First Steps with Python

Matt Steele

Welcome

- Matt Steele
- Data Cleaning and Analysis with Python
- Data Visualization and Presentation with Python
- Date Services Workshops

Resources

- Official Site for Python
- Official Site for Anaconda
- Jupyter Notebooks
- O'Reilly Learning Platform
  - Python for Data Analysis, 3rd Edition
  - Python for Data Analysis: Step-By-Step with Projects

Agenda

1. Introduction to Python and Python Environments
2. Functions and Arguments
3. Creating Variables
4. Comparisons, Conditionals, and Loops
5. Packages and Libraries

Why Python

- Open-source
  - Rich Ecosystem of Libraries
  - Integration with Other Tools and Languages
  - Platform Independent and Non-proprietary
  - Reproducibility and Transparency
  - Integrates into Proprietary Software
  - Share-able
  - Add-on Libraries

Anaconda Navigator

https://www.anaconda.com/

Anaconda is a popular and powerful distribution of Python and R programming languages specifically tailored for data science, machine learning, and scientific computing.

What is JupyterLab?

https://jupyter.org/

- JupyterLab is an interactive development environment for notebooks, code, and data.
- It offers an enhanced interface for working with Jupyter Notebooks, terminals, text editors, and file browsers all in one place.

Opening JupyterLab through Anaconda Navigator

1. **Open Anaconda Navigator**
    - Follow the steps above to launch Anaconda Navigator.
2. **Open JupyterLab**
    - Click on **JupyterLab** within Anaconda Navigator to open it.

Set up your Environment

3. **Select Your Working Directory**
    - Use the directory navigator on the right to sel
4. **Start a New Notebook**
    - Click on **File** > **New Notebook** > **Python 3** to s
5. **Select the Python Kernel**
    - Ensure that the Python kernel is selected. You
6. **Save Your Notebook**
    - Click on **File** > **Save Notebook As** to give your

Setting Your Preferences in JupyterLab

Cells

Unit where you can write and run code, document their work, and format outputs.

There are three main types of cells:

1. **Code Cells** – For writing and running Python code.
2. **Markdown Cells** – For writing formatted text (like headings, lists, links).
3. **Raw Cells** – For displaying unformatted text or data.

Working Data

In Python, you can work with data by defining variables, lists, or other data structures. Let’s start by entering some data and practicing basic operations.

Lists

Lists are used to store multiple items in a single variable. Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

```
1 my_var = [1, 2, 3, 4]
2 print(my_var)
```

Import

Using import for a library will bring all functions for the library into your workspace.

```
import pandas as pd

# Example: Reading a CSV file with pandas
# Assume we have a 'data.csv' file
reviews_df = pd.read_csv("customer_reviews.csv")

# Display the first 5 rows of the data
print(reviews_df.head())
```

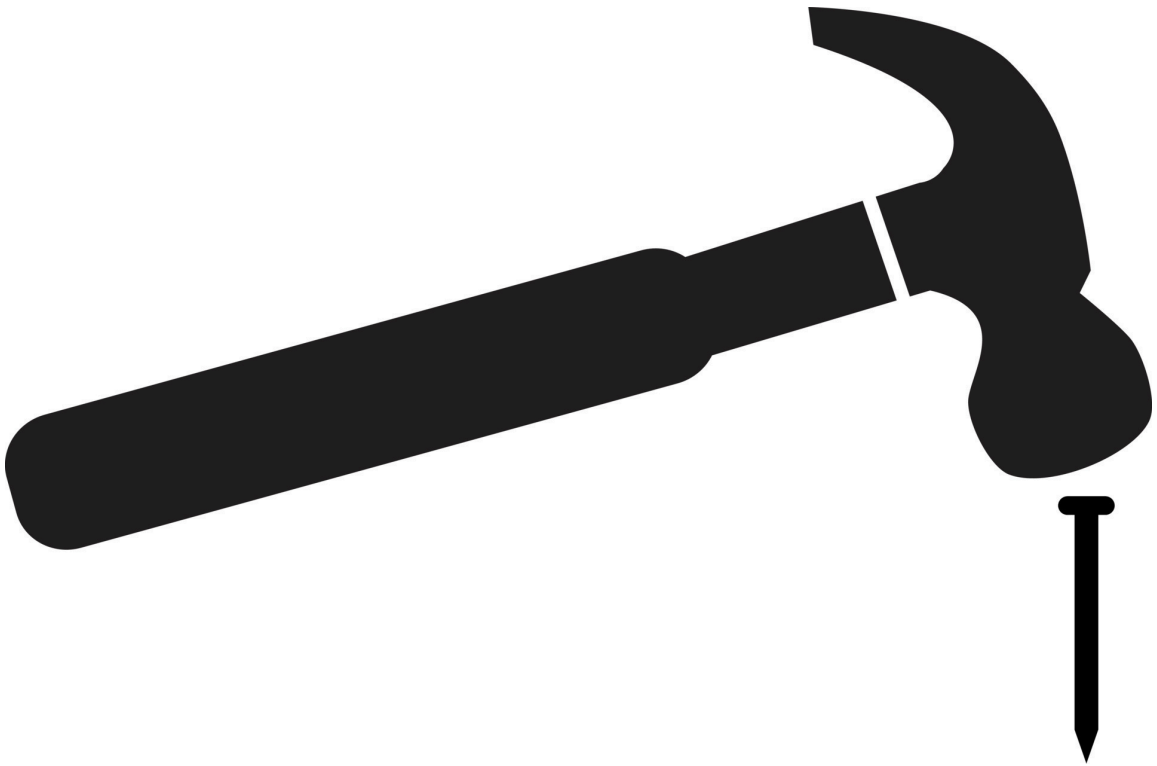
Functions

used to give commands to perform a tasks.

```
round(3.1415926535897793)
```

Arguments

used to provide the details about how your function operates



```
round(3.141592653589793, 5)
```

Documentation

provides the necessary information, explanations, examples, and guidance to help you learn, understand, and effectively use R functions and packages.

- Base Python Standard Library
- Pandas Documentation
- NumPy Documentation

Getting Help in Python

1. Help with Functions and Libraries

- Use the `help()` function to get information about a function or library.

```
help(print)  # Get help on the print function
  help('pandas')  # Get help on the pandas library
```

Variables

used to store and work on data (numbers, words, tables, and more).

Creating and Using Variables

Example 1: Storing Numbers

```
# Storing an integer
age = 25

# Storing a floating-point number in Fahrenheit
temperature_fahrenheit = 100.6

# Converting Fahrenheit to Celsius
temperature_celsius = (temperature_fahrenheit - 32) * 5 / 9


print(age)
print(temperature_celsius)
```

Naming Variables

- Use descriptive and meaningful names that indicate the purpose of the object
- Use lowercase letters.
- Use underscores to separate words (e.g., my_variable_name).
- Avoid using reserved words or functions (e.g., “if,” “else,” “for,” “function”).

Data Types in Python

Type	Description
Integer	Represents whole numbers. Example: `5`, `-3`, `42`.
Float	Represents real numbers (numbers with decimal po
String	Represents text, words, and strings of characters. E
Boolean	Represents binary values used for decision-making
Date	Handles dates and times. Use the `datetime` modu `9, 13)`.

Creating a Function

In Python, functions allow you to package a block of code that can be reused throughout your program. This makes your code more organized, readable, and reusable.

```
# Step 1: Create Variables

name = input("What is your name: ")
age = input("What is your age: ")


#Step 2: Define the function

def demo(name, age):
    print(name, age)

#Step 3: Call the function

demo(name, age)
```

Running Code

In Python, indents are used to define the structure of the code. Unlike some other programming languages that use curly braces {} or keywords to indicate blocks of code, Python uses indentation. This makes the code more readable and easier to understand.

```
1 def check_number(number):  # Starts the function; the indented lines below
2     if number > 0:
3         print("The number is positive.")
4     elif number < 0:
5         print("The number is negative.")
6     else:
7         print("The number is zero.")  # This is where the function's code e
8
9 print("Hello World")  # This starts a new section of code
```

Boolean Operators

Equal	==
Not Equal	!=
Greater Than	>
Less Than	<
Greater Than or Equal to	>=
Less Than or Equal to	<=

Comparions

Use comparison operators to determine if objects in python are identical to each other.

```
#Comparison

fruit1= "apple"
fruit2= "orange"

#Are items/variables == to each other
fruit1 == fruit2

##boolean operators - > Can be combined variables/inputs with boolean opera

6 > 3 and 1 < 2
```

Conditions

If	If the conditions made in the statement are met then perform the operation.
Else	If the conditions made in the statement are NOT met then perform another operations.
Elif	If the condition made in the statement are NOT met by either the IF statement or the ELSE statement perform this operation.

Conditionals

Python operators that looks to see if an object meets sta

```
1  water_cold = True
2
3  if water_cold:
4    print("Brrrrrr")
5
6  if water_cold:
7    print("Brrrrr")
8  else:
9    print("I can drink this)
10
11 water_temp = 0
12
13 if water_temp < 0:
14   print("Brrrrrr")
15 elif water_temp > 100:
16     print("tccchhhhh")
```

Loops

The requested operation will repeat until it is told to stop.

For Loops

The ‘for’ loop is typically used when you know the number of ite

syntax: ***for*** *variable* ***in*** *iterable*

- variable: This is a placeholder variable that will take on the va
- iterable: This is the object over which the loop iterates. It can

```
#for loop

fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

While Loops

The while loop is typically used when you don’t know the number of iterations in advance or when you want to loop as long as a certain condition is true.

```
x = 0
while x <= 20:
    print(x)
    x = x + 1

print("Stop looping")
```

Packages & Libraries

packages are like toolkits or collections of pre-built functions, data sets, and tools that extend the capabilities of the R programming language.

Packages

Install

You must install a package before you can load it. *But you only need to install it one time.*

```
!conda install pandas

!conda list
```

Packages

Import

Using import for a library will bring all functions for the library into your workspace.

```
1  import math
2
3  # Now you can use any tool from the math toolbox
4  print(math.sqrt(16))  # Output: 4.0
5  print(math.pi)        # Output: 3.141592653589793
```

Packages

From

Using from will just bring one particular function from the library into your workspace.

```
1  from math import sqrt
2
3  # Now you can use just the sqrt tool directly
4  print(sqrt(16))  # Output: 4.0
```

Packages

Alias

To make the library easier to reference, you can assign it an alias using the as keyword.

```
1 import pandas as pd
2
3 my_var = pd.Series(my_var)
4
5 my_var.mean()
```

Explore your Environment

```
# see

!conda list

# search

!conda search beautifulsoup

# update

!conda update beautifulsoup4
```

Next Week

- Entering Data
- Loading and Exporting Data Frames
- Cleaning and Transforming Data
- Data Analysis Tools

Conclusion

- Matt Steele
- Data Cleaning and Analysis with Python
- Data Visualization and Presentation with Python
- Date Services Workshops