HIGHLY reproducible (Excel, not so much)
- Open source and free
- But ^ comes with some frustrations, too

# Why use Python, Part II

- Free!
- Repeatable coding, you don't have to repeat a whole process with Excel every time you get new data
- Used by top professionals, from YouTube coders to NASA researchers
- Can be applied to any field of science or engineering
- One of the most helpful and lucrative skills in science

# Python vs. Jupyter vs. Colab

- Python is a language (analogous to English). You write code in Python (or text in English). You could write Python code (or English) anywhere—on a napkin, your notebook, in a text message, in a document on your computer.

# Python vs. Jupyter vs. Colab

- Python is a language (analogous to English). You write code in Python (or text in English). You could write Python code (or English) anywhere—on a napkin, your notebook, in a text message, in a document on your computer.
- Python can do extremely sophisticated machine learning and very basic math. It is the backbone of many famous websites (Instagram, Spotify, Netflix, etc.)
- But it has very specific rules (grammar) that must be followed for it to work

# Python vs. Jupyter vs. Colab

- Python is a language (analogous to English). You write code in Python (or text in English).
- **Jupyter is a program** in which you can execute Python code (continuing our analogy, it's like Microsoft Word). You open Jupyter on your computer in order to run Python code. Jupyter calls its files “notebooks.”

# Python vs. Jupyter vs. Colab

- Python is a language (like English). You write code in Python (or text in English).
- Jupyter is a program in which you can execute Python code (continuing our analogy, it's like Microsoft Word).
- Colab is a **free online website** that allows you to run notebooks in the cloud without downloading anything. You just need a browser. Sort of like Google Docs.

# There are many ways to run Python

- The very hard way: using a text editor and running code from the command line
- The medium hard way: installing Python and its packages using the Python package installer (PyPI) and using Jupyter
- The medium way: installing Python and packages using a tool called Anaconda and using Jupyter notebooks for desktop programming (I recommend this)
- The easiest way: Google Colab. 100% in the cloud—all you need is a browser (I will teach this)



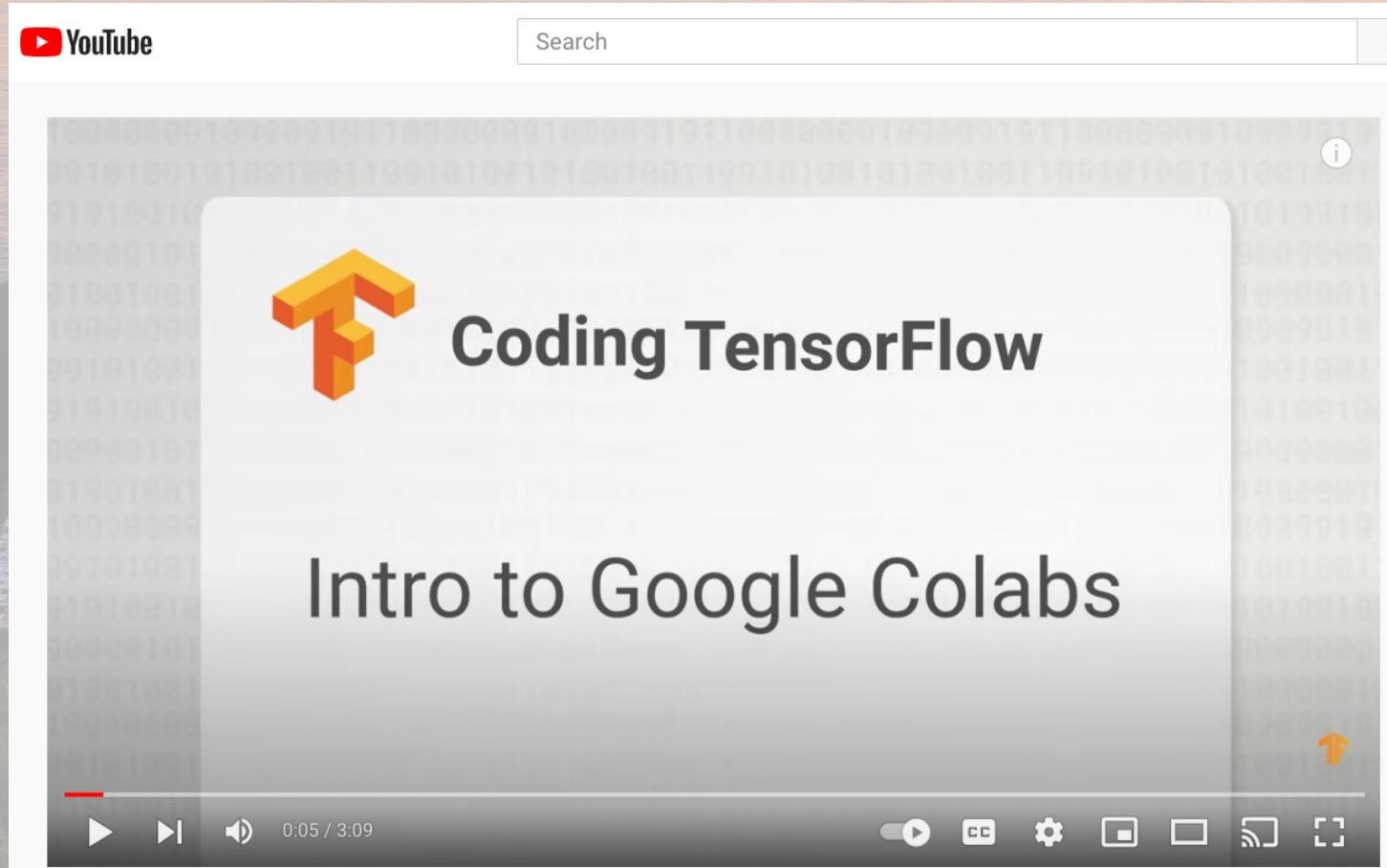
# Colab creates new challenges

- Running Python on your computer means easy access to data files (e.g., BATS.xlsx would be right there)
- Running in the cloud means you have to tell Google where to find data—can be messy
- You have to be extra careful to save your work when working in a browser

# Additional and somewhat confusing details

- Jupyter notebooks can execute code written in other languages, too, but we won't do that in this class
- Colab can only execute Python
- You can (in fact, are encouraged to!) download Python on your computer via Anaconda, but this can get a little complicated. We'll discuss it later this semester. Just use Colab for now, but don't count on using it in other Python-focused classes.

# Intro to Colab



<https://www.youtube.com/watch?v=inN8seMm7UI>

# Intro to Colab

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help Cannot save changes

Share

Table of contents

+ Code + Text

RAM Disk Editing

Getting started

- Data science
- Machine learning
- More Resources
- Machine Learning Examples
- Section

What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

**Getting started**

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

86400

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

0s completed at 8:21 PM

<https://colab.research.google.com/notebooks/intro.ipynb>

# Intro to Colab

The screenshot shows the Google Colab interface with the title "Overview of Collaboratory Features". The left sidebar contains a "Table of contents" with sections like Cells, Code cells, Text cells, Adding and moving cells, Working with python, Integration with Drive, and Section. The main content area is titled "Cells" and describes notebooks as lists of cells containing explanatory text or executable code. It details how to run code cells using various methods (Play icon, Cmd/Ctrl+Enter, Shift+Enter, Alt+Enter) and how to run multiple cells via the Runtime menu. A code cell example is shown with the output "10". Below this, the "Text cells" section is visible, mentioning markdown syntax and LaTeX rendering.

File Edit View Insert Runtime Tools Help

Share

RAM Disk Editing

Table of contents

Cells

Code cells

Text cells

Adding and moving cells

Working with python

System aliases

Magics

Automatic completions and exploring code

Exception Formatting

Rich, interactive outputs

Integration with Drive

Commenting on a cell

Section

+ Code + Text

▼ Cells

A notebook is a list of cells. Cells contain either explanatory text or executable code and its output. Click a cell to select it.

▼ Code cells

Below is a **code cell**. Once the toolbar button indicates CONNECTED, click in the cell to select it and execute the contents in the following ways:

- Click the **Play icon** in the left gutter of the cell;
- Type **Cmd/Ctrl+Enter** to run the cell in place;
- Type **Shift+Enter** to run the cell and move focus to the next cell (adding one if none exists); or
- Type **Alt+Enter** to run the cell and insert a new code cell immediately below it.

There are additional options for running some or all cells in the **Runtime** menu.

```
[ ] a = 10
a
10
```

Text cells

This is a **text cell**. You can **double-click** to edit this cell. Text cells use markdown syntax. To learn more, see our [markdown guide](#).

You can also add math to text cells using **LaTeX** to be rendered by **Math Jax**. Just place the statement within a pair of \$ signs. For example

[https://colab.research.google.com/notebooks/basic\\_features\\_overview.ipynb](https://colab.research.google.com/notebooks/basic_features_overview.ipynb)

# Assignment:

Start with “Overview of Colaboratory Features” notebook from previous slide

OCN350\_HW4 - Overview of Colaboratory Features

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings

RAM Disk Editing

Table of contents

- Cells
- Code cells
- Text cells
- Adding and moving cells
- Working with python
  - System aliases
  - Magics
  - Automatic completions and exploring code
  - Exception Formatting
  - Rich, interactive outputs
- Integration with Drive
- Commenting on a cell

Assignment

Section

+ Code + Text

You can resolve and reply to comments, and you can target comments to specific collaborators by typing `+[email address]` (e.g., `+user@domain.com`). Addressed collaborators will be emailed.

The Comment button in the top-right corner of the page shows all comments attached to the notebook.

Assignment

1. First, make this cell using the `+Text` command above.
2. Write the contents of this screenshot into your own notebook text cell.
3. Make sure that the word "Assignment" shows up as a Header, not just body text.
4. Write down the rest of the contents of this cell in your notebook (i.e., all of this text box) and make sure that everything after the word "Assignment" shows up in a numbered list like this.
5. Move onto the next slide and complete the simple arithmetic code portion.

[2] `x = 3  
y = 4  
# You write the rest.`

0s completed at 9:17 PM

See [https://colab.research.google.com/notebooks/markdown\\_guide.ipynb](https://colab.research.google.com/notebooks/markdown_guide.ipynb) for Markdown syntax

# Requirements

1. To do this, you must have a Colab account (this only requires a normal Google/Gmail account and is completely free)
2. Go to [https://colab.research.google.com/notebooks/basic\\_features\\_overview.ipynb](https://colab.research.google.com/notebooks/basic_features_overview.ipynb)
3. Save a copy in Drive (create a new OCN 350 folder in Drive; we'll use it a lot!)
4. READ and Run each of the cells (either by clicking the play arrow or by clicking “Shift + Return”)
5. Add the portion described on previous slide to the bottom



# fill this code cell

Run cell (⌘/Ctrl+Enter)  
cell has not been executed in this session

# Requirements (cont'd)

6. Add a code cell. Type the following:

```
x = 3  
y = 4  
z = '5'  
a = x*y  
b = y*z
```

7. Add two additional lines to print out the value of a and the value of b.

8. Add a Markdown cell and explain why a and b are what they are.

9. Calculate  $c = x^2$  using Python's exponent notation (look it up) and print out the value of c to the screen

10. Calculate  $d = \sqrt{y}$  using Python's exponent notation and print out the value of d to the screen.

# Submission

- Save and submit the .ipynb file in Canvas

# Final notes

- Notebooks are nice because you can add code, formatted text, and output (graphs, tables, printed values, etc.)
- But many other ways to write and execute code
- We'll use Jupyter notebooks (hosted by Colab) for ease of use and presentation, but many other great options

```
PJB-20190329_googleDoc_decode_16BitIMU_plotly.py > ...
87  def decoded_to_df(df, decoded):
88
89      i = 0
90      # rest vars
91      type_name = np.nan
92      time = np.nan
93      temp = np.nan
94      epoch = np.nan
95      wet = np.nan
96      Ax = np.nan
97      Ay = np.nan
98      Az = np.nan
99      lat = np.nan
100     lon = np.nan
101     text = np.nan
102     batt_mv = np.nan
103
104     try:
105
106         while(i < (len(decoded))):
107             type = (decoded[i]>>4)
108
109             # print("type = ", type)
110
111             # deal with padding if it exists
112             if (type == 0):
113                 type_name = 'Padding'
114                 i = i + 1
115                 # print("type = " + str(type_name))
116                 # print()
117                 continue
118
119             # grab time (wait until after dealing with padding)
120             time = (((decoded[i] & 0x0F) << 16) + (decoded[i+1] << 8) + decoded[i+2])/10.0
121
122             # decode data type
123
124             if (type == TEMP_DATA_TYPE):
```