

ID430B: Data Analytics for Designers 디자인 특강V <디자이너를 위한 데이터 분석>

Lecture 2

Python Basic & Data Format

Tak Yeon Lee <takyeonlee@kaist.ac.kr> (takyeonlee.com)
AI-Experience-Lab (reflect9.github.io/acl)

Goals

1. Python Prerequisites

2. Two typical formats of Data

1. CSV (Comma Separated Values) and JSON (JavaScript Object Notation)
2. Learning how to use CSV and JSON in the context of DIKW

1. Python Prerequisites

Python Prerequisites (1/2)

- If you have not studied Python before, there are many basic usages to learn. However, teaching basics is not part of this course for two reasons – (1) a huge variation of students' coding experience, and (2) so many amazing online tutorials. Instead, we provide a list of relevant concepts and recommends online learning resources.

Prerequisites	Check your knowledge	Recommended Online Resources
Install Python	Most Windows / Mac computers already have Python 3 installed. Check if Python 3.7 or above (do not use Python 2 here) is installed on your machine. If not, install the latest Python 3 (probably 3.9.x)	<ul style="list-style-type: none">- Real Python (Installation & Setup guide) - <i>lengthy, very detailed</i>- Search “checking python version” on Stack Overflow - <i>concise, solution-oriented</i>
Hello World in both interactive and script modes	There are multiple ways to execute Python codes. Check if you know how to use the interactive and the script modes.	<ul style="list-style-type: none">- Tutorial Point (Basic Syntax) – <i>Easy and Concise</i>- Real Python (Interactive with Python) – <i>Nice explanation, but skip IDLE and Thonny sections</i>
Variables & Simple Data Types	Make sure you understand simple data types (e.g. String, Number, Boolean) and variables (e.g. how to store values in variables)	<ul style="list-style-type: none">- Real Python (Data Types & Variables)- Learn-by-Examples (Numbers)
Operators	Check you understand all types of operators supported in Python (e.g. arithmetic, comparison, assignment, logical, bitwise, membership, identity)	<ul style="list-style-type: none">- Learn-by-Examples (Operators)

CONTINUE TO NEXT PAGE

Python Prerequisites (2/2)

Prerequisites	Check your knowledge	Recommended Online Resources
Conditionals	Make sure you are familiar with if, elif, else constructs.	- W3school (Python If...Else)
For Loops, While Loops	Make sure you know how to use Python's "for...in", and "while" loops. Also <code>range()</code> method is used to repeat N times.	- W3School (While Loop , For Loop)
Code block by Indentation	Make sure you know how to set a block of code in Python by adding indentation (by pressing Tab button)	- Programiz (Python Statement, Indentation and Comments) - Ask Python (Indentation)
List	Basic list operations such as creating list, access / update / delete elements in list. Be familiar with nested list (a.k.a 2D array or list-of-list)	- Learn-by-Examples (List , Nested-List , List Slicing) - Real Python (List and Tuples): <i>advanced</i>
Dictionary	What is Python Dictionary? How to create / modify a dictionary? How to iterate through a dictionary?	- Learn-by-Examples (Dictionary , Nested-Dictionary) - Real Python (Dictionaries in Python)

- Hopefully you don't need to spend much time for studying the above concepts.
- Do you feel overwhelmed? You should catch as soon as possible with the above tutorials. Or, it's never too late to drop.
- If you decided to stick with us, trust me. Learning Python is one of the best places to invest your time and effort.

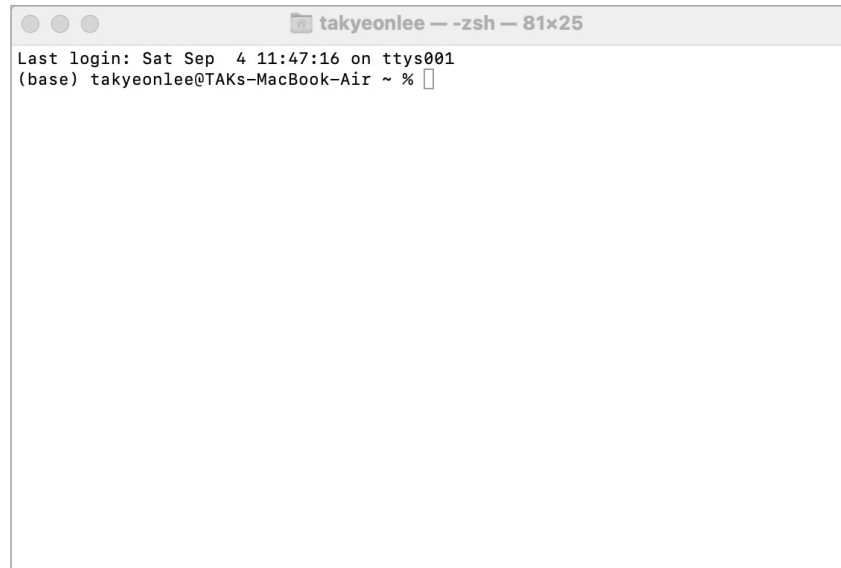
Where to Code

Interactive Console

Python Notebook

IDE (Integrated Dev. Env.)

Online Editors

A screenshot of a terminal window titled 'takeyeonlee - zsh - 81x25'. The window shows the output of the 'last' command: 'Last login: Sat Sep 4 11:47:16 on ttys001'. Below this, the prompt '(base) takeyeonlee@TAKs-MacBook-Air ~ %' is displayed, followed by a cursor. The terminal window has a standard macOS title bar with three colored buttons (red, yellow, green) on the left.

- From Terminal (OSX) or Command line (Windows), execute “python3”
- Included in every Python packages (no need to install anything else)
- Can access local files and libraries
- Execute Line-by-Line (ideal for simple tasks)

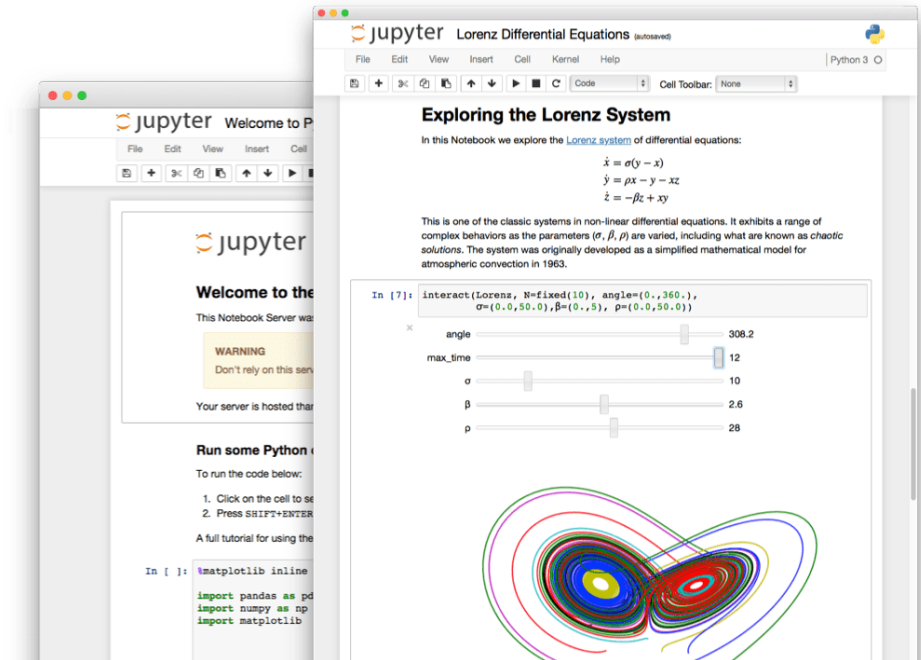
Where to Code

Interactive Console

Python Notebook

IDE (Integrated Dev. Env.)

Online Editors



- Need to install an additional package (e.g. [JupyterLab](#))
- Similar with interactive console (i.e. line-by-line execution), but it runs on web-browser (with a server running on your own machine)
- Can access local files and libraries
- Can share scripts and its visual outcomes + simple UI controls
- Ideal for executing a long scripts; But not suitable for complex applications (since they might have lots of modules running in parallel)
- We will use JupyterLab frequently later in this course.

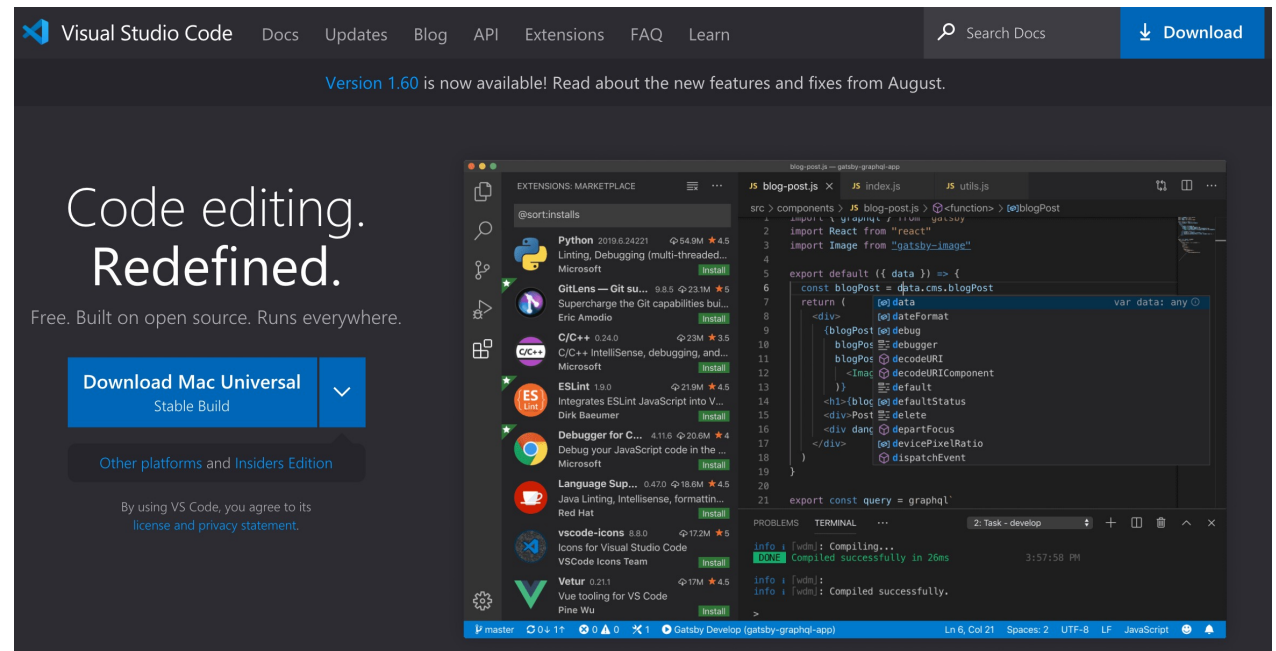
Where to Code

Interactive Console

Python Notebook

IDE (Integrated Dev. Env.)

Online Editors



- Need to install an application (e.g. [Visual Studio Code](#), [Sublime](#), PyCharm, etc)
- Suitable for running automated tasks / building complex applications
- Can access local files and libraries

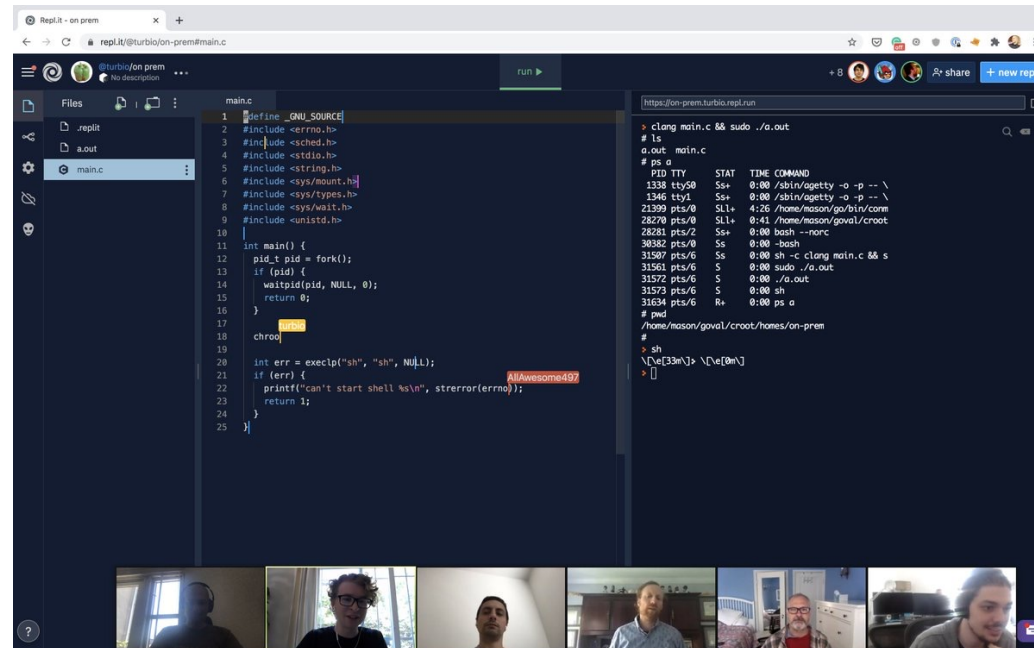
Where to Code

Interactive Console

Python Notebook

IDE (Integrated Dev. Env.)

Online Editors



- IDE running on web-browser
- From fully-functioning IDE (e.g. [Repl.it](#)) to simple editor (e.g. [PythonPad](#))
- Hard to integrate with local resources (e.g. data files, complex libraries)
- Ideal for remote education and collaboration

Where to Code

~~Interactive Console~~

Python Notebook

IDE (Integrated Dev. Env.)

Online Editors

Which one is the best for this course?

- **Interactive console** is easy to start, but lacks essential usability supports. **No need to use Python interactive console**
- Other options are hard to compare because each has its own pros and cons.
 - **Python Notebook** is a standard tool for doing data science – but requires some environmental setting.
 - **IDE** (e.g. Visual Code, Sublime Text) is relatively easy to begin, and suitable for building large applications. However, IDE is not as convenient as Notebook for data analytics.
 - **Online Editor** is ideal for doing small exercises on the fly (or collaborating with other people). We may give some assignments in an Online Editor.

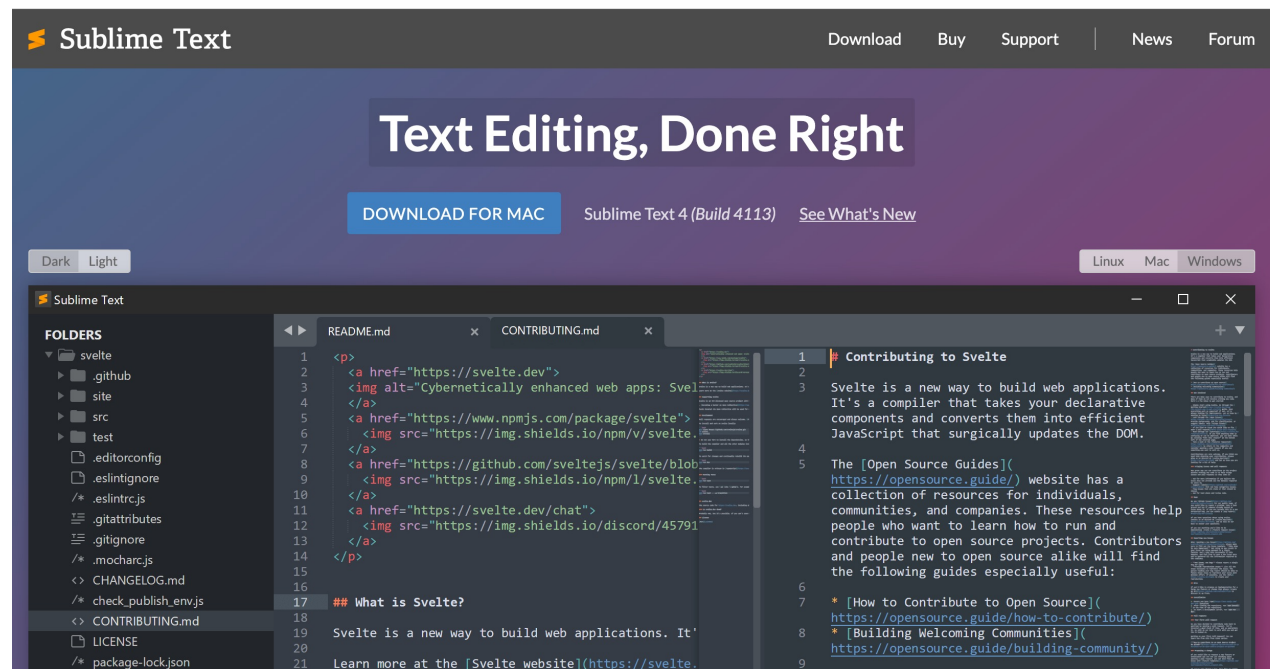
Today we will use an IDE (Sublime Text), but you can use any option that you feel comfortable with. Later in this course you will have chances to use the three options.

IDE > Sublime Text

Sublime Text is a free code editor that you can download from (<https://www.sublimetext.com/>)

Step 1. Create a .py file containing Python code

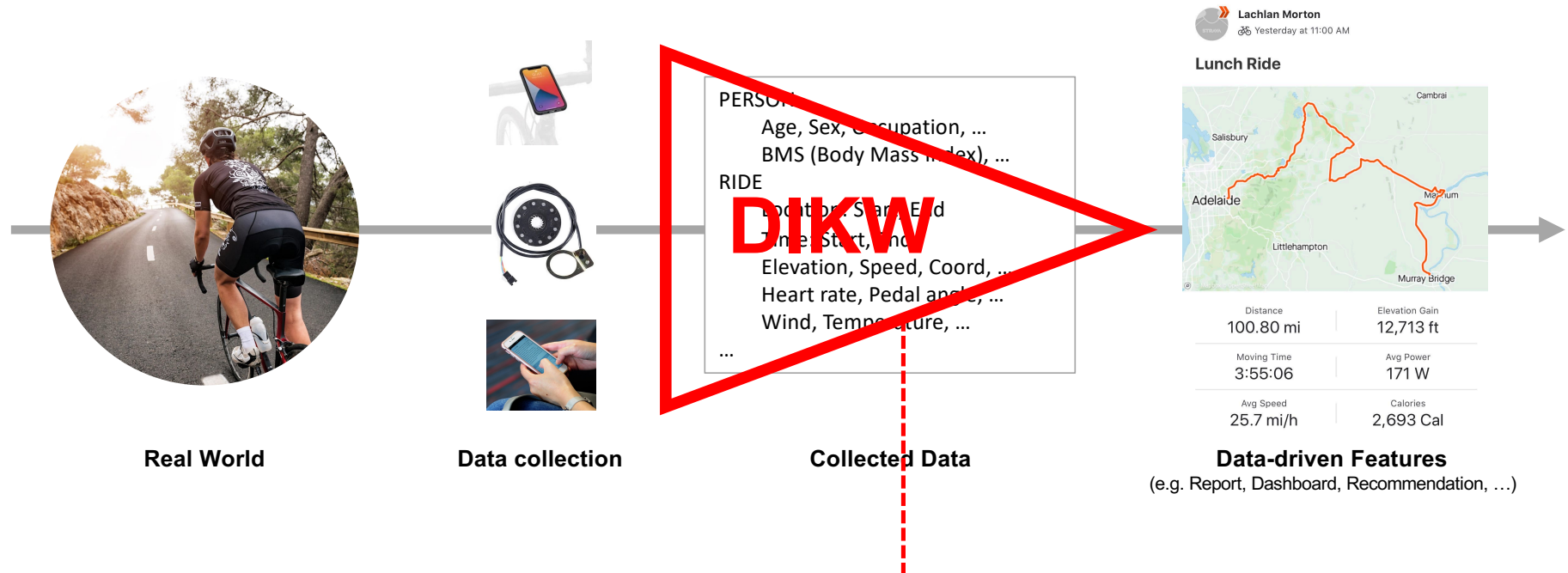
Step 2. On terminal (OSX) or command line (Windows), run the .py files



2. CSV and JSON

Two Typical Data Formats

Why do we care about data format?



- Raw data is usually given as multiple CSV files / streams
- While moving up the ladder of the DIKW model, we restructure CSV (i.e. nested list; tabular data) to JSON (i.e. Python dictionary; hierarchical data), or vice versa.

Original Dataset

ID	FIRSTNAME	LASTNAME	GENDER	AGE
1	Tak Yeon	Lee	male	99
2	Toomim	Tuchinda	feale	30
3	Jane	Doe	female	25

CSV Comma Separated Values

```
ID,FIRSTNAME, LASTNAME, GENDER, AGE
1,Tak Yeon, Lee, male, 99
2,Toomim, Tuchinda, female, 30
3,Jane, Doe, female, 25
```

- A CSV file is a plain text, but represents tabular data that has **rows** and **columns**.
- Rows are separated by line-break (e.g. "\n")
- Columns are separated by a specific character (e.g. comma(,), tab(\t), semi-colon(;), ...), also called delimiter.
- Once CSV data is loaded on Python, it usually becomes a nested list.

```
E.g. [["1", "Tak Yeon", "Lee", "male", "99"],
      ["2", "Toomim", "Tuchinda", "female", "30"],
      ...]
```

JSON JavaScript Object Notation

```
[
  {
    "ID": 1,
    "FIRSTNAME": "Tak Yeon",
    "LASTNAME": "Lee",
    "GENDER": "male",
    "AGE": 99
  },
  ...
]
```

- JSON is plain text too, but stores hierarchical data containing key-value pairs (i.e. same as Python dictionary)
- A value is either text, number, list, or another dictionary

CSV Comma Separated Values

- Rows must have consistent structure (i.e. same # columns and data type)

```
Date,Country,Units,Revenue
2019-01-08,USA,343,15461.36
2019-01-04,Panama,93,4681.26
2019-01-07,Panama,42,2220.36
2019-01-16,Brazil,103,1853.78
2019-01-17,USA,28,286.3
2019-01-24,Canada,372,24826.98
2019-01-26,Canada,61,1592.42
2019-01-28,Canada,264,3228.11
2019-01-13,Canada,27,257.97
2019-01-28,Brazil,323,3024.25
```

- Hard to extend / modify structure

JSON JavaScript Object Notation

- No concept of rows and columns. Values may have different structures

```
1 //Student JSON Object
2 {
3   "rollNumber" : 11,
4   "firstName" : "Saurabh",
5   "lastName" : "Gupta",
6   "permanent" : false,
7   "address" : {
8     "addressLine" : "Lake Union Hill Way",
9     "city" : "Atlanta",
10    "zipCode" : 50005
11  },
12  "phoneNumbers" : [ 2233445566, 3344556677 ],
13  "cities" : [ "Dallas", "San Antonio", "Irving" ],
14  "properties" : {
15    "play" : "Badminton",
16    "interst" : "Math",
17    "age" : "34 years"
18  }
19 }
```

Annotations:

- ↑ JSON for number values without doublequote
- ↑ JSON for String values with double quote
- ↑ JSON for boolean allow values true/false
- ↑ JSON for address object with in curly bracket
- ↑ Array of Numeric Values
- ↑ Array of String values
- ↑ JSON to represent map in key value pairs

(from) JSON Example | [Facing Issues on IT](#)

- Easy to extend / modify structure

CSV Comma Separated Values

- 2D data can directly construct a matrix, which is suitable for machine learning (ML)

Title	Author	Pages	Genre
Alice in Wonderland	Lewis Carroll	124	Fantasy
Hamlet	William Shakespeare	98	Drama
Treasure Island	Robert L. Stevenson	280	Adventure
Peter Pan	J.M. Barrie	94	Fantasy

Features
X

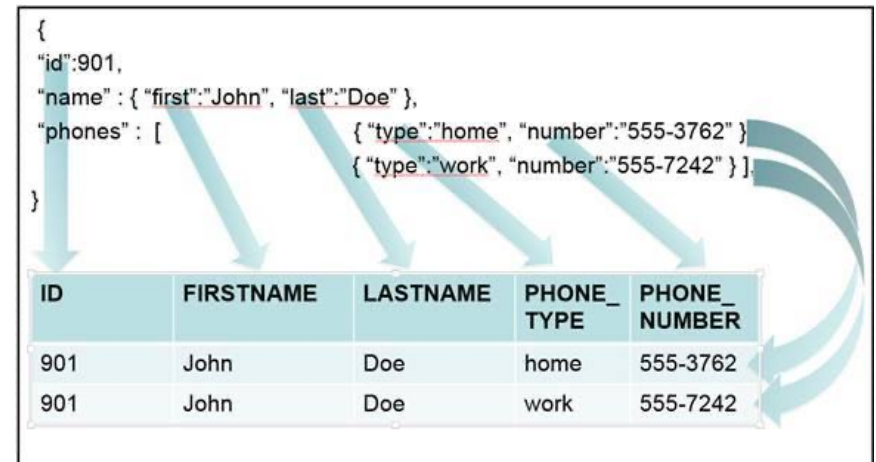
Labels
y

ML Goal: "We need a ML model that predicts a movie's *Genre* based on its *Title*, *Author*, and *Page*"

The **Genre** column becomes **y** (labels to predict), and the **Title**, **Author**, and **Pages** columns become **X** (features).

JSON JavaScript Object Notation

- Hierarchical data must be manually transformed to a matrix (i.e. relational form) before feeding ML models



(from) <https://developer.ibm.com/articles/i-json-table-trs/>

Common Pattern > Log Analysis



Each rows of CSV represents low-level unit data such as events

UUID	EVENT	TIMESTAMP
11011	VISIT	9/5 13:01
21022	VISIT	9/5 13:03
11011	PURCHASE	9/5 13:05

Ideal for storing unit data

Transformed for specific goals

```
{
  "11011": {
    "NUM_EVENTS": {
      "VISIT": 1,
      "PURCHASE": 1
    }
  },
  "21022": {
    "NUM_EVENTS": {
      "VISIT": 1
    }
  }
}
```

Frequency of events per user

- What % of visitors did purchase items?
- How users did visit and purchase?
- Is there any user showing interesting behavioral patterns?

A lot of hands-on experience needed

```
{
  "VISIT": {
    "UUID": [
      11011, 21022
    ]
  },
  "PURCHASE": {
    "UUID": [
      11011
    ]
  }
}
```

Users per event type

- How many visitors do we have?
- What is the proportion of two event types?

Answers to the above questions are simple values or CSV

Common Pattern > Bicycle Riding



The below CSV data is collected every minute while cycling

TIMESTAMP	SPEED	TERRAIN
9/5 13:01	6	UNPAVED
9/5 13:02	10	PAVED
9/5 13:03	12	PAVED
9/5 13:04	14	PAVED
9/9 17:25	8	PAVED
9/9 17:26	15	PAVED
9/9 17:27	3	UNPAVED

From the target activity / phenomenon

- Collect unit data stored in CSV
- Read CSV, and convert to JSON (i.e. Python dictionary; hierarchical data) for specific purposes
- Gain Knowledge and Wisdom for making decisions

JSON often contains aggregated information to answer specific questions

```
{  "UNPAVED": {    "SPEED": [      6, 3    ]  },  "PAVED": {    "SPEED": [      10, 12, 14,      8, 15    ]  }}
```

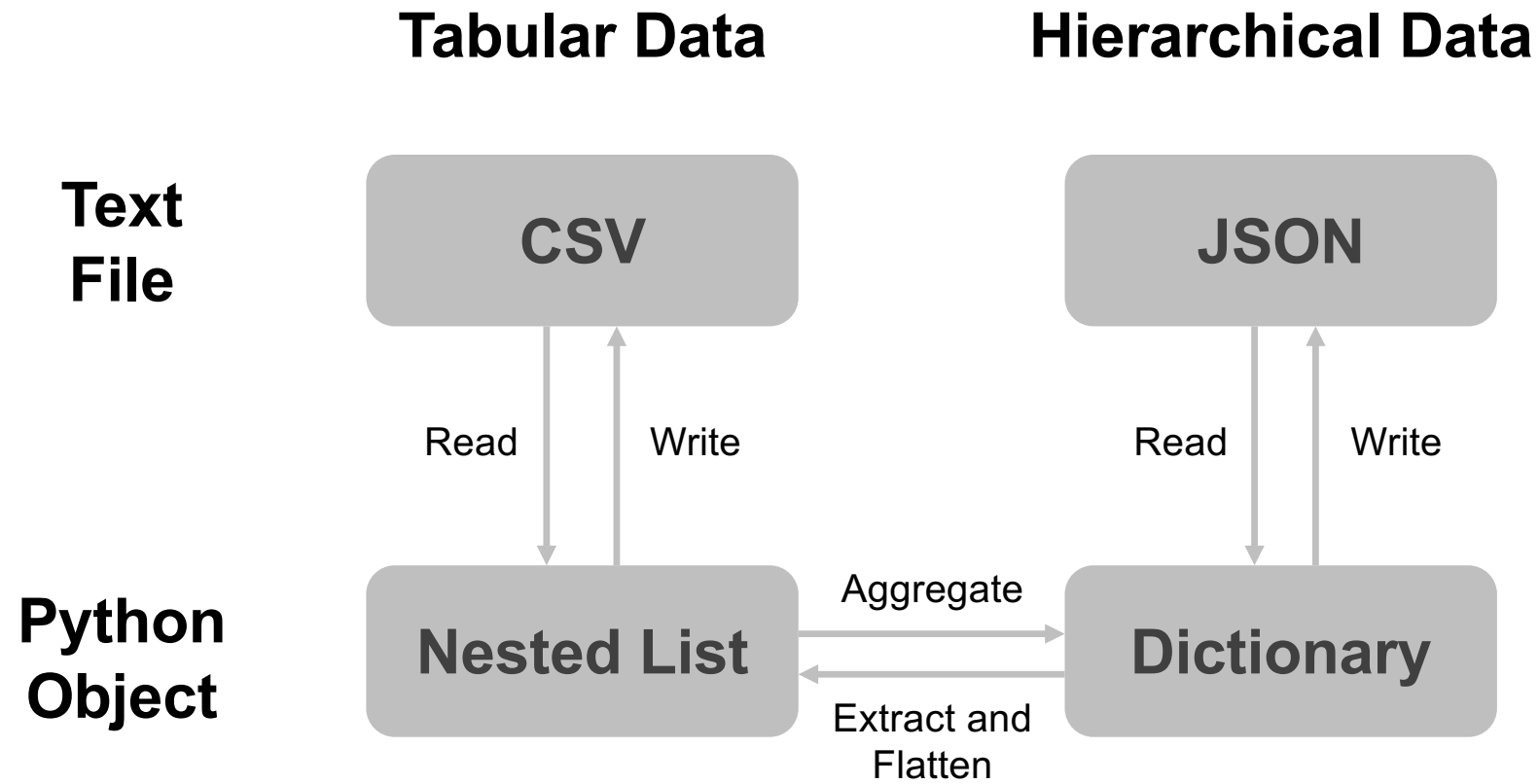
Speed aggregated for types of terrains

- Compare distributions of speed per types of terrain

```
{  "TIMESTAMP_GROUP_1": {    "DURATION": 00:04,    "TERRAIN": {      "UNPAVED": 1,      "PAVED": 3    }  },  "TIMESTAMP_GROUP_2": {    "DURATION": 00:02,    "TERRAIN": {      "UNPAVED": 1,      "PAVED": 2    }  }}
```

Rows grouped by Neighboring Timestamps

- Calculate the user's average duration of each group
- Predicting terrain for a given duration



Summary

CSV	JSON
Both CSV and JSON files are plain text	
Represents tabular data E.g. Nested list	Represents hierarchical data E.g. Dictionary
Often stores Unit data	Often stores aggregated information for specific purposes
In most cases, we utilize data by transforming between tabular (CSV) and hierarchical (JSON) structure	