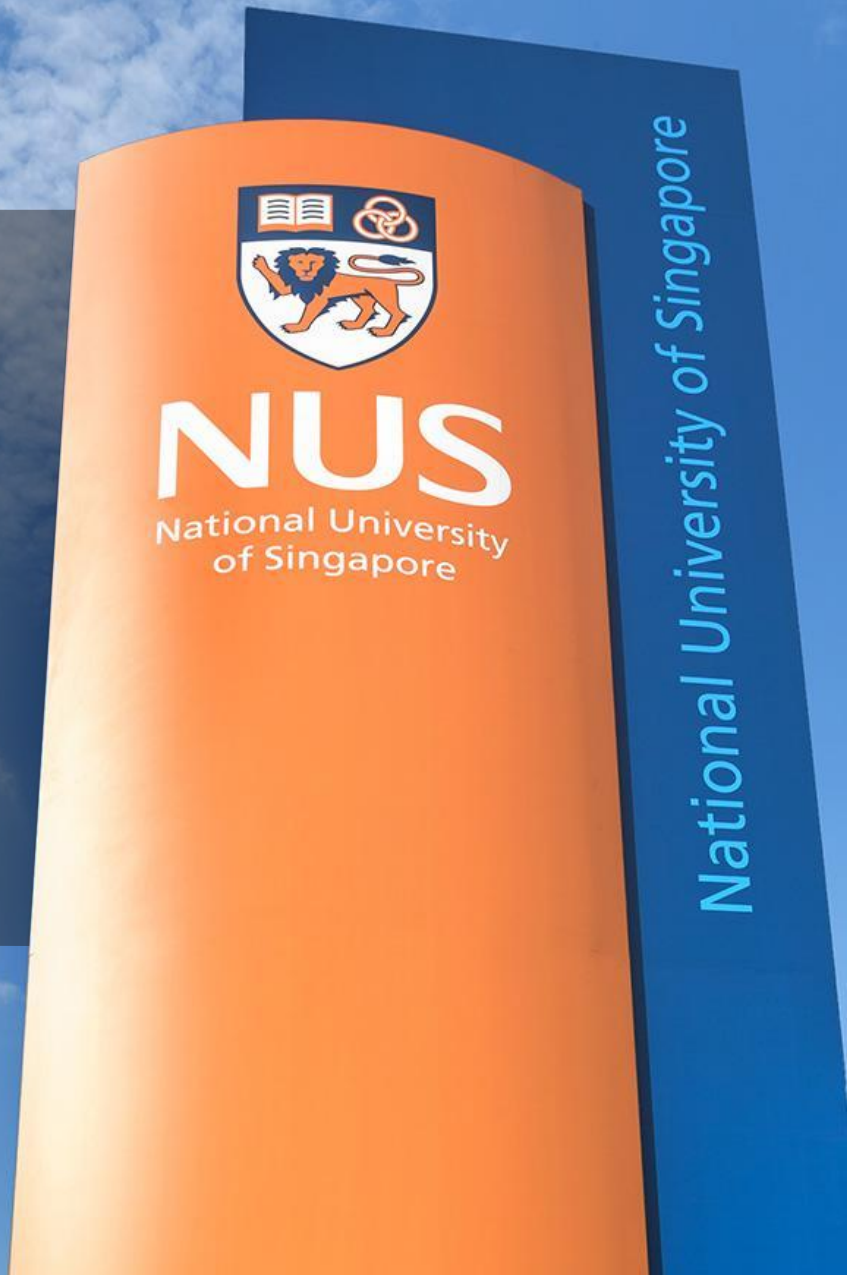
Quantitative Insights and Data Analytics I

Day 1

SingStats
Division: HSE



Schedule I

Time	Agenda
9:00 - 10:30 am	● Introduction ● Intro to Python and JupyterLab ● Package management
10:30 - 10:50 am	Break
10:50 - 12:30 pm	Python data structures: ● **Part 1:** Lists, dictionaries, tuples, sets ● **Part 2:** Series and dataframes
12:30 - 1:30 pm	Lunch

Schedule II

Time	Agenda
1:30 - 3:00 pm	Intro to data handling with Python: ● Reading and writing CSV files ● Basic data operations: filtering, sorting, merging ● Introduction to data exploration
3:00 - 3:20 pm	Break
3:20 - 4 pm	● Introduction to data visualization with Python
4 - 5 pm	● Project-based assessment

Overarching Agenda:

Course 1: Introduction to Python

Course 2: Modelling, Data Manipulation, Data visualization

Course 3: Applied case studies + advanced topics in Python

Introductions

My name is Viktoria:

- Research Associate
- Previously a Data Systems Engineer
- Masters from University of British Columbia
- Bachelors from Yale-NUS College
- Experience with Python, R, Javascript, Google Cloud, databases, LLMs, etc.

Introductions

Let’s go around the room with introductions:

- Your name
- Work focus
- What do you hope to learn

Expectations:

- To maximize learning, extensive AI usage is discouraged
- This is a hands-on workshop
- Questions are encouraged - your experience improve our future discussions
  - The more we know about you, the better
- All code samples and exercises will be provided for future reference (Jupyter Notebook format)
- Assignment submission is mandatory

Why Python for Data Analysis?

Key Advantages:

- **Open Source:** No licensing costs or vendor lock-in for companies or individuals
- **Extensive libraries** for data science (pandas, numpy, scikit-learn)
- **Large community** for support and resources
- **Versatility:** One language for data cleaning, analysis, ML, visualization, and web apps

Python vs Other Options

Python vs R

Python: General-purpose language with strong data science performance

R: Mode specialized for statistical modelling and academic applications

Python vs SQL

SQL: Specialized for database maintenance, editing, queries

Python: Can work with SQL + additional data processing and analysis capabilities

Python vs Excel

Excel: Good for quick visualizations and simple analyses

Python: Automation, reproducibility, version control, handling large datasets

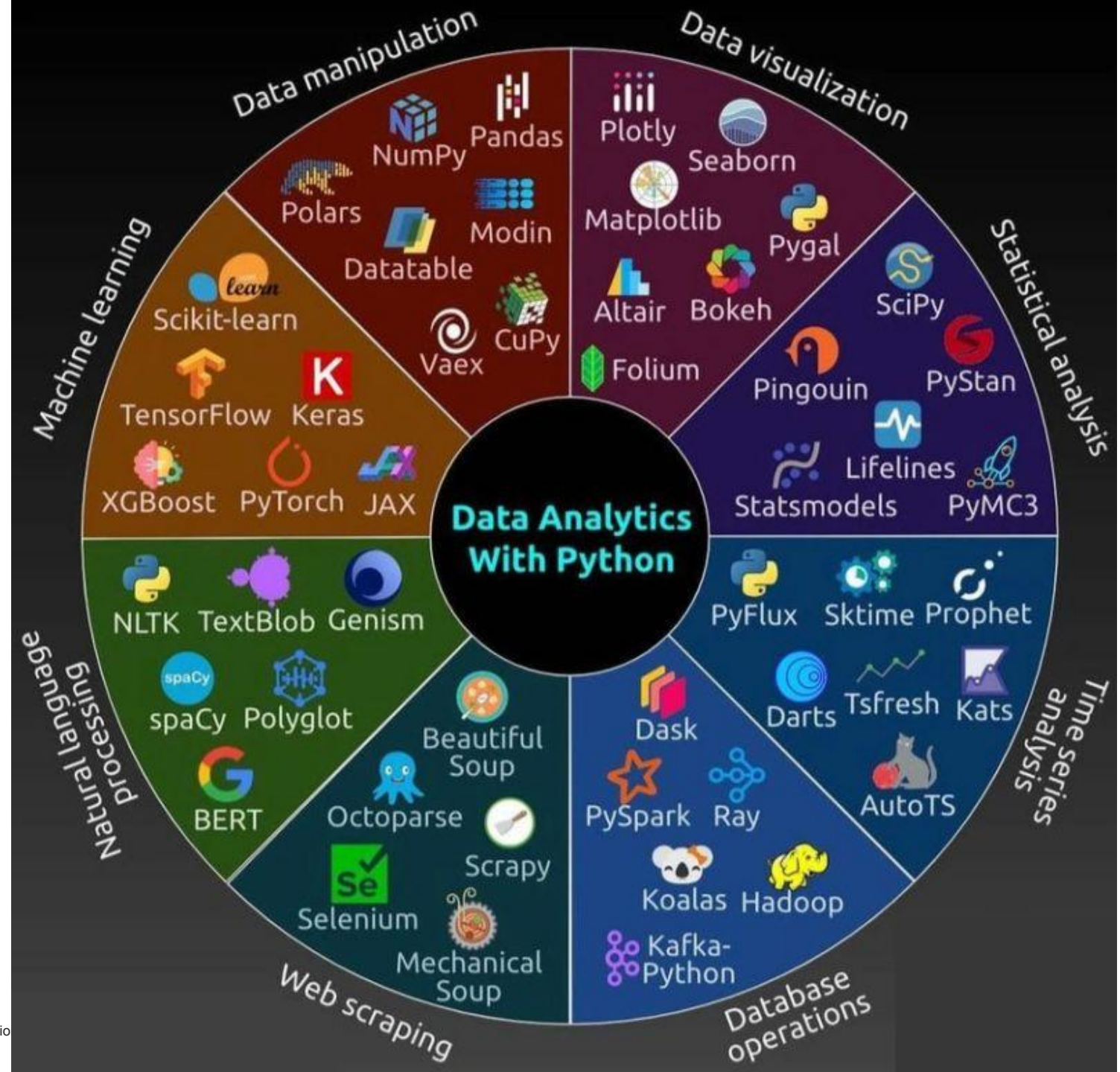
Python Libraries (just some!)

Data Science:

- pandas: Essential for working with structured data (like Excel/csv sheets)
- numpy: A package for working with arrays and math
- matplotlib/seaborn/altair: Data visualization
- scikit-learn/TensorFlow: Machine learning

When you need to use a package, you need to (1) have the package installed on your computer, and (2) import the package to Python before using it.

- **If you need to install a package, you need to open the terminal and type in “pip install [package name]” in the terminal.**



Course Content Location

Desktop > FOSVL > Course Content > For SingStats > HSE

Copy/drag the HSE folder to your Desktop

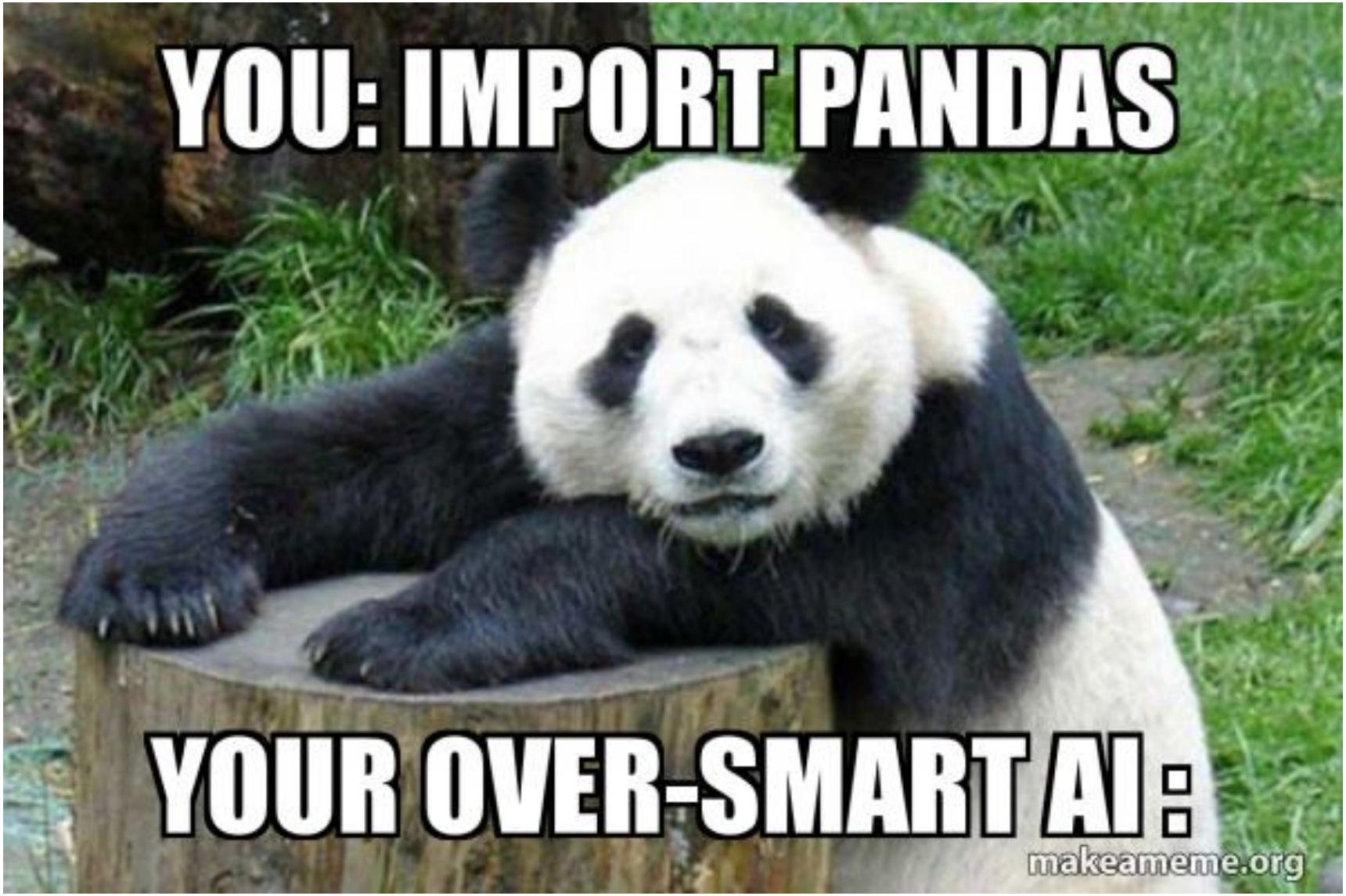
Put your name on the folder name:

HSE_[name]

```
493
494
495
496
497
498
499
500 import numpy as pandas
501 import pandas as numpy
```

Command line

Standard Input: Interactive Console Text



Jupyter Notebook (file format)

Jupyter Notebooks (.ipynb)

- Interactive documents combining:
  - Live code
  - Rich text (Markdown)
  - Visualizations
  - Mathematical equations
- Cells can be executed independently
- Perfect for:
  - Data analysis
  - Research
  - Teaching
  - Documentation

Python Files (.py)

- Traditional script files containing pure Python code
- Executed from start to finish in one go
- Best for:
  - Production code
  - Reusable modules
  - Command-line applications
  - Software development

```
def greet(name):
    print(f"Hello, !")

greet("World")
# Output: Hello, World!
```

Jupyter Notebook

Jupyter Notebooks (.ipynb)

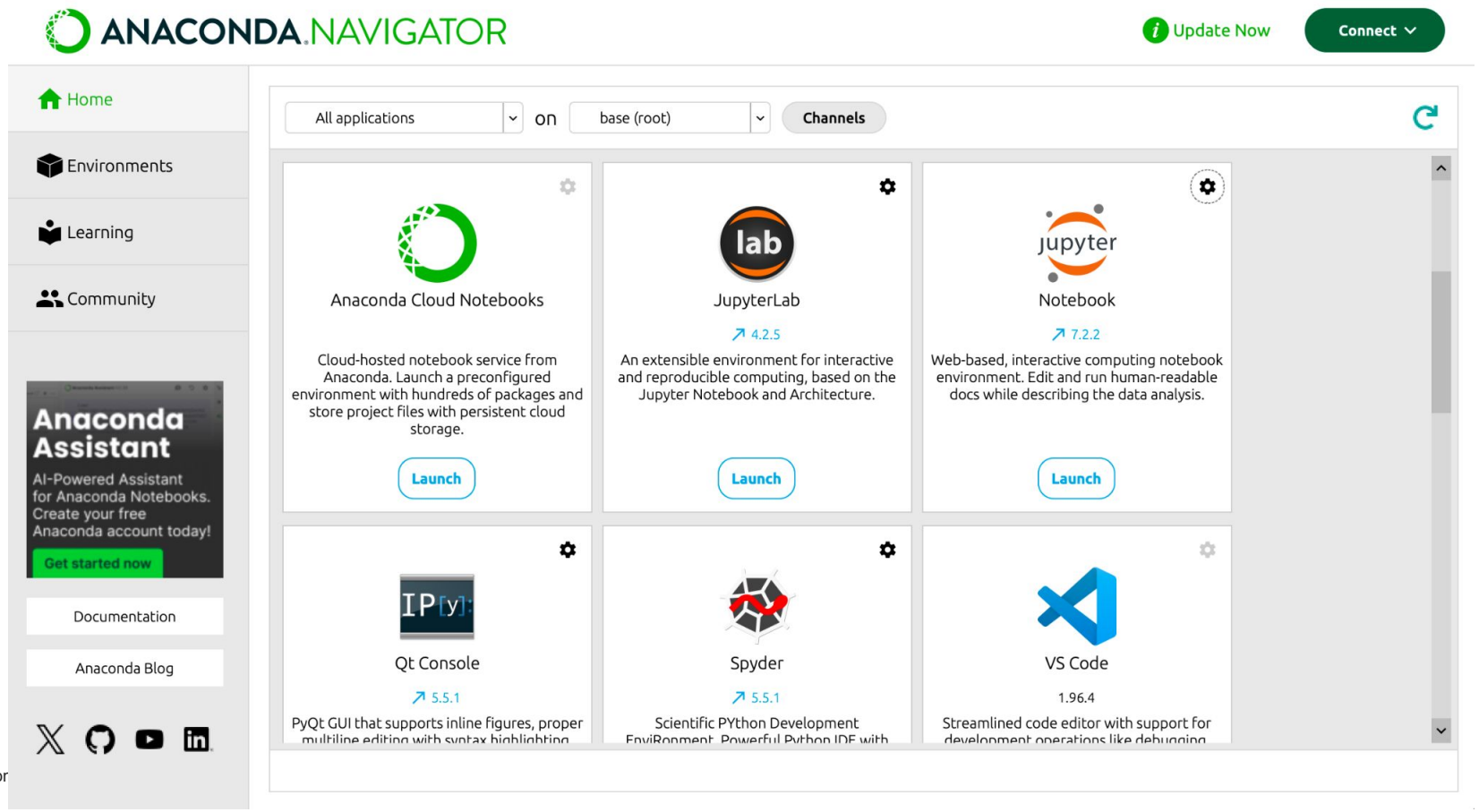
- Interactive documents combining:
  - Live code
  - Rich text (Markdown)
  - Visualizations
  - Mathematical equations
- Cells can be executed independently
- Perfect for:
  - Data analysis
  - Research
  - Teaching
  - Documentation

Throughout this course, we will use this notebook:

Day1_HSE.ipynb

JupyterLab Interface walkthrough

Let’s open JupyterLab:



Jupyter Lab Interface

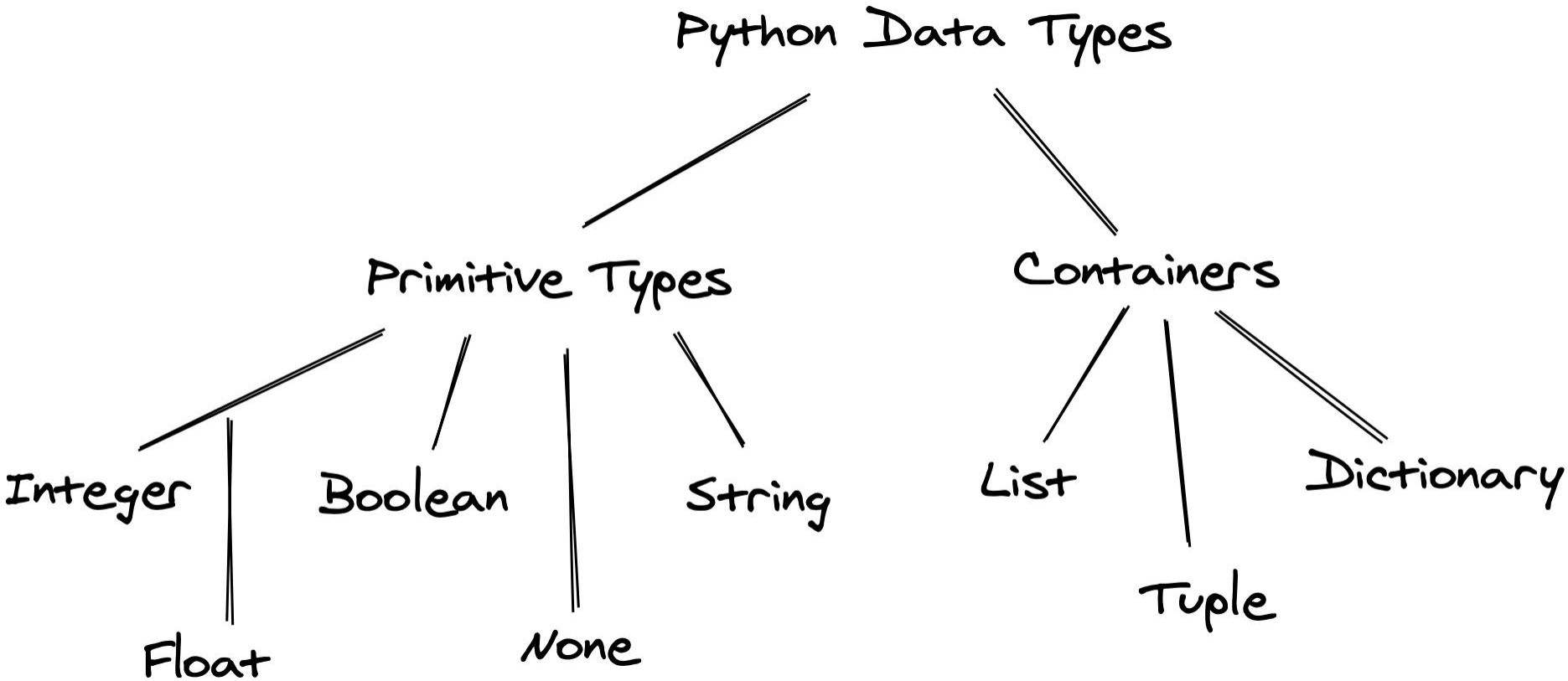
The JupyterLab interface consists of a main work area containing **tabs of documents and activities**, a collapsible left **sidebar**, and a menu bar. The left sidebar contains a file browser, the list of running kernels and terminals, the command palette, the notebook cell tools inspector, and the tabs list.

JupyterLab time!

We will cover the following sections:

- Python Introduction
- Modules and Packages
- Variables and Objects

Section II: Data Structures



Built-in Python Data Structures

1. **Use Lists when:**
   - Need ordered elements
   - Will modify contents frequently
   - Need duplicate elements
   - Need slicing operations

2. **Use Tuples when:**
   - Data shouldn't change
   - As dictionary keys
   - Returning multiple values from functions
   - Performance is critical

3. Use Sets when:

- Need unique elements
- Performing logical operations (union, intersection)
- Checking membership
- Order doesn't matter

4. Use Dictionaries when:

- Need key-value pairs
- Need lookups
- Working with JSON-like data

Here is a good cheat sheet: link.

JupyterLab time!

We will cover the following sections:

- Python Data Structures
- Pandas Data Structures

Lists

Ordered, mutable sequences for storing collections of items

Examples:

```
fruits = ['apple', 'banana', 'orange']
fruits.append('grape')       # Add at end
fruits.insert(0, 'kiwi')    # Add at specific position
fruits.remove('banana')      # Remove specific value
```

Key Characteristics:

- Ordered, mutable sequence
- Use when: Order matters and elements need to be modified

Dictionaries

Key-value pairs for storing and retrieving data by unique keys

Example:

```
person = {'name': 'John',
    'age': 30,
    'city': 'New York'
}
```

Key Characteristics:

- Key-value pairs, unordered (Python 3.7+ preserves insertion order)
- Use when: Need to associate values with unique keys

Tuples

Immutable sequences for storing fixed collections. Tuples are like lists but they cannot be changed.

Key Characteristics:

- Ordered, immutable sequence
- Use when: Data shouldn't be modified and/or as dictionary keys

Examples:

```
point = (3, 4)
```

```
person = ('Hello', 30, 'Singapore')
```

Sets

Unordered collections of unique elements

Characteristics:

- Unordered collection of unique elements
- Use when: Need to ensure uniqueness or perform set operations

Example:

```
fruits = {'apple', 'banana', 'orange'}
numbers = set([1, 2, 2, 3, 3, 4])  # Creates {1, 2, 3, 4}
```

Curly vs. Square

Summary:

- **Round brackets ()**: Function calls, tuples, grouping, generator expressions.
- **Square brackets []**: Lists, indexing/slicing, list comprehensions.
- **Curly brackets {}**: Dictionaries, sets, set comprehensions.

Additional information

```
# Using ? (IPython/Jupyter)
df?                    # Get basic info about DataFrame object
df.head?               # Info about head() method
pd.read_csv?           # Details about read_csv function

# 2. Using help()
help(pd.DataFrame)     # Detailed documentation
help(df.groupby)       # Help on groupby method
help(len)              # Help on built-in functions

# Examples of useful queries:
df.dtypes?             # Info about data types
pd.merge?              # Understanding merge options
```

Pandas Data Structures

Pandas: Series

1D labeled array that can hold data of any type

- Size-immutable
- Values mutable
- Labels/index

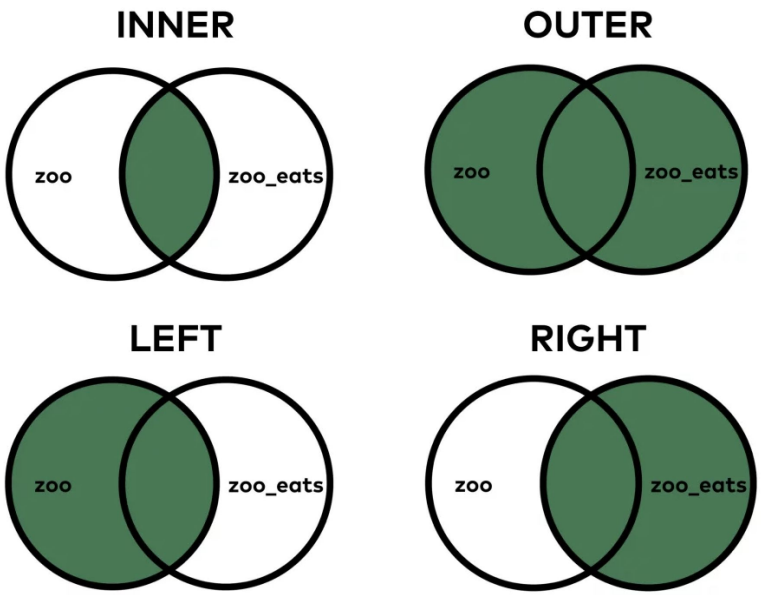
Pandas: Dataframes

2D labeled data structure with columns of potentially different types

- Column-oriented
- Size-mutable
- Labels for both rows and columns

Section III: Data Handling

Data Merging



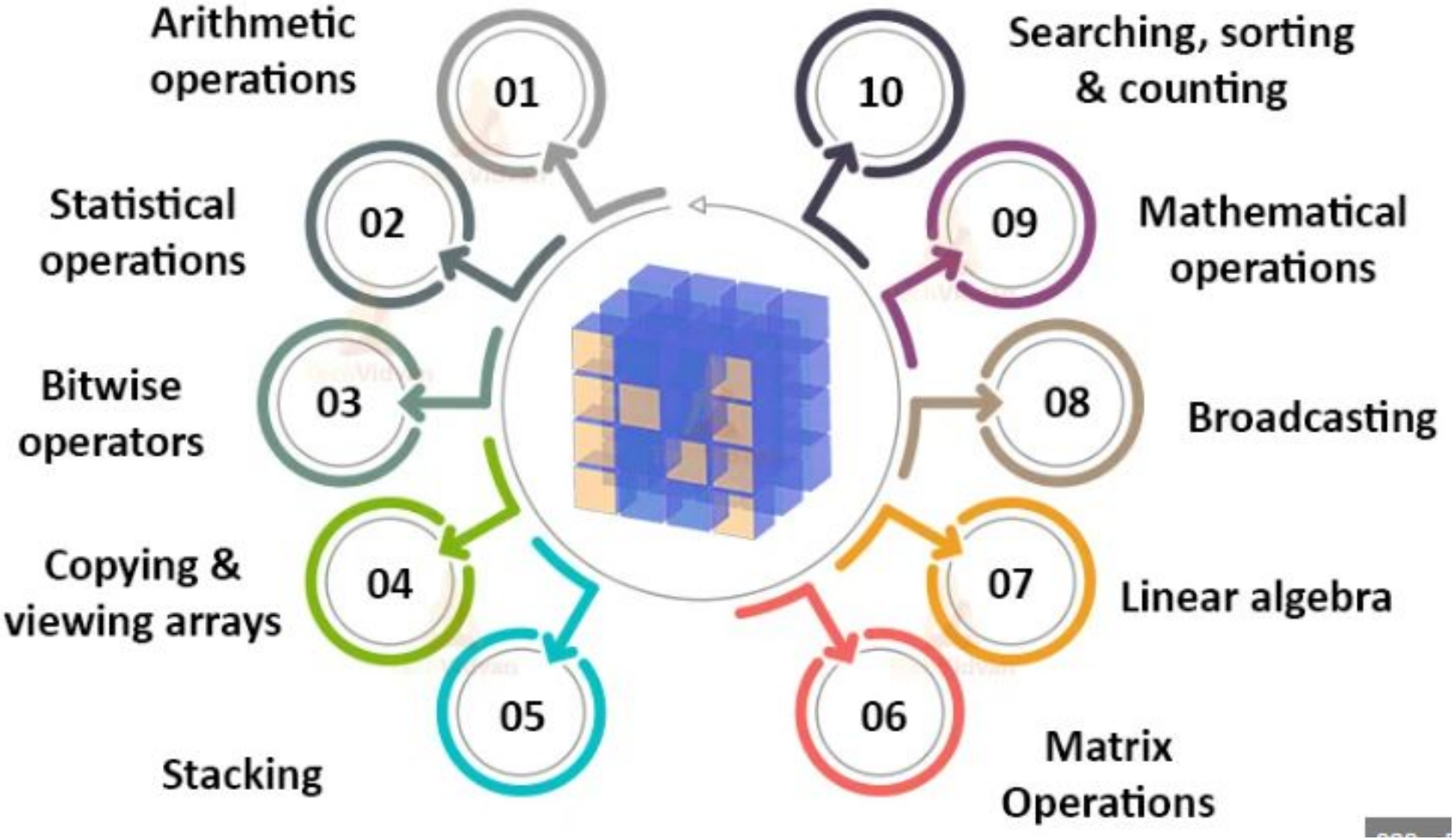

```
# Merging DataFrames using the 'UEN' column

# Basic merge (INNER JOIN)
merged_df = pd.merge(df1, df2, on='UEN')

# Left merge (keep all rows from df1)
left_merged = pd.merge(df1, df2, on='UEN', how='left')

# Right merge (keep all rows from df2)
right_merged = pd.merge(df1, df2, on='UEN', how='right')

# Outer merge (keep all rows from both)
outer_merged = pd.merge(df1, df2, on='UEN', how='outer')

# If 'UEN' column has different names in each DataFrame:
merged_df = pd.merge(df1, df2, left_on='UEN_1',
right_on='UEN_2')
```

JupyterLab time!

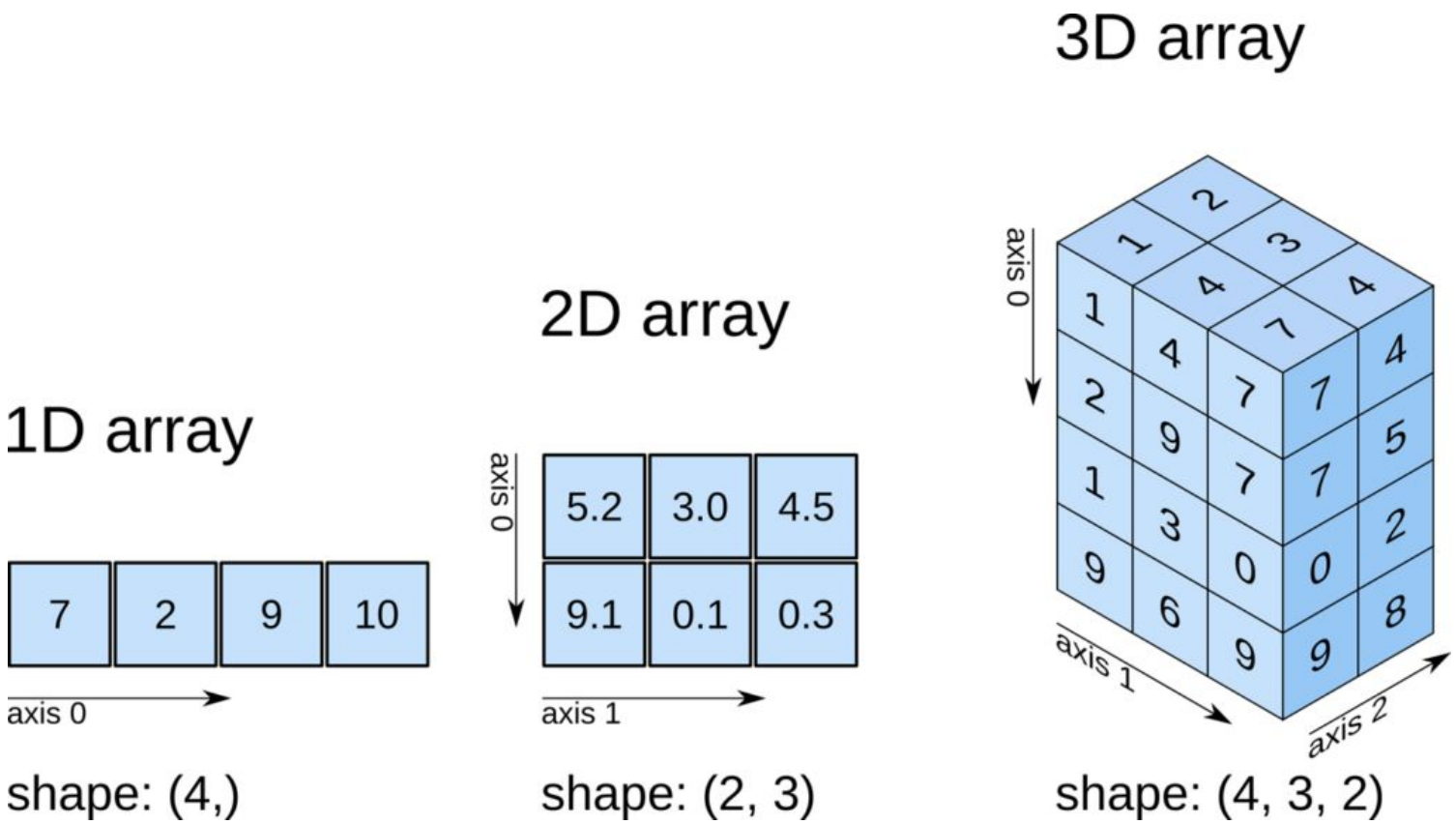
We will cover the following sections:

- Loading and saving data

Uses of NumPy



NumPy: Statistical Modelling



Here is a [good cheat sheet!]

Section IV: Data Visualization

Data Visualization Packages

We will cover two packages:

- matplotlib
- altair

Matplotlib

The most popular library for data visualization in Python

Cheat sheet 1

Cheat sheet 2

Altair

Altair website has lots of useful code for all sorts of different graphs!

More data visualization in course 2 and 3!

JupyterLab time!

We will cover the following sections:

- Introduction to Data Visualization

Then, we will do Day 1 Project!

Assignment Submission

Please put your name on the file:

Final_Assignment_[name].ipynb

Submit here: FOSVL > Assignments > 2025 > Run 4

Create requirements.txt

Requirement.txt documents exactly the packages you are using in your Python session. This is great for code reproducibility!

```
# After installing packages, type in terminal
pip freeze > requirements.txt

# Install from requirements.txt
pip install -r requirements.txt
```

SSG TRAQOM Quality Surveys

Data Analytics Begins With Me

(6 March 2025)

Please complete the Quality Survey via the QR code / URL by keying in:

1. The last four characters of your NRIC/FIN
2. Course Run ID: 1057204

https://ssgtraqom.qualtrics.com/jfe/form/SV_3K9i7rTJ9OLsauW?Q_CHL=qr

viktoria@nus.edu.sg

THANK YOU