DATA SCIENCE WITH PYTHON

A hands-on guide to turning raw data into clear, useful findings

LEARNSAPPEAL | ORIGINAL COURSE

Ask a good question. Explore the evidence. Explain what you found.

Use Python tools to load, clean, analyze, and visualize data, then communicate conclusions with care.

Before you start

Basic Python knowledge is useful: variables, lists, functions, and reading simple code. The examples use pandas, Matplotlib, and scikit-learn. You can read the ideas first, then install the libraries to run the examples.

Course map: Data and Python foundations (Modules 1-4), followed by analysis, communication, and a complete project (Modules 5-12). Includes code examples, self-checks with answers, projects, and a glossary.

How to use this course

Work through the steps in order: understand the question, inspect the source, tidy the data, explore patterns, and share the result. Keep the original data unchanged. Save each transformation in code so that you can reproduce it later.

By the end, you can: load common data formats; inspect and clean tables; summarize and visualize patterns; join related datasets; use statistics carefully; build a simple predictive workflow; and communicate findings with limitations.

Course map

- 1. The data science workflow
- 2. Set up a Python workspace
- 3. Load and inspect tabular data
- 4. Clean data and handle missing values
- 5. Summarize with pandas
- 6. Visualize patterns
- 7. Compare groups with statistics
- 8. Join tables and work with dates
- 9. Build a predictive baseline
- 10. Make analysis reproducible
- 11. Communicate results responsibly
- 12. Capstone: course catalog insights

DATA AND PYTHON FOUNDATIONS

1. The data science workflow

Data science connects a question to evidence. Define the decision or curiosity first; find out how data was collected; explore its structure; clean and transform it; analyze patterns; communicate what the evidence supports. Results can describe what happened, help explain why, predict what might happen, or support a decision. These are different goals and need different methods.

Example

```
question = 'Which course topics have the most enrollments?'
# Plan: inspect enrollment records, summarize by topic,
# check date coverage, and explain what is missing
```

Check your understanding

- Why define the question before choosing a chart?
- Name one thing to learn about data collection.
- Does a pattern automatically prove cause and effect?

Suggested answers

1. The question determines which data and analysis are relevant.
2. Who or what was included, when it was collected, or how fields were defined.
3. No. A relationship may have other explanations.

2. Set up a Python workspace

A project folder keeps data, code, and outputs organized. A virtual environment isolates the packages used by one project. Jupyter notebooks are useful for exploration because code and notes sit together; Python scripts are useful for repeatable tasks. Keep private data out of shared folders and code repositories.

Example

```
# In a terminal, create and activate a virtual environment
# Then install the course libraries:
python -m pip install pandas matplotlib seaborn scikit-learn jupyter

import pandas as pd
print(pd.__version__)
```

Check your understanding

- Why use a virtual environment?
- When are notebooks useful?
- What should you avoid committing to a public repository?

Suggested answers

1. To keep project packages separate from other work.
2. Exploration that benefits from code, notes, and outputs together.
3. Secrets, credentials, or private data.

3. Load and inspect tabular data

A DataFrame is a table with labeled rows and columns. Read a CSV with pandas, then inspect a few rows, column types, dimensions, and summary information. Check units, date ranges, duplicates, and category values before analysis. A column named “age” is not meaningful until you know whose age, how it was measured, and whether missing values exist.

Example

Examples use pandas, Matplotlib, or scikit-learn. Install the libraries in your project environment before running them.

```
import pandas as pd

df = pd.read_csv('enrollments.csv')
print(df.head())
print(df.shape)
print(df.dtypes)
print(df.isna().sum())
```

Check your understanding

- What does head() show?
- What does shape tell you?
- Why inspect dtypes?

Suggested answers

1. A preview of the first rows.
2. The number of rows and columns.
3. Types affect how values can be analyzed and may reveal import issues.

4. Clean data and handle missing values

Cleaning includes correcting types, removing exact duplicate records when appropriate, standardizing category spelling, and deciding how to handle missing values. Do not delete rows automatically: missingness may be meaningful. Record each decision and compare important summaries before and after cleaning. Never replace unknown values with a convenient guess without justification.

Example

Examples use pandas, Matplotlib, or scikit-learn. Install the libraries in your project environment before running them.

```
df['enrolled_at'] = pd.to_datetime(
    df['enrolled_at'], errors='coerce'
)
df['topic'] = df['topic'].str.strip().str.lower()
df = df.drop_duplicates()
print(df['topic'].value_counts(dropna=False))
```

Check your understanding

- Why use errors="coerce" when parsing dates?
- What can inconsistent category spelling do?
- Why document cleaning decisions?

Suggested answers

1. Invalid values become missing so they can be inspected.
2. It can split one real category into several apparent groups.
3. So the analysis is reproducible and assumptions are visible.

ANALYZE, EXPLAIN, AND APPLY

5. Summarize with pandas

Descriptive statistics summarize the data you have. Use counts for categories, medians and quartiles for skewed numeric values, and groupby to compare subgroups. A mean can be pulled by extreme values. Always include the denominator: 20 events out of 30 is different from 20 out of 20,000.

Example

```
summary = (
    df.groupby('topic')['learner_id']
      .nunique()
      .sort_values(ascending=False)
)
print(summary)
print(df['days_to_complete'].describe())
```

Check your understanding

- What does nunique() count?
- When might a median describe a typical value better than a mean?
- Why report a denominator?

Suggested answers

1. The number of distinct values.
2. When a distribution is skewed or has extreme values.
3. To show the scale and context behind a rate or count.

6. Visualize patterns

A chart should answer a question clearly. Bar charts compare categories; histograms show numeric distributions; line charts show ordered time trends; scatter plots show relationships between two numeric measures. Label axes and units, choose scales carefully, and avoid visual tricks that exaggerate small differences. A plot is a view of the data, not a substitute for checking it.

Example

Examples use pandas, Matplotlib, or scikit-learn. Install the libraries in your project environment before running them.

```
import matplotlib.pyplot as plt

summary.plot(kind='bar', color='#087E8B')
plt.title('Unique learners by topic')
plt.xlabel('Course topic')
plt.ylabel('Unique learners')
plt.tight_layout()
plt.show()
```

Check your understanding

- Which chart could show a time trend?
- Why label units on an axis?
- What can a truncated axis do?

Suggested answers

1. A line chart.
2. So viewers can interpret the magnitude correctly.
3. It can make small differences look much larger.

7. Compare groups with statistics

A sample is only part of the population you care about. Sampling choices affect what conclusions are reasonable. A confidence interval describes uncertainty under a statistical model; it does not remove bias in the data collection. Statistical significance is not the same as practical importance. Start with a plot and effect size, then choose an appropriate test if the question requires one.

Example

```
# Compare group summaries before choosing a test
group_stats = df.groupby('topic')['days_to_complete'].agg(
    ['count', 'median', 'mean']
)
print(group_stats)
```

Check your understanding

- What can a confidence interval communicate?
- Can significance correct a biased sample?
- What should accompany a p-value?

Suggested answers

1. Uncertainty around an estimate under stated assumptions.
2. No. A test cannot repair how the data was collected.
3. Effect size, sample context, assumptions, and practical relevance.

8. Join tables and work with dates

Real analyses often combine multiple tables. A merge joins rows based on keys; check whether keys are unique and whether the join multiplies rows unexpectedly. Dates can be grouped by month or weekday, but time zones and missing dates matter. After a join, compare row counts and inspect unmatched keys.

Example

```
courses = pd.read_csv('courses.csv')
joined = df.merge(
    courses[['course_id', 'course_title']],
    on='course_id', how='left', validate='many_to_one'
)
print(len(df), len(joined))
print(joined['course_title'].isna().sum())
```

Check your understanding

- What does a left join preserve?
- Why check row counts after a merge?
- What does validate help detect?

Suggested answers

1. All rows from the left table, whether or not a match exists.
2. Unexpected duplicate matches can multiply records.
3. Whether the relationship between join keys matches expectations.

9. Build a predictive baseline

Prediction uses historical examples to estimate an outcome for new cases. Split examples into training and test sets before fitting transformations or models. Start with a simple baseline and choose metrics based on the decision. A model's score is not proof that it will work for future people, dates, or settings.

Example

Examples use pandas, Matplotlib, or scikit-learn. Install the libraries in your project environment before running them.

```
# Example workflow outline
from sklearn.model_selection import train_test_split
from sklearn.dummy import DummyClassifier

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=12, stratify=y
)
baseline = DummyClassifier(strategy='most_frequent')
baseline.fit(X_train, y_train)
print(baseline.score(X_test, y_test))
```

Check your understanding

- Why use a simple baseline?
- What does the test set estimate?
- Why might random splitting be inappropriate for time data?

Suggested answers

1. It provides a reference to beat and can expose a weak approach.
2. Performance on examples not used to fit the model.
3. It can let future information influence the apparent prediction of the past.

10. Make analysis reproducible

A reproducible project records where data came from, which packages and versions are used, how to run the code, and what outputs to expect. Use clear file paths, named functions, and a fixed random seed where randomness affects results. Separate raw data from derived outputs. A notebook should run from top to bottom without hidden state.

Example

```
from pathlib import Path
DATA = Path('data') / 'enrollments.csv'

def load_data(path):
    return pd.read_csv(path)

df = load_data(DATA)
print(f'Loaded {len(df)} rows from {DATA}')
```

Check your understanding

- Why separate raw data from outputs?

- What does a random seed help with?
- What is notebook hidden state?

Suggested answers

1. To preserve the source and make transformations traceable.
2. It can make randomized steps repeatable.
3. Results that depend on cells or variables left over from earlier runs.

11. Communicate results responsibly

A useful report explains the question, data source, time period, method, findings, and limits. Protect personal information and only use data with appropriate permission. Avoid presenting a correlation as causation. Check whether conclusions hold across relevant groups and note who may be missing. Use clear charts and plain language so a reader can question the reasoning.

Example

```
# A clear finding includes
# - the observed result and its time period
# - the population and data source
# - uncertainty and important exclusions
# - a practical interpretation, not an unsupported cause
```

Check your understanding

- Why mention exclusions?
- What is a correlation?
- Why check results across groups?

Suggested answers

1. Readers need to know who or what the result may not represent.
2. A measured relationship between variables.
3. Aggregate results can hide different experiences or systematic gaps.

12. Capstone: course catalog insights

Use a small, approved dataset of course enrollments and course metadata. Ask a focused question such as how enrollment changes across topics and months. Inspect and clean the tables, summarize meaningful measures, create two readable charts, and write a short report. Include the data period, definitions, caveats, and one recommendation that is proportional to the evidence.

Example

```
# Capstone outline
# 1. define the question and key measures
# 2. inspect source tables and data dictionary
# 3. clean and validate joins
# 4. summarize by topic and month
# 5. chart findings and check anomalies
# 6. write conclusions, limits, and next questions
```

Check your understanding

- Why begin with a focused question?
- Name one check after joining tables.
- What belongs in the report besides charts?

Suggested answers

1. It keeps the analysis relevant and limits unnecessary work.
2. Row counts, unmatched keys, or duplicate matches.
3. The question, data source, definitions, findings, and limitations.

Project workshop

1. Learning activity report

Create a report from a small, privacy-safe dataset. Summarize activity by week and topic, show counts and rates with denominators, and describe missing periods or fields.

2. Public dataset exploration

Choose an openly licensed dataset with a data dictionary. Ask one question, inspect data quality, create a distribution chart and a comparison chart, and document what the data cannot tell you.

3. Reproducible analysis

Turn a notebook exploration into a clean project with reusable functions, a README, clear input and output folders, and a fresh run from top to bottom.

Project checklist
Is the question specific? Are the source and time period clear? Can the steps be rerun? Are chart labels and units present? Are uncertainty, privacy, and limits explained?

Glossary

Term	Meaning
DataFrame	A labeled two-dimensional table provided by pandas.
Feature	An input variable used in analysis or prediction.
Join	Combining tables using one or more matching keys.
Median	The middle value after sorting observations.
Missing value	A value that is absent, unknown, or not recorded.
Population	The complete group a question is about.
Sample	The subset of a population represented in collected data.
Target	The outcome a supervised model is asked to predict.
Visualization	A chart or graphic used to examine or communicate data.
Data leakage	Information entering analysis or training that would not be available at the intended decision time.
Reproducibility	The ability to rerun documented steps and obtain consistent results.
Correlation	A measure of how variables vary together; it does not alone show causation.

Further study

Official documentation is a useful companion as Python data tools evolve. Check versions and examples when applying these workflows to current projects.

- pandas. *Getting started.* https://pandas.pydata.org/docs/getting_started/
- Matplotlib. *Quick start guide.* https://matplotlib.org/stable/users/explain/quick_start.html
- scikit-learn. *User Guide.* https://scikit-learn.org/stable/user_guide.html
- Python Software Foundation. *The Python Tutorial.* https://docs.python.org/3/tutorial/

Course completion checklist
I can turn a question into a data workflow; load and inspect tables; clean and join data; summarize and visualize patterns; interpret statistics with care; build a simple predictive baseline; reproduce my analysis; and communicate findings, limits, and responsible next steps.