



# Welcome to the **Co**Grammar Datasets and DataFrames

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.



## Data Science Session Housekeeping

---

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.  
**(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

## Data Science Session Housekeeping cont.

---

- For all **non-academic questions**, please submit a query: [www.hyperiondev.com/support](http://www.hyperiondev.com/support)
- Report a **safeguarding** incident: [www.hyperiondev.com/safeguardreporting](http://www.hyperiondev.com/safeguardreporting)
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

# Skills Bootcamp

## 8-Week Progression Overview

### Fulfil 4 Criteria to Graduation

#### ✓ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks

Guided Learning Hours (GLH):

Minimum of 15 hours

Task Completion: First four tasks

**Due Date: 24 March 2024**

#### ✓ Criterion 2: Mid-Course Progress

**60** Guided Learning Hours

Data Science - **13 tasks**

Software Engineering - **13 tasks**

Web Development - **13 tasks**

**Due Date: 28 April 2024**

# Skills Bootcamp Progression Overview

## ✓ Criterion 3: Course Progress

Completion: All mandatory tasks,  
including Build Your Brand and  
resubmissions by study period end  
Interview Invitation: Within 4 weeks  
post-course  
Guided Learning Hours: Minimum of  
112 hours by support end date  
(10.5 hours average, each week)

## ✓ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship  
Outcome: Document within 12  
weeks post-graduation  
Relevance: Progression to  
employment or related  
opportunity

# CoGrammar

## Datasets and DataFrames

April 2024



# Learning objectives

Learn how to **read** and **manipulate data** with **Pandas**

- ❖ Python packages for data science: NumPy and Pandas
- ❖ Datasets & DataFrames
- ❖ Read data from different sources, select specific columns and rows in a DataFrame
- ❖ Apply built-in DataFrame methods, grouping operations
- ❖ Jupyter Notebook

# Python packages for data science

## NumPy





# Python packages for data science: NumPy

❖ **NumPy** - stands for **Nu**merical **Py**thon

```
pip install numpy
```

❖ Numpy arrays differ from python lists

- Elements of same datatype
- Can handle arithmetic operations
- Preferred for larger chunks of data
- Requires proper modules to perform operations on them

❖ Array oriented programming, NumPy has smaller memory consumption, better runtime, and ease of data manipulation

# Using NumPy

```
import numpy as np
```

```
import numpy as np

#Define a list
distances = [1, 13.1, 26.2, 100]
print(type(distances))
print(distances)
#Output: <class 'list'>
#Output: [1, 13.1, 26.2, 100]

#Convert to numpy array
numpy_dist = np.array(distances)
print(type(numpy_dist))
print(numpy_dist)
#Output: <class 'numpy.ndarray'>
#Output: [ 1.  13.1 26.2 100.]
```

# Using NumPy

```
import numpy as np

#Define a List
distances = [1, 13.1, 26.2, 100]

#Convert to numpy array
numpy_dist = np.array(distances)

#Convert distances in miles to km
#Using numpy scalar mutliplication
conversion = numpy_dist * 1.60934
print(conversion)

#Output: [1.60934, 21.082354, 42.164708, 160.934]
```

```
#Using core Python

#Define a list
distances = [1, 13.1, 26.2, 100]

#Define an empty array to store km distances
conversion = []

#Using a for loop for conversion
for x in distances:
    conversion.append(x*1.60934)

print(conversion)

#Output: [1.60934, 21.082354, 42.164708, 160.934]
```

# Python packages for data science

## Pandas



# Pandas

- ❖ **Pandas**, built on NumPy, is a Python module that contains high-level data structures and tools designed for fast and easy data analysis.

```
pip install pandas
```

```
import pandas as pd
```

- ❖ Fundamental pandas data structures
  - Series (1-dimensional labelled array, can hold any data type)
  - DataFrame (2-dimensional)
  - Panel (pandas -> **panel data**)

	Name
0	Asha
1	Ben
2	Candice
3	Derek
4	Miriam
5	Seth
6	Zara

	Name	Age	Marks
0	Asha	12	96
1	Ben	12	92
2	Candice	13	94
3	Derek	12	96
4	Miriam	12	95
5	Seth	13	93
6	Zara	12	95

		DOB	Attending
	Gender	Dietary	Location
	Name	Age	Marks
0	Asha	12	96
1	Ben	12	92
2	Candice	13	94
3	Derek	12	96
4	Miriam	12	95
5	Seth	13	93
6	Zara	12	95



# Datasets





# Datasets

- ❖ A dataset is a **structured collection of information** relevant to a specific investigation or project
- ❖ In data science, they provide the **raw material** for **analysis** and **modeling**. Understanding different dataset formats ensures you can work with data from various sources (databases, online repositories, etc.).
- ❖ With the help of **Pandas DataFrames**, we can effortlessly manipulate data to suit our needs.

# DataFrames



# DataFrames

- ❖ A DataFrame is the way the **Pandas library** in Python represents **tabular data**. It's like a powerful *spreadsheet* within your code.
- ❖ **Rows**: Each row represents a **single observation** or data point (e.g., a person, a product, a transaction).
- ❖ **Columns**: Each column represents a **variable or feature** (e.g., height, price, date). Data within a column usually shares the **same data type** (numbers, text, etc.).

# Pandas DataFrame



# Pandas DataFrame

- ❖ The pandas' library documentation defines a DataFrame as a “two-dimensional, size-mutable, with labelled rows and columns.”

```
import pandas as pd
```

columns  
axis=1

column name

more columns to display

index label

index  
axis=0

missing values

data  
(values)

	color	director_name	num_critic_for_reviews	duration	...	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	...	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	...	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	...	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	...	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	...	12.0	7.1	NaN	0

Anatomy of a DataFrame



# Pandas DataFrame

- ❖ Pandas provides functions like `pd.read_csv()`, `pd.read_excel()`, `pd.read_sql()`, to bring your data directly into your coding environment as DataFrames.
- ❖ This is where you start turning your raw data into something easily workable.

```
import pandas as pd
```

```
# url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv'  
# df = pd.read_csv(url)  
from sklearn import datasets  
iris = datasets.load_iris()  
df = pd.DataFrame(iris.data, columns=iris.feature_names)
```



# Exploring datasets

- ❖ **df.head(), df.tail():** Peek at the top and bottom rows for initial understanding

```
df.head()
```

✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

# Exploring datasets

- ❖ **df.head(), df.tail()**: Peek at the top and bottom rows for initial understanding

```
df.tail()
```

✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

# Exploring datasets

- ❖ **df.shape:** Tells you the dimensions (rows, columns) of your data.

```
df.shape
```

```
✓ 0.0s
```

```
(150, 5)
```

# Exploring datasets

- ❖ **df.info():** Gives the **data types** of each column, and if columns have missing values

```
df.info()
✓ 0.0s
```

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 150 entries, 0 to 149  
Data columns (total 5 columns):

#	Column	Non-Null Count	Dtype
0	sepal length (cm)	150 non-null	float64
1	sepal width (cm)	150 non-null	float64
2	petal length (cm)	150 non-null	float64
3	petal width (cm)	150 non-null	float64
4	species	150 non-null	int64

dtypes: float64(4), int64(1)  
memory usage: 6.0 KB

# Exploring datasets

- ❖ **df.describe():** Quick summary statistics for numerical columns.

```
df.describe()
```

✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000



# What does a DataFrame represent in Pandas?

1. A two-dimensional array with labeled axes.
2. A single-dimensional array of data.
3. A database management system.
4. A type of Python function.





# What does a DataFrame represent in Pandas?

1. **A two-dimensional array with labeled axes.**
2. A single-dimensional array of data.
3. A database management system.
4. A type of Python function.



# Which method can be used to read a CSV file into a DataFrame?

1. `pd.to_csv()`
2. `pd.csv_reader()`
3. `pd.read_csv()`
4. `pd.open_csv()`



# Which method can be used to read a CSV file into a DataFrame?

1. `pd.to_csv()`
2. `pd.csv_reader()`
3. **`pd.read_csv()`**
4. `pd.open_csv()`

# What does the `df.describe()` method provide?

1. A list of all the columns in a DataFrame.
2. A count of missing values in each column.
3. Summary statistics for numerical columns.
4. The first five rows of the DataFrame.



# What does the `df.describe()` method provide?

1. A list of all the columns in a DataFrame.
2. A count of missing values in each column.
- 3. Summary statistics for numerical columns.**
4. The first five rows of the DataFrame.



# Manipulating Data





# Manipulating Data

- ❖ **Selecting Columns:** You often work with a **subset of features**.
- ❖ Using `df[['column1', 'column2']]` gets you only specific columns.

```
df.columns
```

```
✓ 0.0s
```

```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',  
      'petal width (cm)', 'species'],  
      dtype='object')
```

```
# Select specific columns
```

```
df_selected = df[['species', 'petal length (cm)', 'petal width (cm)']]
```

```
✓ 0.0s
```

# Manipulating Data

- ❖ **Filtering Rows:** Focus on specific subsets meeting certain conditions, e.g., `df[df['species'] == 'setosa']`

```
# Filter by flower species
```

```
df_setosa = df[df['species'] == 'setosa']
```

✓ 0.0s

# Manipulating Data

- ❖ **Creating New Columns:** Derived features, e.g., calculating area from length and width.

```
# Create a new calculated column
```

```
df['petal area (cm^2)'] = df['petal length (cm)'] * df['petal width (cm)']
```

```
✓ 0.0s
```

# Manipulating Data

- ❖ **Renaming/Dropping:** Improve clarity or get rid of unneeded data.

```
# Rename a column
```

```
df = df.rename(columns={'sepal length (cm)': 'sepal_len'})
```

```
✓ 0.0s
```

- ❖ Data manipulation gives you a **highly customized DataFrame** focused on your exact analysis needs.

# Built-in Methods

- ❖ Pandas offers a toolbox of functions for calculations:
  - **mean()** - Computes the mean for each column.
  - **min()** - Computes the minimum for each column.
  - **max()** - Computes the maximum for each column.
  - **std()** - Computes the standard deviation for each column.
  - **var()** - Computes the variance for each column.
  - **unique()** - Computes the number of unique values in each column.
- ❖ This is the start of understanding the characteristics of your data.



# Grouping and Aggregation

- ❖ `df.groupby()`: Divide your data **based on categories** in a column (e.g., group by species).

```
print(df['petal area (cm^2)'].mean())  
print(df['species'].nunique())  
print(df.groupby('species')['petal length (cm)'].std())
```

✓ 0.0s

5.794066666666667

3

species

0 0.173664

1 0.469911

2 0.551895

Name: petal length (cm), dtype: float64

# Grouping and Aggregation

- ❖ **.agg()**: Apply calculations within each group (e.g., average length, maximum width).

```
df.groupby('species').agg(  
    mean_petal_length=('petal length (cm)', 'mean'),  
    max_sepal_width=('sepal width (cm)', 'max')  
)
```

✓ 0.0s

	mean_petal_length	max_sepal_width
species		
0	1.462	4.4
1	4.260	3.4
2	5.552	3.8

# 1. How can you select a subset of columns from a DataFrame?


1. `df['column1', 'column2']`
2. `df[['column1', 'column2']]`
3. `df(column1, column2)`
4. `df.columns['column1', 'column2']`

# 1. How can you select a subset of columns from a DataFrame?

1. `df['column1', 'column2']`
2. **`df[['column1', 'column2']]`**
3. `df(column1, column2)`
4. `df.columns['column1', 'column2']`



# What does the `groupby()` method do?

1. Sorts the DataFrame based on specified columns
  2. Combines several columns into a new one
  3. Splits the DataFrame into groups based on some criteria
  4. Applies a function to each row in a DataFrame
- 



# What does the `groupby()` method do?

1. Sorts the DataFrame based on specified columns
2. Combines several columns into a new one
- 3. Splits the DataFrame into groups based on some criteria**
4. Applies a function to each row in a DataFrame



## Which method is used to get a quick overview of a DataFrame's structure?

1. `df.overview()`
2. `df.describe()`
3. `df.summary()`
4. `df.info()`



# Which method is used to get a quick overview of a DataFrame's structure?

1. `df.overview()`
2. `df.describe()`
3. `df.summary()`
4. **`df.info()`**



# Jupyter Notebook



# Jupyter Notebook

- ❖ An **interactive environment** perfect for **data science work**. They let you combine **code**, the **results of the code** (output), and **explanatory text** (like in a scientific report).
- ❖ This fosters **clear data exploration** and **storytelling**, all in one place

## Installation

```
pip install jupyter
```

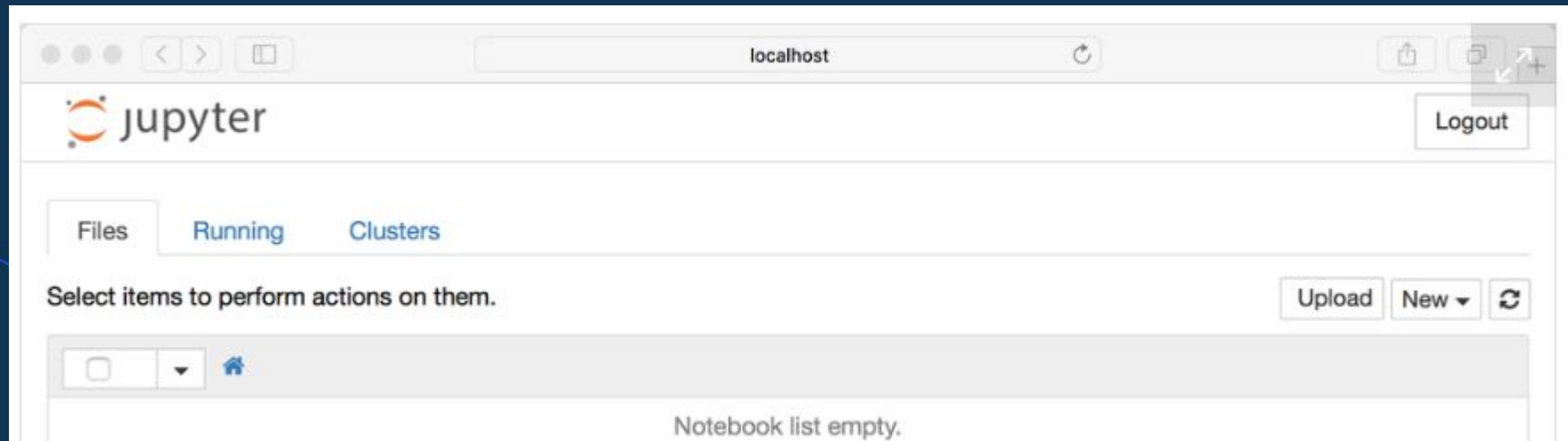
## Running

```
jupyter notebook
```

```
python -m notebook
```

# Jupyter Notebook

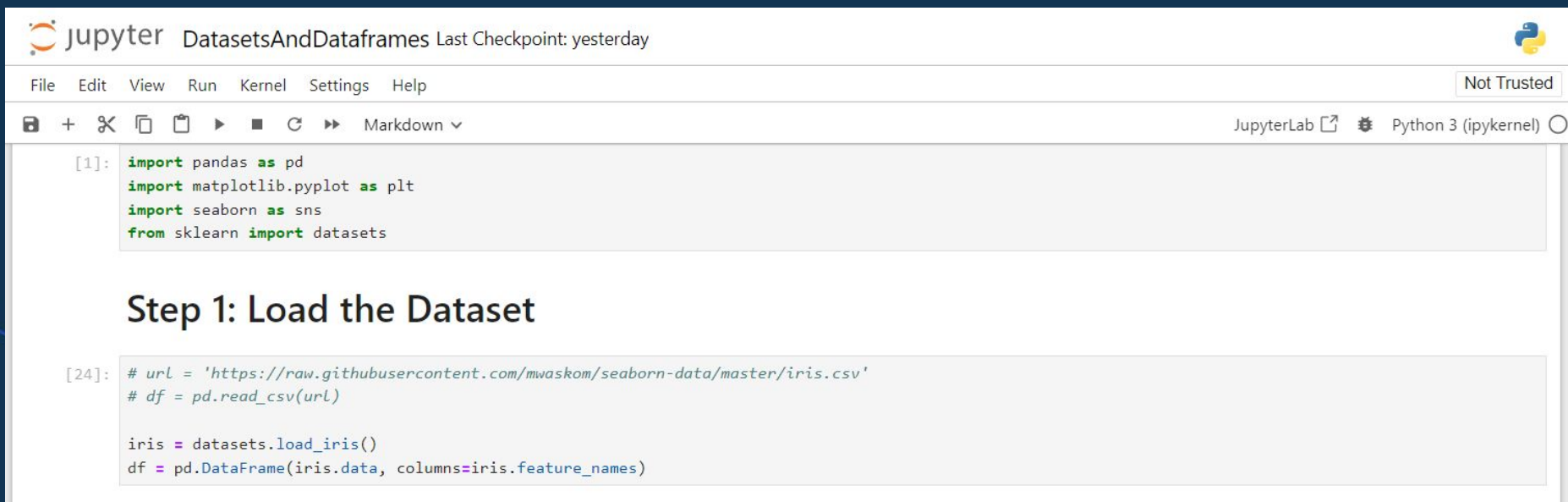
Starting the Jupyter Notebook Server, default browser goes to the URL <http://localhost:8888/tree>





# Jupyter Notebook

Creating, naming notebook, code, markdown



The screenshot displays a Jupyter Notebook window titled "DatasetsAndDataframes" with a "Last Checkpoint: yesterday" status. The interface includes a top menu bar with "File", "Edit", "View", "Run", "Kernel", "Settings", and "Help". A "Not Trusted" warning is visible in the top right. Below the menu is a toolbar with icons for saving, adding, deleting, copying, pasting, running, and other actions. The notebook content is divided into two cells. The first cell, labeled "[1]:", contains Python code to import necessary libraries: pandas as pd, matplotlib.pyplot as plt, seaborn as sns, and sklearn datasets. The second cell, labeled "[24]:", contains code to load the Iris dataset from a GitHub URL and create a DataFrame. The code in the second cell is as follows:

```
[24]: # url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv'
# df = pd.read_csv(url)

iris = datasets.load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
```

# Next Lecture

## Data visualisation



# Data Visualisation

- ❖ There is a whole lecture dedicated to this, libraries like **Matplotlib** and **Seaborn** make visually exploring data easy.
- ❖ **Plots (scatter plots, histograms, etc.)** rapidly uncover relationships and distributions that are less clear from tables of numbers.
- ❖ Some good examples are in the **Seaborn Gallery**.

# Questions and Answers





# Thank you for attending



Department  
for Education

CoGrammar

