Algorithmic Methods of Data Mining

Basic Tools and First Practice

Ioannis Chatzigiannakis

Sapienza University of Rome

Laboratory 2

Development Tools

Programming Tool

A programming tool or software development tool is a computer program that software developers use to create, debug, maintain, or otherwise support other programs and applications.

- Source Code Editor
- Debugger or Profiler
- Bug Tracking System
- Documentation Generators
- Revision Control
- Performance Analysis
- Collaborative Programming
- Cloud-based IDEs

Integrated Development Environment (IDE)

A programming tool or software development tool is a computer program that software developers use to create, debug, maintain, or otherwise support other programs and applications. The IDE is meant to make programming a more productive process.

- Organize project files
- Searching
- Source Code Editor
- Debugger
- Tasks & Annotations related to code
- Documentation Generators
- Revision Control
- Code Analysis

Jupyter Notebook

- Interactively developing and presenting data science projects.
- A single document integrates: code and its output, visualizations, narrative text, mathematical equations, and other rich media.

Installation & Execution

Installation:

- For Windows - make sure you first install Anaconda, then use pip.
- For Mac / Linux use directly pip:

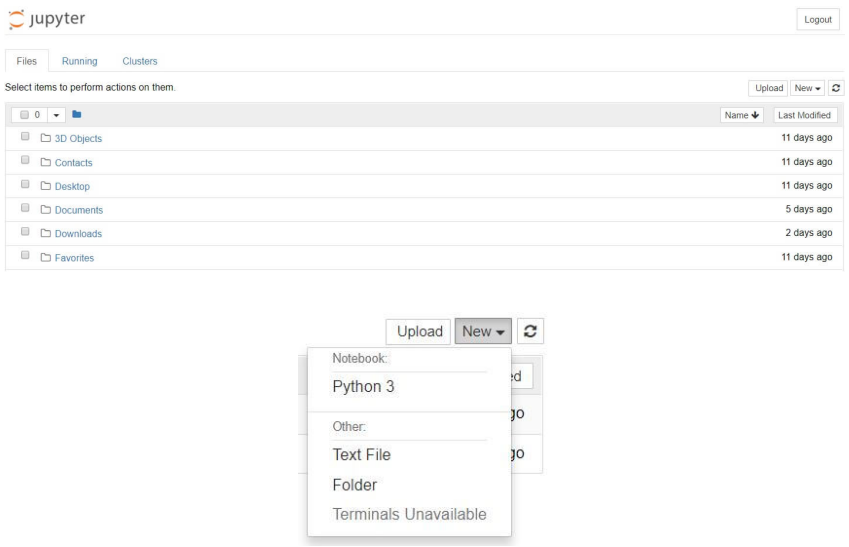
```
pip3 install jupyter
```

Execution:

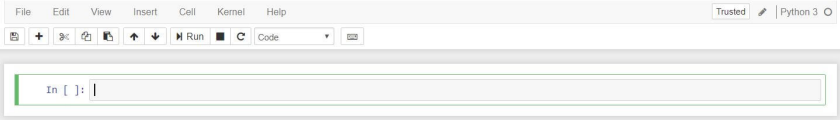
- For Windows - via Anaconda.
- For Mac / Linux from the command line:

```
jupyter notebook
```

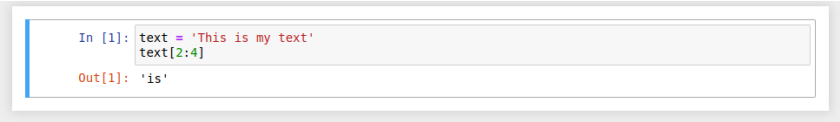
- The jupyter interface is avalable at `http://localhost:8888/tree`



The Notebook Interface



- A Cell can be either Code or Markdown
- Use the ▶| Run button or `CTRL+ENTER` to execute the Code or present the Markdown.



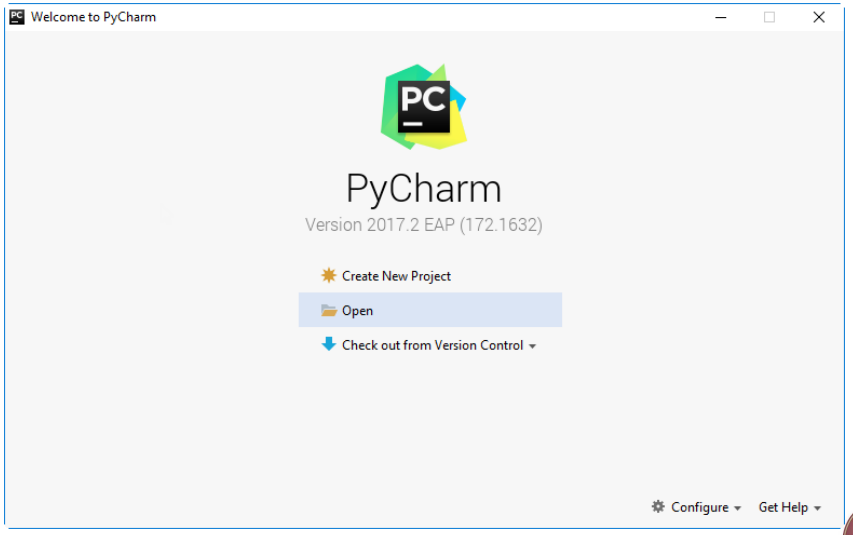
- Check out that In [ ]: has changed to In [1]:
- When python is processing the code we get In [*]:

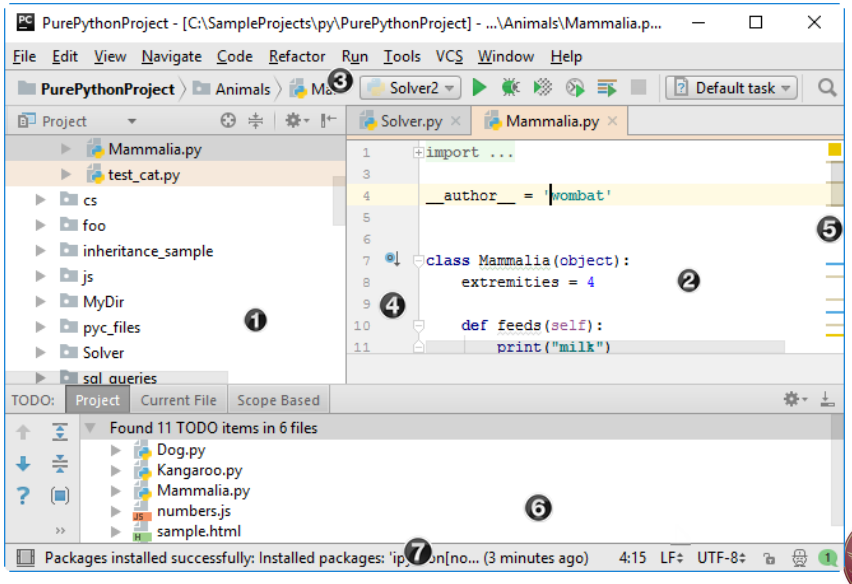
Navigating the Notebook with the Keyboard

- There is always a Cell Active.
- You can **Stop Editing** by using the ESC key.
- You can **Start Editing** by using the ENTER key.
- When **NOT** in Editing mode:
  - We can go up/down the cells using the Up and Down keys.
  - To change the Cell type to Markdown use the M key.
  - To change the Cell type to Code use the Y key.
  - To insert a new Cell above the current Cell use the A key.
  - To insert a new Cell below the current Cell use the B key.
  - To delete a Cell use the D key **twice**.
  - To UNDO a delete command use the Z key.

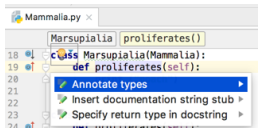
pyCharm: Python IDE for Professional Developers

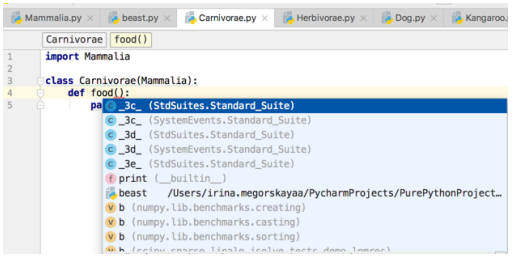
- Keyboard-centric approach
- Smart assistance
- Code quality tools
- Cross technology development
- Navigation and Refactoring
- Database support
- Scientific tools





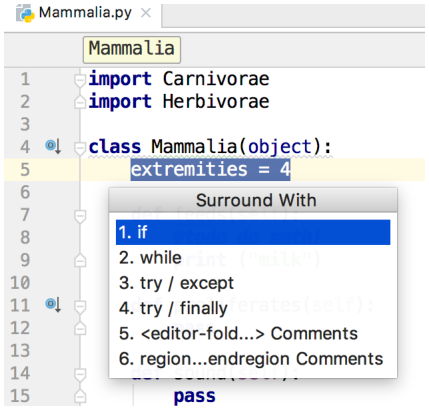
Code with smart assistance

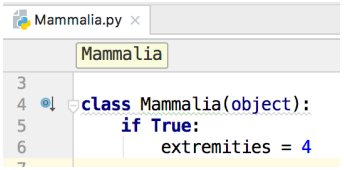




- Intention Action – indicated with a bulb ALT+Enter
  - Suggestions based on the action that you do that intend to save time.
  - Remark that the code needs to be correct for this feature to work.
- Code completion
  - Auto-complete function/variable names.

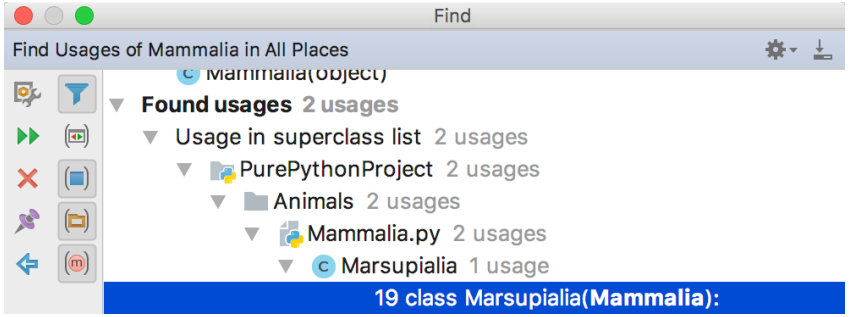
Live Templates





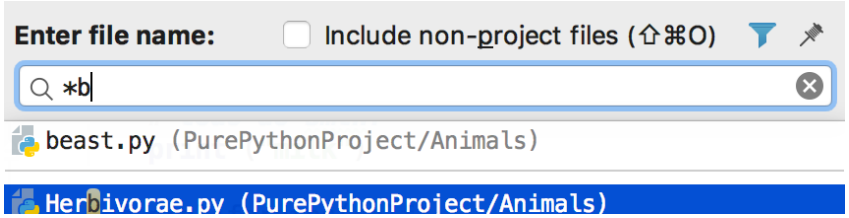
- Live Template CTRL+J produce entire code constructs.
- A library of ready-to-use templates.

Search for Usages



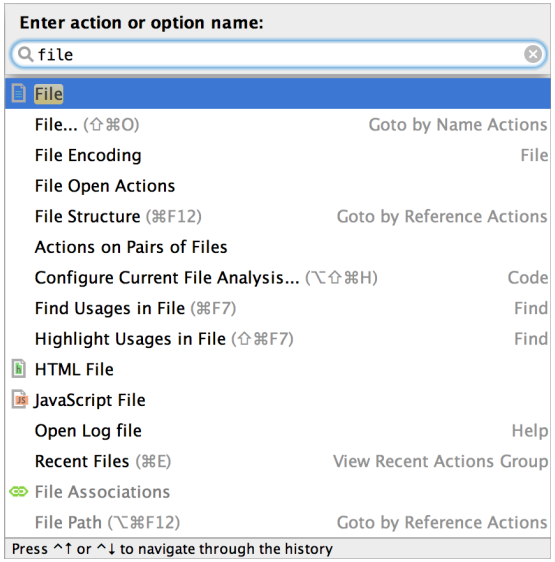
- As the project grows, or when you work with someone else's code.
- To find where a particular symbol is used, ALT+F7
  - All files are searched.

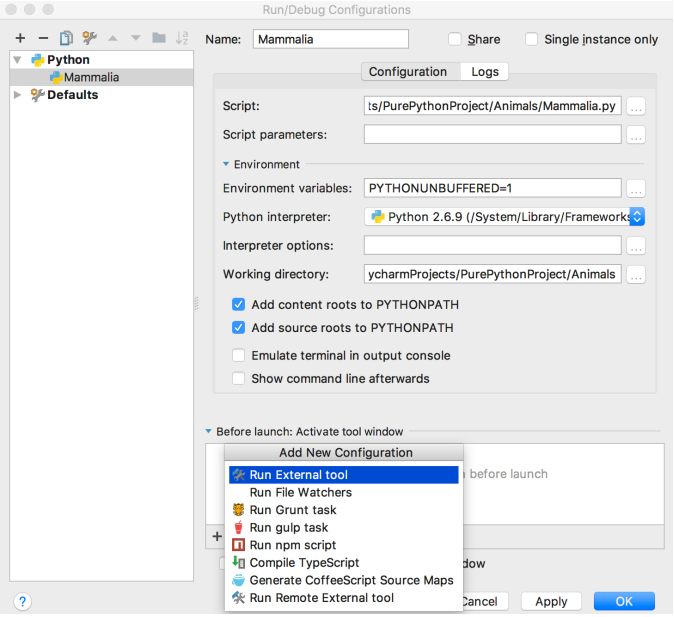
Project navigation – Find by name

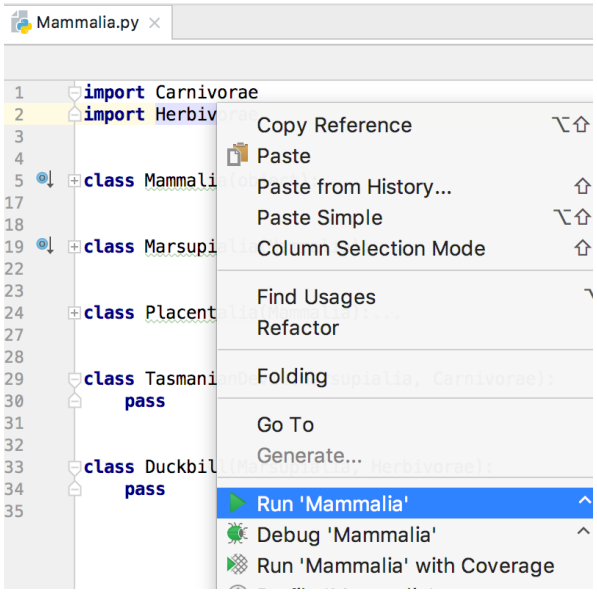


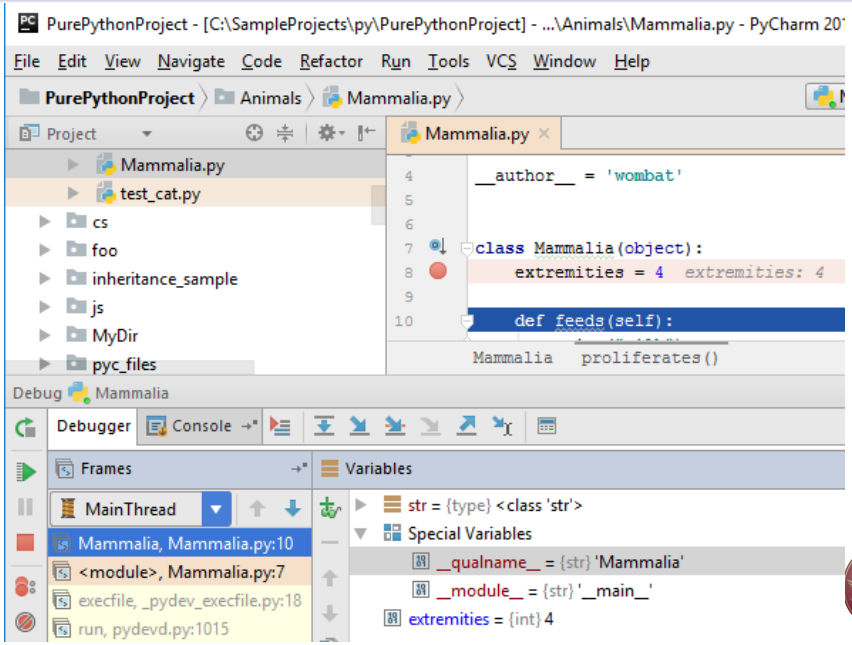
- Search only Classes by name, CTRL+N
- Search only based on filenames, CTRL+Shift+N
- Search Variable, CTRL+Shift+ALT+N
- Search Declaration, CTRL+B
- Search Class/Function, CTRL+U

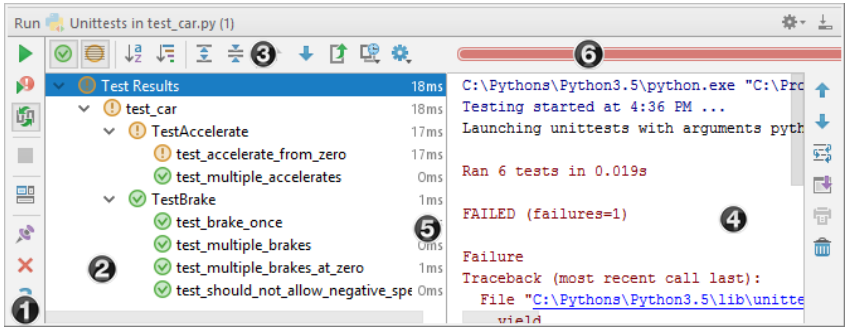
Find Action – CTRL+Shift+A



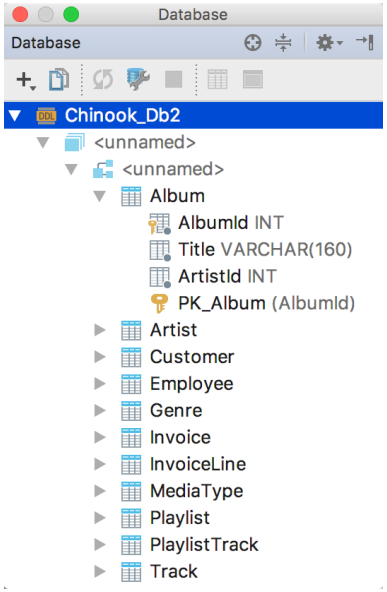

Run/Debug Configurations
Name: Mammalia
Share
Single instance only
Python
Mammalia
Defaults
Configuration
Logs
Script: !s/PurePythonProject/Animals/Mammalia.py
Script parameters:
Environment
Environment variables: PYTHONUNBUFFERED=1
Python interpreter: Python 2.6.9 (/System/Library/Frameworks
Interpreter options:
Working directory: ycharmProjects/PurePythonProject/Animals
Add content roots to PYTHONPATH
Add source roots to PYTHONPATH
Emulate terminal in output console
Show command line afterwards
Before launch: Activate tool window
Add New Configuration
Run External tool
Run File Watchers
Run Grunt task
Run gulp task
Run npm script
Compile TypeScript
Generate CoffeeScript Source Maps
Run Remote External tool
Cancel
Apply
OK


Mammalia.py
import Carnivorae
import Herbiv
class Mammali
class Marsupi
class Placent
class Tasmani
pass
class Duckbil
pass
Copy Reference
Paste
Paste from History...
Paste Simple
Column Selection Mode
Find Usages
Refactor
Folding
Go To
Generate...
Run 'Mammalia'
Debug 'Mammalia'
Run 'Mammalia' with Coverage


PurePythonProject - [C:\SampleProjects\py\PurePythonProject] - ...\Animals\Mammalia.py - PyCharm 20
File Edit View Navigate Code Refactor Run Tools VCS Window Help
PurePythonProject Animals Mammalia.py
Project
Mammalia.py
test_cat.py
cs
foo
inheritance_sample
js
MyDir
pyc_files
__author__ = 'wombat'
class Mammalia(object):
extremities = 4 extremities: 4
def feeds(self):
Mammalia proliferates()
Debug Mammalia
Debugger
Console
Frames
Variables
MainThread
Mammalia, Mammalia.py:10
<module>, Mammalia.py:7
execfile, _pydev_execfile.py:18
run, pydevd.py:1015
str = {type} <class 'str'>
Special Variables
__qualname__ = {str} 'Mammalia'
__module__ = {str} '__main__'
extremities = {int} 4


Run Unittests in test_car.py (1)
Test Results 18ms
test_car 18ms
TestAccelerate 17ms
test_accelerate_from_zero 17ms
test_multiple_accelerates 0ms
TestBrake 1ms
test_brake_once
test_multiple_brakes
test_multiple_brakes_at_zero 1ms
test_should_not_allow_negative_spe 0ms
C:\Pythons\Python3.5\python.exe "C:\Prc
Testing started at 4:36 PM ...
Launching unittests with arguments pyth
Ran 6 tests in 0.019s
FAILED (failures=1)
Failure
Traceback (most recent call last):
File "C:\Pythons\Python3.5\lib\unitte

pyCharm



Introduction

What is open data?

- Open data is data that anyone can access, use and share.
- Open data becomes usable when made available in
  - a common format,
  - a machine-readable format.
- Open data must be licensed, permitting people to
  - use the data in any way they want,
  - transform it,
  - combine it with other data,
  - sharing it with others, even commercially.

Introduction

Is open data free?

- Open data must be free to use, but this does not mean that it must be free to access.
- There is often a cost to creating, maintaining and publishing usable data.
- This cost tends to be negligible for many datasets.
- Live data and big data can incur ongoing costs related to reliable service provision.
- Once the user has the data, they are free to use, reuse and redistribute it – even commercially.

Introduction

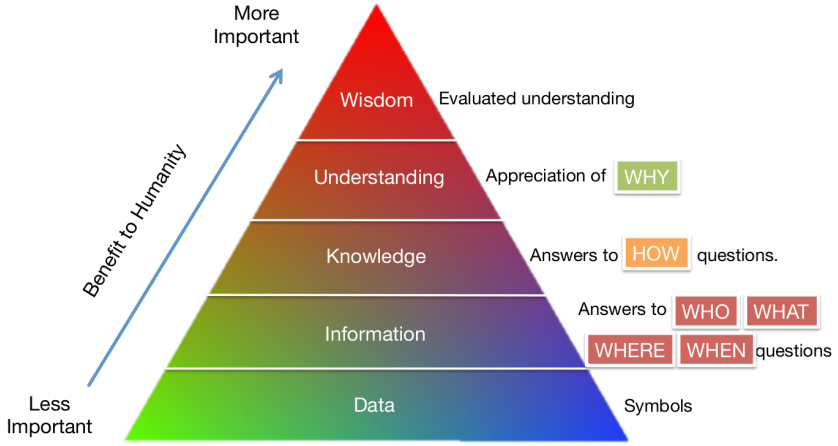
Why do we need open data?

- Open data can help bring diverse benefits to
  - Governments
    - Make governments more transparent.
    - Provide the evidence that public money is being well spent.
    - Provide the evidence that policies are being implemented.
  - Businesses
    - New opportunities for businesses to connect with customers.
  - Civil Society
    - Help us understand our personal impacts on the environment, and take steps to improve it.

`https://www.europeandataportal.eu/`

`https://github.com/GoogleCloudPlatform/covid-19-open-data`

From raw material to new information and knowledge



A common, machine-readable format

- Several machine-readable formats exist:
  - CSV, JSON, XML, ...
- CSV – comma separated values
  - A spreadsheet format, e.g., Excel or Google Sheets
  - Each row is one observation, and the same values are recorded for each observation.
  - A flat data format – you only need to know the row number + column number to get a value.
- JSON – JavaScript Object Notation
  - A very common way to store data on the web.
  - A series of objects that span in more than one line.
  - Each object can have multiple keys/values pairs or other objects within it.
  - A hierarchical data format – you may need to know the structure of the objects to get a value.

CSV Example

```
"id","name","address","regular"
1,"John","12 Totem Rd. Aspen",true
2,"Bob",null,false
3,"Sue","\"Bigsby\", 345 Carnival, WA 23009",false
```


JSON Examples

```
{"id":1, "name":"John",
 "address":"12 Totem Rd. Aspen",
 "regular":true},
{"id":2, "name":"Bob", "regular":false},
{
 "id":3,
 "name":"Sue",
 "address":"\"Bigsby\", 345 Carnival, WA 23009",
 "regular":false
}
```

Kaggle: eCommerce behavior data

- Kaggle is probably world's largest data science community.
- We will work with data on eCommerce behavior from multi category store.
  - Data for 7 months: October 2019 . . . April 2020
  - 285 million users' events from the eCommerce website
  - https://www.kaggle.com/mkechinov/ecommerce-behavior-data-from-multi-category-store
- The dataset is formated using CSV
  - Each row in the file represents an event.
  - All events are related to products and users.
  - Each event is like many-to-many relation between products and users.

CSV + Python + Pandas – Example

- Download the small version of the CSV dataset from: https://www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop
- The dataset contains 5 files, one file for each month.
- We will use Python and Pandas to load the data.
- You need to install Pandas:

```
pip3 install pandas
```


CSV + Python + Pandas – Example

```
import pandas as pd
dataset = pd.read_csv('2020-Jan.csv', sep=',',
                      delimiter=None, header='infer',
                      names=None, index_col=None,
                      usecols=None,
                      encoding = "ISO-8859-1",
                      nrows=20)
```

Check out the manual page for details on the different parameters used:

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html


```
In [1]: import pandas as pd

In [2]: dataset = pd.read_csv('2020-Jan.csv', sep=',',
                              delimiter=None, header='infer',
                              names=None, index_col=None,
                              usecols=None,
                              encoding = "ISO-8859-1",
                              nrows=20)

In [3]: dataset.head()
```

Out[3]:

	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
0	2020-01-01 00:00:00 UTC	view	5809910	1602943681873052386	NaN	grattol	5.24	595414620	4adb70bb-edbd-4981-b60f-a05bfd32683a
1	2020-01-01 00:00:09 UTC	view	5812943	1487580012121948301	NaN	kinetics	3.97	595414640	c8c5205d-be43-4f1d-aa56-4828b8151c8a
2	2020-01-01 00:00:19 UTC	view	5798924	1783999068867920626	NaN	zinger	3.97	595412617	46a5010f-bd69-4fbe-a00d-bb17aa7b46f3
3	2020-01-01 00:00:24 UTC	view	5793052	1487580005754995573	NaN	NaN	4.92	420652863	546f6af3-a517-4752-a98b-80c4c5860711
4	2020-01-01 00:00:25 UTC	view	5899926	2115334439910245200	NaN	NaN	3.92	484071203	cff70ddf-529e-4b0c-a4fc-f43a749c0acb

File structure

1. event_time – when the event happened (in UTC).
2. event_type – one of the following:
   - view - a user viewed a product
   - cart - a user added a product to shopping cart
   - removefromcart - a user removed a product from shopping cart
   - purchase - a user purchased a product

   Example of a typical funnel: view → cart → purchase.
3. product_id – unique identity of the product.
4. category_id – unique identity of the category of the product.
5. category_code – product's category taxonomy.
6. brand – name of the brand of the product.
7. price – price of the product (float).
8. user_id – unique identity of the user.
9. user_session – changes every time user comes back to online store after a long pause.

JSON + Python + Pandas – Example

- Download the JSON dataset from: https://sapienza2020adm.s3.eu-central-1.amazonaws.com/2020-Jan.zip


```
{
   "event_time":"2020-01-01 00:00:00 UTC",
   "event_type":"view",
   "product_id":5809910,
   "category_id":1602943681873052386,
   "category_code":"",
   "brand":"grattol",
   "price":5.24,
   "user_id":595414620,
   "user_session":"4adb70bb-edbd-4981-b60f-a05bfd32683a"
}
```


JSON + Python + Pandas – Example

```
import pandas as pd

dataset = pd.read_json('2020-Jan.json',
                       lines=True, nrows=20)
```

Check out the manual page for details on the different parameters used:

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_json.html


```
In [1]: import pandas as pd

In [2]: dataset = pd.read_json('2020-Jan.json', lines=True,
                               nrows=20)

In [3]: dataset.head()
```

Out[3]:

	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_sessio
0	2020-01-01 00:00:00+00:00	view	5809910	1602943681873052386		grattol	5.24	595414620	4adb70bb edbd-4981-b60 a05bfd32683
1	2020-01-01 00:00:09+00:00	view	5812943	1487580012121948301		kinetics	3.97	595414640	c8c5205c be43-4f1c aa56-4828b8151c8
2	2020-01-01 00:00:19+00:00	view	5798924	1783999068867920626		zinger	3.97	595412617	46a5010 bd69-4fbe-a00c bb17aa7b46f
3	2020-01-01 00:00:24+00:00	view	5793052	1487580005754995573			4.92	420652863	546f6af3 a517-4752 a98b-80c4c586071
4	2020-01-01 00:00:25+00:00	view	5899926	2115334439910245200			3.92	484071203	cff70ddf-529e-4b0c a4fc-f43a749c0ac