Python for Data Analytics

Module 2: Data structures and descriptive statistics

DeepLearning.AI

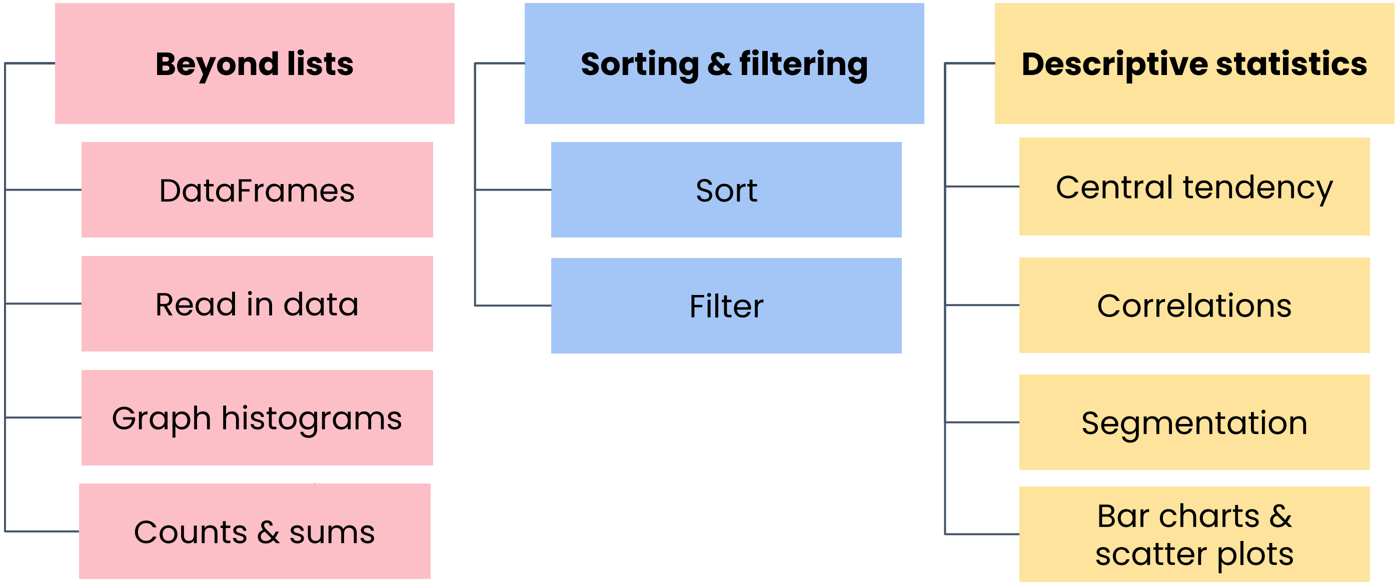
Data structures and descriptive statistics

Module 2 introduction

DeepLearning.AI

Module 2 outline

Coding habits



Beyond lists

- DataFrames
- Read in data
- Graph histograms
- Counts & sums

Sorting & filtering

- Sort
- Filter

Descriptive statistics

- Central tendency
- Correlations
- Segmentation
- Bar charts & scatter plots

DeepLearning.AI

Sean Barnes

Data structures and descriptive statistics

Beyond lists

DeepLearning.AI

Data structures

- How data is arranged and organized

Lists

✅ Quick analysis

✅ Collection of items

✅ Action on items

⛔ One dimensional

⛔ Too flexible

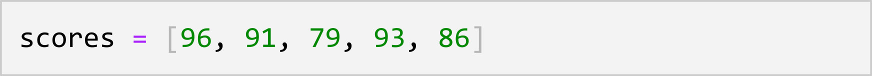
⛔ Inefficient at scale

Other data structures

✅ Two-dimensional

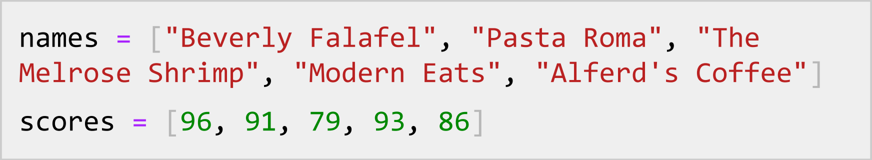
✅ Store multiple columns of data together

"What score did The Melrose Shrimp have?"



```
scores = [96, 91, 79, 93, 86]
```

❌ Missing key information



```
names = ["Beverly Falafel", "Pasta Roma", "The
Melrose Shrimp", "Modern Eats", "Alferd's Coffee"]
scores = [96, 91, 79, 93, 86]
```

❌ This approach is clunky

❌ No inherent organization of rows and columns

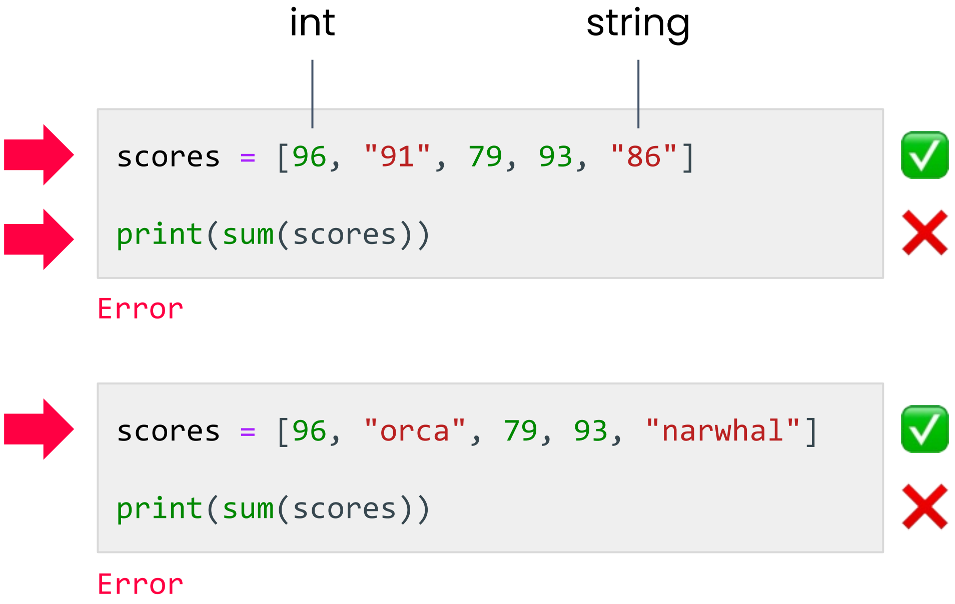
Flexibility

✅ Can store multiple data types

⛔ Doesn’t have a built in way to add **`int`** and **`string`**

⛔ Can hinder analysis if you introduce different types

- Each column should contain one type:
  - All ints
  - All strings
- Select a data structure that prevents introducing multiple types into a column



```
scores = [96, "91", 79, 93, "86"]

print(sum(scores))
```

```
Error
```

```
scores = [96, "orca", 79, 93, "narwhal"]

print(sum(scores))
```

```
Error
```

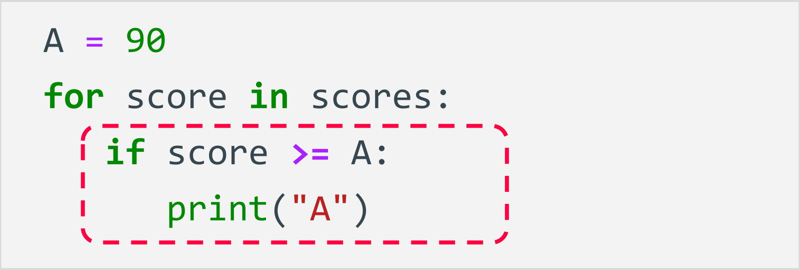
Efficiency

Lists

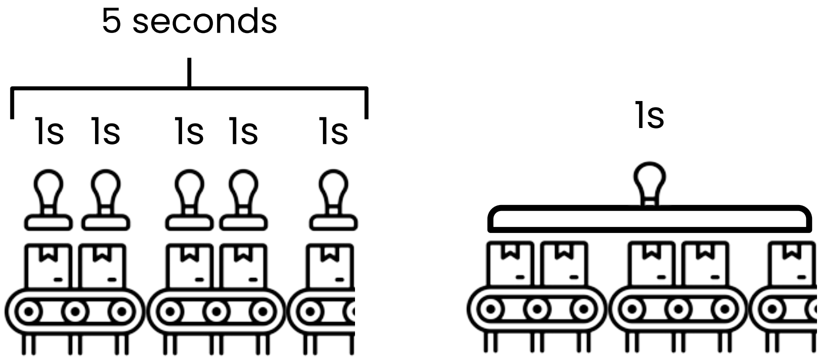
- Introduce inefficiencies for very large datasets (i.e. data at scale)
- **for** loops are not the most efficient option at scale

Specialized data structures

- Make this type of operation much more efficient
- **Examples**: Dataframes, Series
- Perform the same operation on the entire data structure at once



```
A = 90
for score in scores:
    if score >= A:
        print("A")
```



Vectorization: Performing an operation on the entire data structure at once

Data structures and descriptive statistics

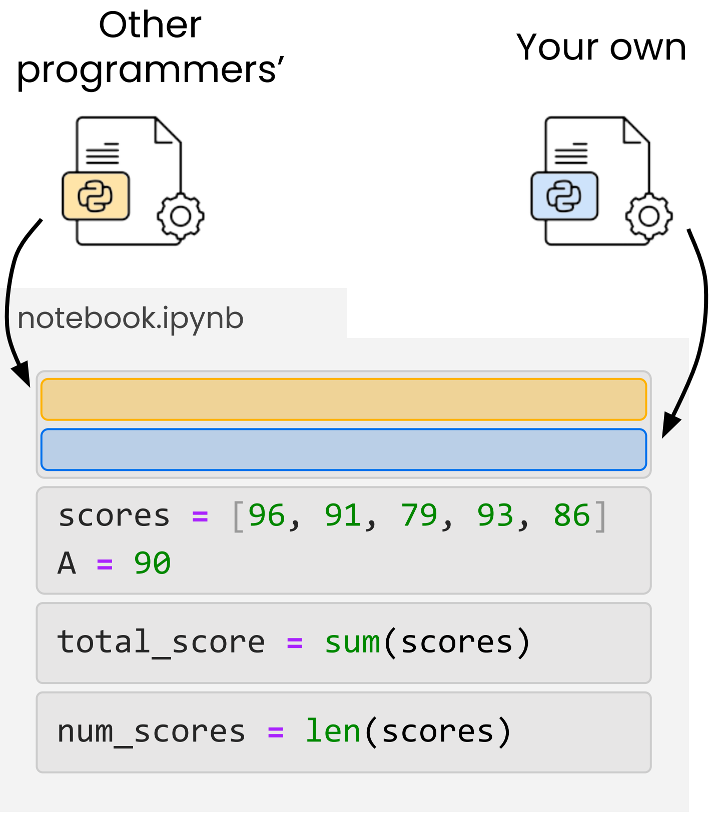
Importing modules

DeepLearning.AI

Importing code

- Automatically have access to many built-in Python functions
- Borrow other code using:

```
import
```



```
scores = [96, 91, 79, 93, 86]
A = 90
```

```
total_score = sum(scores)
```

```
num_scores = len(scores)
```

Recap: Importing modules

A **module** is a file containing Python code that's already been written

module function_name

1. Specific functions:

```
from helper_functions import get_restaurant_list
get_restaurant_list()
```

2. All functions:

```
import helper_functions
helper_functions.get_restaurant_list()
```

```
import helper_functions as hf
hf.get_restaurant_list()
```

helper_functions.py

Data structures and descriptive statistics

Pandas

DeepLearning.AI

pandas

- Extremely popular data science module
- Provides powerful data structures and functionality

DataFrame

Fiscal Year	Library	Operating expenses
1996	Avon	516583
1996	Plainville	49690
2005	Eastford	471864

Series

0	81003
1	72072
2	82698
3	96435

- Efficiently manipulate, clean, and analyze large datasets

- Provides function to:
    - Easily handle missing data
    - Sort, filter, and pivot your data
    - Merge and join datasets
    - Group and aggregate data
    - Perform time-series analysis
- Supports importing data from various file formats like:
    - CSV
    - Excel
    - Databases

Data structures and descriptive statistics

Reading CSV files into Python

DeepLearning.AI

Scenario

You
Data Analyst

🎯 **Task**: Investigate characteristics of programmers

🏆 **Goal**: Report about new programmers:

- Demographics
- Descriptive statistics
- Visualizations

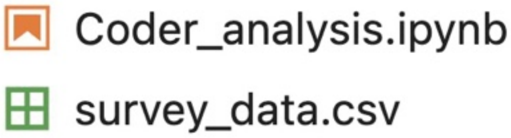
📊 **Data:** Online survey of new programmers

1. Load into Python
2. Explore it
3. Characterize different features

Recap: Reading CSV files into a dataframe

Goal: Get CSV data loaded into your notebook to analyze it with Python

1. Know the name of the file



Coder_analysis.ipynb

survey_data.csv

2. Use **`pd.read_csv`** and with one argument: name of file as a string

3. Assign that data into a variable

Creates a **`DataFrame`** with the csv data

```
import pandas as pd

df = pd.read_csv(“survey_data.csv”)
```

```
data = pd.read_csv(“survey_data.csv”)
```

```
survey_data = pd.read_csv(“survey_data.csv”)
```

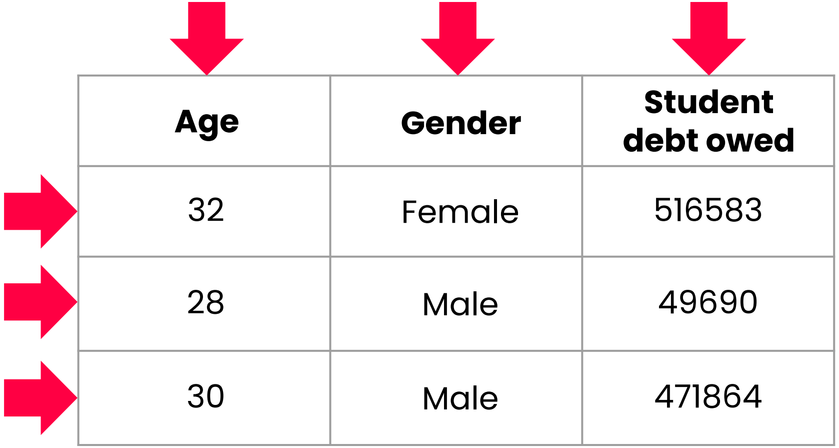
Data structures and descriptive statistics

DataFrames

DeepLearning.AI

DataFrame

- Represent a table or spreadsheet of data
- A type of data structure that represents data in **rows** and **columns**
- Similar to lists stored together, each representing a column



Age	Gender	Student debt owed
32	Female	516583
28	Male	49690
30	Male	471864

```
age = [32, 28, 30, ...]

gender = ["Female", "Male", "Male", ...]

student_debt_owed = [516583, 49690, 471864, ...]
```

Object type

- Store data that is more complex than numbers or true/false values.
- Pandas uses object type for texts to handle strings of any length
- When you see object:
    - Assume column contains text

Numbers:

✉ - same amount of space

Text:

✉ - “Great nail clippers!”

📦 - Several paragraphs

Data structures and descriptive statistics

Attributes and methods

Attributes and methods

Attributes:

`df.columns`

`df.dtypes`

- Do not need parentheses
- Something DataFrame **has**
- Computer needs to fetch it
- No calculations needed

Method:

`df.info()`

- Something DataFrame can **do**
- **Generate** a summary of its data
- Fancy word for function

Commands:

`sum()`

`max()`

`len()`

- Have parentheses because these are **actions**
- Computer adds up all values

Analogy

	Yourself 👤	Computer 🖥️
Attributes	• “What’s your name?” • “What’s your age?”	• Commonly accessed, fundamental info
Method	• “How many days has it been since you visited a park?”	• More complex questions • Generating summary data • Most operations

Big picture: You’ll need to differentiate between a DataFrame’s attributes and methods.

- When you’re accessing an **attribute**, like `columns` or `dtypes`, you don’t need to use parentheses
- When you’re using a **method**, you will use parentheses, and often arguments to the method as well

Data structures and descriptive statistics

Selecting columns

Selection

- Choosing specific part of data:
    - Rows
    - Columns
    - Individual values

- Selection with **lists**:

```
scores[0]
```

Selects the first item in `scores` list

Selecting columns

1️⃣ Select a single column with a line of code:

Returns a Pandas `Series`, a 1 dimensional data structure

```
languages = df["LanguagesAtHome"]
```

2️⃣ Select multiple columns by:

1. Saving the columns in a list

2. Placing that list inside the brackets

```
columns = ["CountryOfResidence","LanguageAtHome"]

country_columns = df[columns]
```

Data structures and descriptive statistics

Counts, sums, & histograms

DeepLearning.AI

Recap: Counts, sum, & histograms

1️⃣ Use the **`.count()`** method directly on:

- A dataframe to see the number of non-null values for each column:

```
df.count()
```

- A series:

```
df["Age"].count()
```

2️⃣ Use **`.hist()`** to plot histogram showing the distribution of a numerical feature

```
df["Income"].hist()
```

3️⃣ Use the **`.sum()`** method to add up all the values in a numeric column

```
df["HoursSpentLearningToCode"].sum()
```

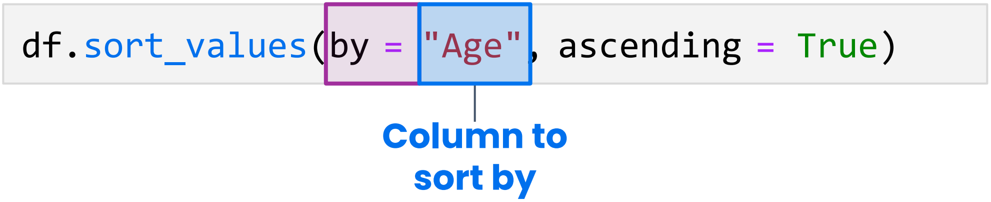
Data structures and descriptive statistics

Sorting

DeepLearning.AI

Recap: Sorting

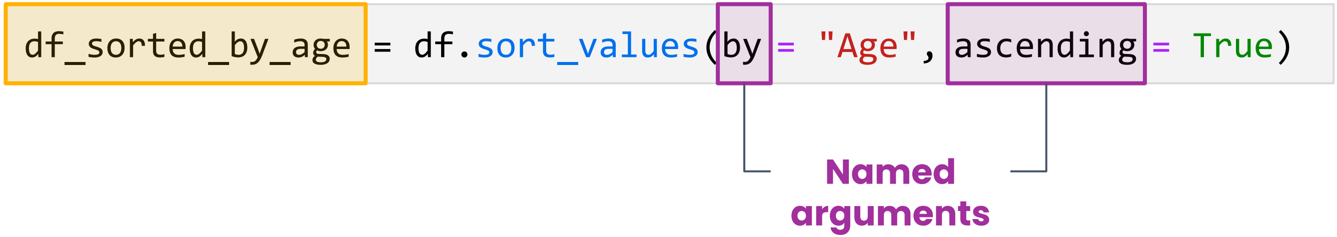
- Sort DataFrames using:



```
df.sort_values(by = "Age", ascending = True)
```

Column to sort by

- Save data frame in new variable



```
df_sorted_by_age = df.sort_values(by= "Age", ascending= True)
```

Named arguments

Data structures and descriptive statistics

Sorting by multiple columns

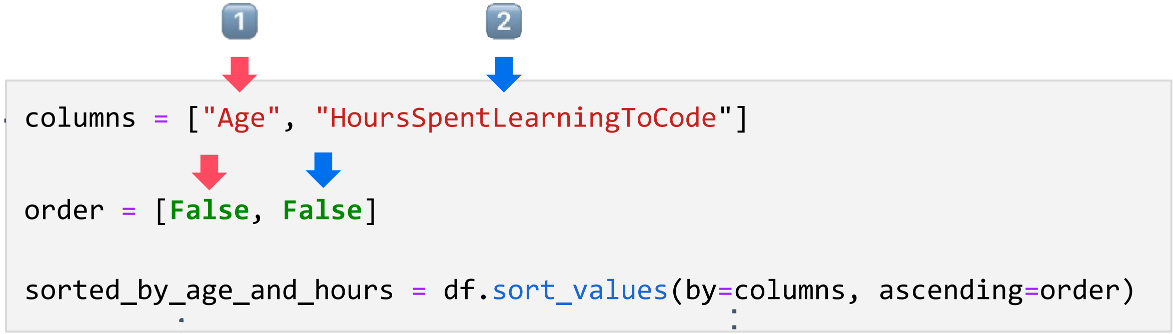
DeepLearning.AI

Recap: Sorting by multiple columns

List 1: Columns to sort by

List 2: Values for ascending

📉 In both cases, sort in descending order



```
columns = ["Age", "HoursSpentLearningToCode"]

order = [False, False]

sorted_by_age_and_hours = df.sort_values(by=columns, ascending=order)
```

Save `DataFrame` into a new variable, `sorted_by_age_and_hours`

Taking DataFrame and sorting by values:

- Sort by columns in the `columns` list
- Specifying to sort ascending or descending

DeepLearning.AI

Sean Barnes

Data structures and descriptive statistics

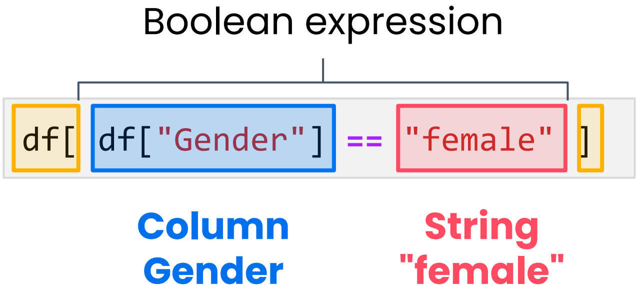
Filtering

Recap: Filtering

- To select rows that match condition:
    - Select from data frame with a boolean expression

- The boolean expression inside the brackets can be:
    - Equal to ==
    - Greater than >
    - Less than <

Example: Select only responses where the survey taker answered "female" for gender



Boolean expression

```
df[df["Gender"] == "female"]
```

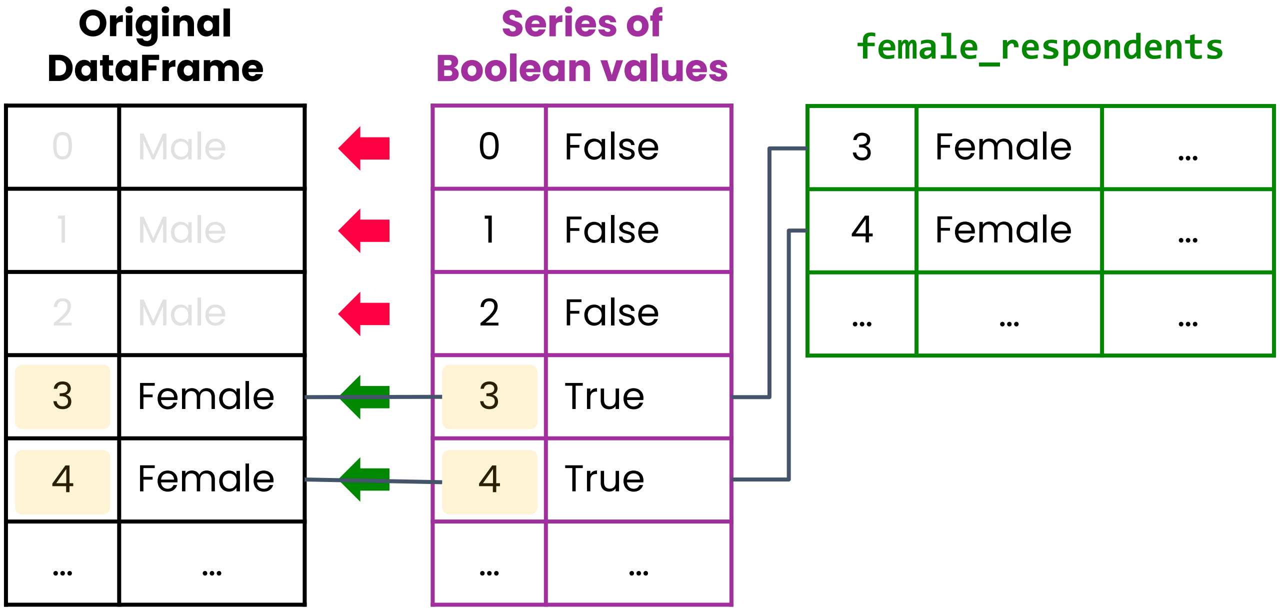
Column Gender

String "female"

Recap: Filtering

```
female_respondents = df[df["Gender"] == "female"]
```

- **True** if row is female
- **False** in any other case



Original DataFrame

0	Male
1	Male
2	Male
3	Female
4	Female
…	…

Series of Boolean values

0	False
1	False
2	False
3	True
4	True
…	…

`female_respondents`

3	Female	…
4	Female	…
…	…	…

Data structures and descriptive statistics

Filtering by multiple conditions

DeepLearning.AI

Recap: Filtering by multiple conditions

Filter by multiple conditions using:

- **And operator:** select rows that meet both conditions

```
female_above_30 = df[(df['gender'] == 'female') & (df['age'] >= 30)]
```

- **Or operator:** select rows that meet at least one conditions

```
female_or_above_30 = df[(df['gender'] == 'female') | (df['age'] >= 30)]
```

Data structures and descriptive statistics

Selecting rows

DeepLearning.AI

Indices

- **Numerical** indices

  Each row has a unique identifier that’s a number, starting with 0

- Pandas gives option to assign custom indices like:

  🎓 Student IDs

  🌍 Country codes

- Assign strings or other types

DataFrame

	Age	Hours
0	19	8
1	21	23
2	18	4
3	26	11
4	18	12

Series

0	Male
1	Male
2	Male
3	Female
4	Female

Recap: Selecting rows

- Access rows based on indices:

```
indexed_df = df.set_index("ID")

indexed_df.loc[25447]
```

- Select row based on its position:

First row is at index 0

```
indexed_df.iloc[0]
```

- To select a slice of rows:

```
indexed_df.iloc[1000:1006]
```

a	1000	✅
	1001	✅
	1002	✅
	1003	✅
	1004	✅
b - 1	1005	✅
b	1006	❌

- Select all the rows from a, including a, to b-1, not b

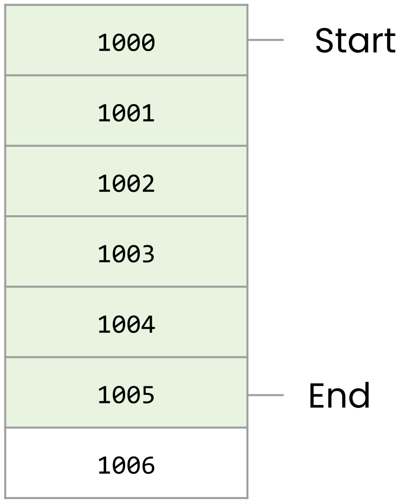
```
indexed_df.iloc[a:b]
```

A note on slicing

```
df.iloc[1000:1006]
```

- Counterintuitive that you type in the value when slicing, but don’t get that value

- **Main benefit**: Length of the slice is always end minus start



1000 Start

1001

1002

1003

1004

1005 End

1006

$$1006 - 1000 = 6$$

✅ Length of this slice is 6.

❌ If slice included last value, the length would be 7

Data structures and descriptive statistics

Central tendency, variability, and skewness

DeepLearning.AI

Recap: Central tendency, variability, and skewness

- Use **`.describe()`** on a series or DataFrame to calculate descriptive statistics about numerical features:

```
hours.describe()
```

```
count      14942.000000
mean          15.323317
median         10.00000
std           14.274867
min            0.000000
25%            5.000000
50%           10.000000
75%           20.000000
max          100.000000
```

- Calculate these values individually using the different pandas methods:

```
.mean()     .median()
.skew()
.max()       .min()
.std()       .var()
.quantile()
```

No arguments

```
hours.quantile(0.5)

hours.quantile([0.25, 0.5, 0.75])
```

DeepLearning.AI

Sean Barnes

Data structures and descriptive statistics

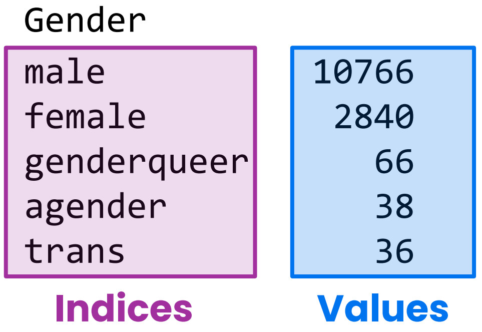
Categorical data

Recap: Categorical data

With categorical data you can:

- Use the **`.value_counts()`** method, which returns:
  - A `Series` in descending order of the number of occurrences
- Create a column chart by:
  - Saving the result of **`value_counts`**
  - Using **`.plot()`** method with the named argument **`kind = "bar"`**

```
df["Gender"].value_counts()
```



```
Gender
male           10766
female          2840
genderqueer       66
agender           38
trans             36
```

```
counts_of_gender = df["Gender"].value_counts()
counts_of_gender.plot(kind="bar")
```

Data structures and descriptive statistics

Correlation

Recap: Correlation

To create a scatter plot:

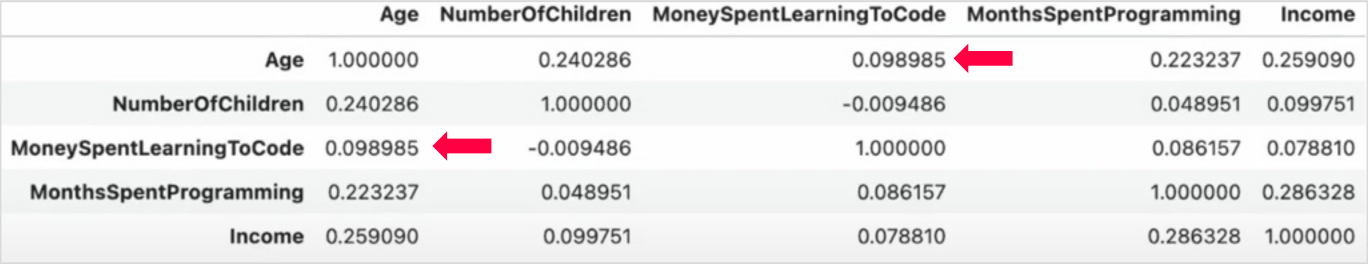
- □ Use `.plot()` method on a data frame
- □ “kind” named argument is “`scatter`”
- □ “x” named argument is feature on x axis
- □ “y” named argument is feature on y axis

To calculate correlations:

- □ Select only the numerical columns
- □ Use the `.corr()` method on the subset

```
selected_columns = df[columns]
selected_columns.corr()
```

```
df.plot(kind="scatter", x="Income", y="MoneySpentLearningToCode")
```



	Age	NumberOfChildren	MoneySpentLearningToCode	MonthsSpentProgramming	Income
Age	1.000000	0.240286	0.098985 ⬅	0.223237	0.259090
NumberOfChildren	0.240286	1.000000	-0.009486	0.048951	0.099751
MoneySpentLearningToCode	0.098985 ⬅	-0.009486	1.000000	0.086157	0.078810
MonthsSpentProgramming	0.223237	0.048951	0.086157	1.000000	0.286328
Income	0.259090	0.099751	0.078810	0.286328	1.000000

Data structures and descriptive statistics

Segmentation by one feature

DeepLearning.AI

Recap: Segmentation by one feature

Groupby **doesn't calculate** until you take the next step to perform some calculations

For segmentation, use the **`.groupby()`** method:

```
df.groupby("NumberofChildren")
```

```
pandas.core.groupby.generic.DataFrameGroupBy
```

Select columns from the groupby:

```
df.groupby("NumberofChildren")["HoursSpentLearningToCode"]
```

Perform computations:

```
df.groupby("NumberofChildren")["HoursSpentLearningToCode"].count()
```

… sum, or mean, or other descriptive statistics

```
NumberOfChildren

1.0          1050
2.0           938
3.0           342
4.0           109
5.0            37
6.0            10
7.0             1
8.0             1
9.0             2
10.0            2
12.0            1
15.0            0
18.0            1
```

Data structures and descriptive statistics

Segmentation by multiple features

DeepLearning.AI

Recap: Segmentation by multiple features

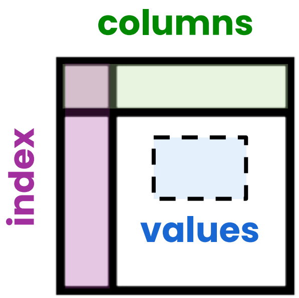
Use to group data by more than one feature:

```
df.pivot_table(index = "NumberOfChildren",
               columns = "IsSoftwareDev",
               values = "HoursSpentLearningToCode" )
               aggfunc = "sum")
```

Features to group by

Outcome of interest

Default: values will be summarized using the mean



- Use the aggfunc named argument to specify a different summarization

  Some options: sum, count, std, max, and min

- Provide a list for aggfunc argument, which gives a pivot table containing summarization functions

```
...aggfunc = ["min", "median", "max"])
```

DeepLearning.AI

Sean Barnes