Pandas Basics: Tables in Python

A pandas DataFrame is a table you can work with in Python. Read a small table, peek at its top and bottom, pick a column, filter rows, and describe the numbers.

Introduction

So far you have worked with data as lists and dictionaries. That works for ten rows, but it gets painful at a thousand. **Pandas** is the Python library that data scientists use for tables. Its main object is the **DataFrame** — think of it as a spreadsheet inside Python, with named columns and numbered rows.

With a DataFrame you can do in one line what takes a loop in plain Python: show the first five rows, pick one column, keep only the rows that pass a test, or summarise a whole column. That is why almost every data science tutorial starts with pandas. It is not the only tool, but it is the standard one, and learning its few basic moves unlocks everything else.

This lesson uses small tables you can type yourself. Every pandas example below needs `pip install pandas` first — install it once, then the code runs as shown.

One point trips up almost every beginner, so let us settle it early. `df["marks"]` gives you one column — a Series. `df[["marks"]]` (with double brackets) gives you a table with one column — a DataFrame. They look almost the same when printed, but they behave differently: you can ask a DataFrame for `describe()`, while a Series has its own methods like `value_counts()`. When something behaves unexpectedly, check first whether you are holding a Series or a DataFrame. Most pandas confusion is really this one distinction.

(Roman Urdu: Pandas Python ki woh library hai jo table ko asaan bana deti hai — DataFrame matlab Python ke andar ek spreadsheet, jis mein column ke naam aur row ke number hote hain.)

Key points

- **A DataFrame is a table.** Columns have names, rows have numbers, and each column holds one type of value.
- **`pd.DataFrame()`** builds a table from a dictionary of lists, or from a list of dictionaries.
- **`df.head()`** shows the first 5 rows; **`df.tail()`** shows the last 5. Use them to check your table the moment you load it.
- **`df["column"]`** picks one column. The result is a **Series** — a single column of values.
- **Filtering keeps rows that pass a test:** `df[df["marks"] >= 80]` keeps only rows where marks are 80 or more.
- **`df.describe()`** summarises every number column: count, average, smallest, largest, and more.
- **`df.shape`** tells you the size: `(rows, columns)`. Check it after every cleaning step.

- **Plain Python can do the same jobs on small data** — pandas just does them faster and with less code as tables grow.
- `df["subject"].value_counts()` counts text values: how many Maths rows, how many English rows. It is the fastest way to understand a category column.
- `df.sort_values("marks")` sorts rows by a column, smallest first; add `ascending=False` for biggest first.
- `df.groupby("subject")["marks"].mean()` gives the average per group — one line that answers "which subject is stronger?", as lesson DS.7 uses it.

Examples

Example 1 — Build a table and peek at it

This example needs `pip install pandas` first:

```
import pandas as pd

df = pd.DataFrame({
    "name": ["Ali", "Sara", "Bilal", "Ayesha", "Usman"],
    "subject": ["Maths", "Maths", "English", "Maths", "English"],
    "marks": [85, 92, 74, 88, 61],
})

print(df.head(3))   # first 3 rows only
print(df.tail(2))   # last 2 rows only
print(df.shape)     # (5, 3): 5 rows, 3 columns
```

What this means: `head()` and `tail()` are your first check on any new table — do the columns look right, are the values what you expected? `shape` confirms the size before you start filtering.

Example 2 — Pick a column, filter rows, describe numbers

This example needs `pip install pandas` first:

```
import pandas as pd

df = pd.DataFrame({
    "name": ["Ali", "Sara", "Bilal", "Ayesha", "Usman"],
    "marks": [85, 92, 74, 88, 61],
})

print(df["marks"])                  # one column only
print(df["marks"].mean())           # average marks
top = df[df["marks"] >= 85]         # rows where marks are 85 or more
print(top)
print(df.describe())                # summary of the marks column
```

What this means: the filter `df["marks"] >= 85` keeps Ali, Sara and Ayesha, and drops the rest — the original table is unchanged, the result is a new, smaller table. `describe()` gives you count, mean, minimum and maximum at a glance, which is the fastest way to sense-check a column.

Example 3 — The same jobs in plain Python, no install

```
students = [
    {"name": "Ali", "marks": 85},
    {"name": "Sara", "marks": 92},
    {"name": "Bilal", "marks": 74},
    {"name": "Ayesha", "marks": 88},
    {"name": "Usman", "marks": 61},
]

marks = [s["marks"] for s in students]
print("Average:", sum(marks) / len(marks))
top = [s for s in students if s["marks"] >= 85]
print("Top students:", top)
print("Highest:", max(marks), "Lowest:", min(marks))
```

What this means: everything pandas did above, plain Python can also do — with a little more code. Learn the idea here first, then let pandas save you time when tables get big. The answers match: average 80.0, top students Ali, Sara and Ayesha.

Example 4 — Average marks per subject, sorted

This example needs `pip install pandas` first:

```
import pandas as pd

df = pd.DataFrame({
    "name": ["Ali", "Sara", "Bilal", "Ayesha", "Usman"],
    "subject": ["Maths", "Maths", "English", "Maths", "English"],
    "marks": [85, 92, 74, 88, 61],
})

averages = df.groupby("subject")["marks"].mean()
print(averages)
print(averages.sort_values(ascending=False))
```

What this means: Maths averages (85 + 92 + 88) / 3 = 88.33, English averages (74 + 61) / 2 = 67.5. Sorting puts the stronger subject first, so the answer to "which subject is stronger?" is visible at the top of the output, not buried in a table.

Example 5 — Adding a pass/fail column and counting it

This example needs `pip install pandas` first:

```
import pandas as pd

df = pd.DataFrame({
    "name": ["Ali", "Sara", "Bilal", "Ayesha", "Usman"],
    "marks": [85, 92, 74, 88, 61],
})

df["passed"] = df["marks"] >= 70
print(df)
print(df["passed"].value_counts())
```

What this means: a whole new column appears from one comparison — True for Ali, Sara, Bilal and Ayesha, False for Usman. `value_counts()` then tells you the result in one line: 4 passed, 1 failed. Making new columns from old ones is how raw tables turn into answers.

Example 6 — Real-world case: a school office's monthly report

Every month, a school office turns an attendance-and-marks register into a report for parents: average per class, per subject, and a list of students below 70 who need extra help. By hand, that report takes a day and invites copying mistakes. With pandas it is the moves from this lesson: load the table, check `shape` and `head()`, group by subject for averages, filter for marks below 70, and print the result. The office still checks the numbers by eye before sending them — pandas speeds up the work, but the human still owns the answer.

(Roman Urdu: School ki monthly report bhi inhi chand pandas moves se banti hai — group karo, filter karo, phir aankh se check karo.)

Quick recap

- A DataFrame is a table in Python; a Series is one column of it.
- `head()` and `tail()` peek at the table; `shape` gives its size.
- `df["column"]` picks a column; a true/false test filters rows.
- `describe()` summarises all number columns in one call.
- The same ideas work in plain Python — pandas just scales them up.

Common mistakes

1. **Forgetting the quotes around a column name** — `df[marks]` fails, because `marks` alone is not a column name. Fix: always write `df["marks"]` with quotes.
2. **Filtering but forgetting to save the result** — `df[df["marks"] >= 85]` on its own changes nothing; the filtered table is only shown, then lost. Fix: save it: `top = df[df["marks"] >= 85]`.
3. **Calling `describe()` and expecting text columns in it** — `describe()` summarises number columns by default and skips text. Fix: that is normal behaviour, not an error; use `df["subject"].value_counts()` to count text values instead.
4. **Using `df.marks` instead of `df["marks"]`** — dot access works until a column name has a space ("total marks") or matches a built-in word, and then it fails mysteriously. Fix: always use square brackets with quotes.
5. **Forgetting that filtering keeps the old row numbers** — after filtering, row numbers may jump (0, 1, 4) instead of running 0, 1, 2. That is normal and harmless, but it confuses beginners who expect a tidy new table. Fix: if tidy numbers matter, add `.reset_index(drop=True)` to the filtered table.

Practice

1. Build a DataFrame of 6 students with names and marks, then print `head(2)` and `shape`.
2. Filter it to students with marks below 70, and save the result in a new variable. Print it.
3. Run `describe()` on your table and write down, in one line each, the average, the highest mark and the lowest mark.
4. Rebuild the same table as a plain-Python list of dictionaries and compute the average without pandas. Check both answers match.

1. Build the 5-student table from Example 4 and use `value_counts()` on the subject column. Which subject has more students?
2. Add a `passed` column like Example 5, then filter to only the students who failed, and print just their names and marks.
3. Sort your table by marks, biggest first, and write down the top two students. Then sort smallest first and check the bottom student matches your expectation.

Practice quiz

1. What is a pandas DataFrame?

- A chart
- A table with rows and columns that you can filter and summarise in Python
- A database server
- A file format

Answer: A table with rows and columns that you can filter and summarise in Python

2. What does df.head() show?

- The last rows
- The first few rows of the table
- The column names only
- A chart

Answer: The first few rows of the table

3. How do you keep only some rows of a DataFrame?

- Delete the file
- Give a condition — only rows that match it stay
- Sort the table
- Rename the columns

Answer: Give a condition — only rows that match it stay

Watch the video lesson: https://youtu.be/_z_jBZn3QDM

Read online: https://scopetreasure.com/notes/ds-pandas-basics · More free tools and notes at https://scopetreasure.com