Algorithmic Methods of Data Mining

Computational Thinking, Basic Tools and First Practice

Ioannis Chatzigiannakis

Sapienza University of Rome

Laboratory 1

Computational Thinking

Wing, J. M. 2006 Computational thinking. CACM 49, 33–35

Computational thinking is taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamental to computing.

Wing, J. M. 2006 Computational thinking. CACM 49, 33–35

Computational thinking represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Wing, J. M. 2006 Computational thinking. CACM 49, 33–35

Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction.

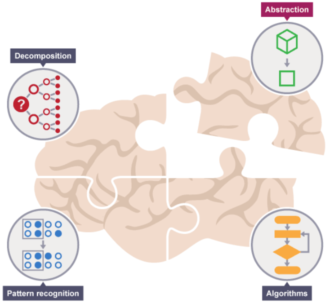
The riddle of machine intelligence

Computational thinking confronts the riddle of machine intelligence:

- What can humans do better than computers?
- What can computers do better than humans?
- What is computable?

Computational Thinking

- Computers are here to help us.
- What do we need from computers?
- What is our problem?
- Computational Thinking allows us to understand what needs to be solved.
- Four key techniques (cornerstones) to computational thinking:
  1. Decomposition – breaking down a complex problem or system into smaller, more manageable parts
  2. Pattern Recognition – looking for similarities among and within problems
  3. Abstraction – focusing on the important information only, ignoring irrelevant detail
  4. Algorithms – developing a step-by-step solution to the problem, or the rules to follow to solve the problem



Computational Thinking vs Programming

Thinking computationally is not programming.

- . . . not even thinking as a computer.
- Programming tells computer what to do / how to do it.
- Computational thinking enables us to understand what we need to tell to computers.
- . . . what to program.

Examples:

- Explain to a friend how to drive to your house
- Organize a party at the park
- Prepare your luggage
- Teach a kid addition/subtraction
- . . .

Decomposition

Turn a complex problem into one we can easily understand.

- . . . probably you already do every day.
- The smaller parts are easier to solve.
- . . . we already know/have the solutions.

Examples:

- Brushing our teeth
  Which brush? How long? How hard? What toothpaste?
- Solving a crime
  What crime? When? Where? Evidence? Witnesses? Recent similar crimes?
- . . .

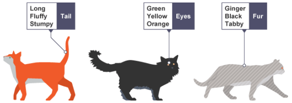
Pattern Recognition

We often find patterns among the smaller problems we examine.

- The patterns are similarities or characteristics that some of the problems share.

Example: Cats

- All cats share common characteristics.
  they all have eyes, tails and fur.
- Once we know how to describe one cat we can describe others, simply by following this pattern.



Abstraction

Hiding irrelevant details to focus on the essential features needed to understand and use a thing

- A compression process – multiple different pieces of constituent data to a single piece of abstract data.
  e.g., "cat"
- Ambiguity – multiple different references.
  e.g., "happiness", "architecture"
- Simplification – no loss of generality
  e.g., "red" - many different things can be red

Thought process wherein ideas are distanced from objects

Abstraction Example: Car vs Car Breaks

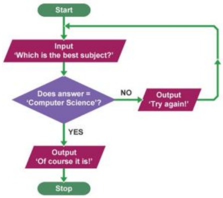
- Do we know how car breaks work?
- Do we know how to use them?

Filter out (ignore) the characteristics that we don't need in order to concentrate on those that we do.

Algorithms

A plan, a set of step-by-step instructions to solve a problem.

- In an algorithm, each instruction is identified and the order in which they should be carried out is planned.



Data Scientist's skill set

- Statistics, data analysis methods
  - Lots of data
  - High noise levels, missing values
  - #attributes $\gg$ #data points
- Programming languages
  - Scripting languages: Python, Perl, Ruby, ...
  - Extensive use of text file formats: need parsers
  - Integration of both data and tools
- Data structures, databases
  - Huge quantities of data need to be stored and indexed.
- Scientific computation packages
  - R, Matlab/Octave, ...
- Cloud computing
  - Amazon Web Services, Microsoft Azure, Google Cloud ...

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:

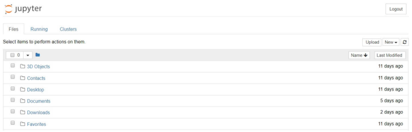
```
pip3 install jupyter
```

Execution:

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

```
jupyter notebook
```

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

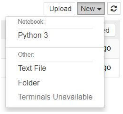


jupyter Logout

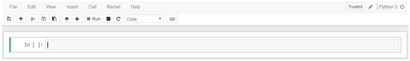
Files Running Clusters

Select items to perform actions on them.

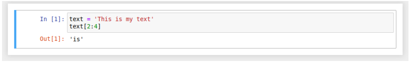
Name	Last Modified
3D Objects	11 days ago
Contacts	11 days ago
Desktop	11 days ago
Documents	5 days ago
Downloads	2 days ago
Favorites	11 days ago



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.



```
In [1]: text = 'This is my text'
        text[2:4]

Out[1]: 'is'
```

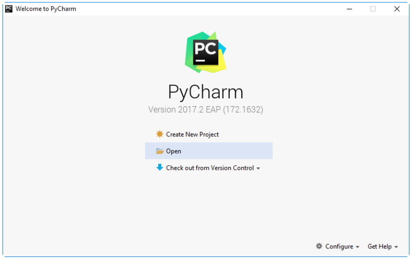
- Check out that In [ ]: has changed to In [1]:
- When python is processing the code we get In [*]:

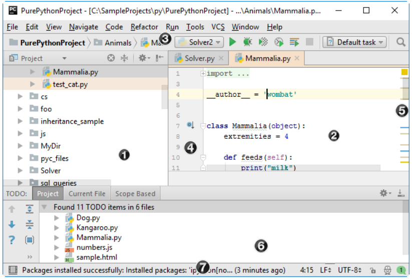
Navigating the Notebook with the Keyboard

- 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

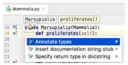




```
import ...

__author__ = 'wombat'


class Mammalia(object):
    extremities = 4

    def feeds(self):
        print("milk")
```

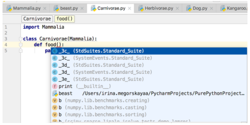
Packages installed successfully: Installed packages: 'ip...on[no... (3 minutes ago)   4:15   LF   UTF-8

Code with smart assistance



Mammalia.py
Marsupialia
proliferates()
Annotate types
Insert documentation string stub
Specify return type in docstring

```
class Marsupialia(Mammalia):
    def proliferates(self):
```

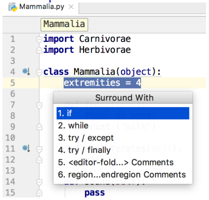


```
import Mammalia

class Carnivorae(Mammalia):
    def food():
        pa
```

```
_3c_  (StdSuites.Standard_Suite)
_3c_  (SystemEvents.Standard_Suite)
_3d_  (StdSuites.Standard_Suite)
_3d_  (SystemEvents.Standard_Suite)
_3e_  (StdSuites.Standard_Suite)
print  (__builtin__)
beast    /Users/irina.megorskayaa/PycharmProjects/PurePythonProject…
b  (numpy.lib.benchmarks.creating)
b  (numpy.lib.benchmarks.casting)
b  (numpy.lib.benchmarks.sorting)
```

- 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



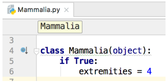
Mammalia.py

Mammalia

```
import Carnivorae
import Herbivorae

class Mammalia(object):
    extremities = 4
```

Surround With
1. if
2. while
3. try / except
4. try / finally
5. <editor-fold...> Comments
6. region...endregion Comments

```
        pass
```



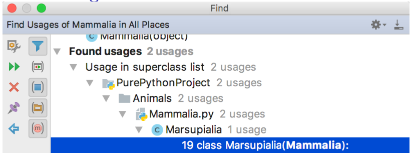
Mammalia.py

Mammalia

```
class Mammalia(object):
    if True:
        extremities = 4
```

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

Search for Usages



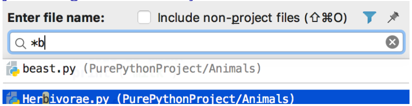
Find

Find Usages of Mammalia in All Places

Mammalia(object)
- Found usages 2 usages
  - Usage in superclass list 2 usages
    - PurePythonProject 2 usages
      - Animals 2 usages
        - Mammalia.py 2 usages
          - Marsupialia 1 usage
            - 19 class Marsupialia(**Mammalia**):

- 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



Enter file name: Include non-project files (⇧⌘O)

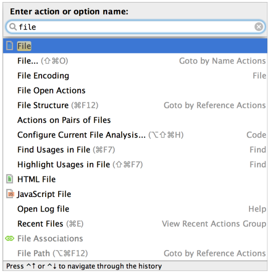
*b

beast.py (PurePythonProject/Animals)

Herbivorae.py (PurePythonProject/Animals)

- 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

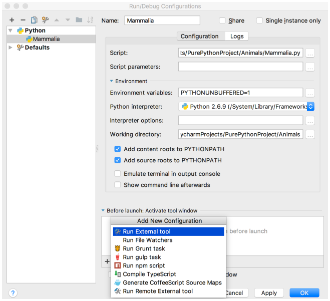


Enter action or option name:

file

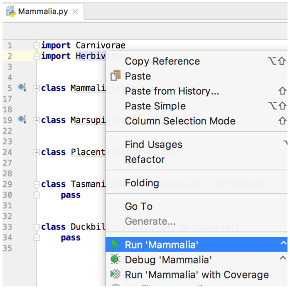
File	
File... (⇧⌘O)	Goto by Name Actions
File Encoding	File
File Open Actions	
File Structure (⌘F12)	Goto by Reference Actions
Actions on Pairs of Files	
Configure Current File Analysis... (⌥⇧⌘H)	Code
Find Usages in File (⌘F7)	Find
Highlight Usages in File (⇧⌘F7)	Find
HTML File	
JavaScript File	
Open Log file	Help
Recent Files (⌘E)	View Recent Actions Group
File Associations	
File Path (⌥⌘F12)	Goto by Reference Actions

Press ^↑ or ^↓ to navigate through the history



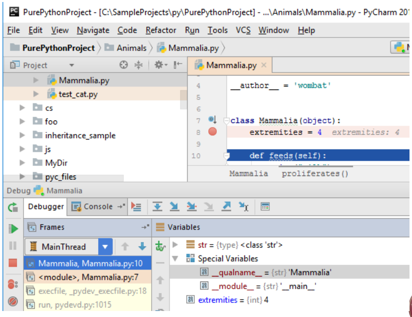
Run/Debug Configurations

Name: Mammalia
Script: ts/PurePythonProject/Animals/Mammalia.py
Script parameters:
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



Mammalia.py

```
import Carnivorae
import Herbiv
class Mammali
class Marsupi
class Placent
class Tasmani
    pass
class Duckbil
    pass
```

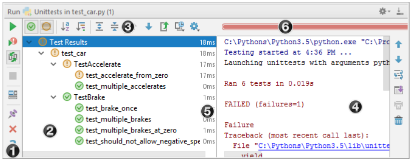


PurePythonProject - [C:\SampleProjects\py\PurePythonProject] - ...\Animals\Mammalia.py - PyCharm 20

```
__author__ = 'wombat'

class Mammalia(object):
    extremities = 4   extremities: 4

    def feeds(self):
```

Frames: MainThread
Mammalia, Mammalia.py:10
<module>, Mammalia.py:7
execfile, _pydev_execfile.py:18
run, pydevd.py:1015

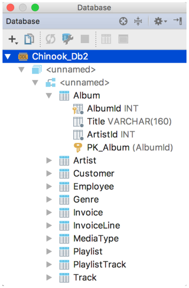
Variables:
str = {type} <class 'str'>
Special Variables
__qualname__ = {str} 'Mammalia'
__module__ = {str} '__main__'
extremities = {int} 4



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 0ms
test_multiple_brakes_at_zero 1ms
test_should_not_allow_negative_spe 0ms

```
C:\Pythons\Python3.5\python.exe "C:\Pro
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
    yield
```



Database

Database

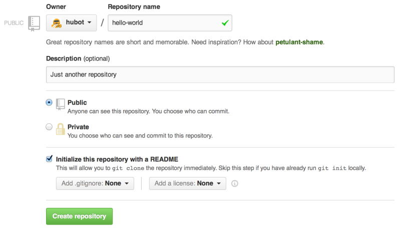
Chinook_Db2
- <unnamed>
  - <unnamed>
    - Album
      - AlbumId INT
      - Title VARCHAR(160)
      - ArtistId INT
      - PK_Album (AlbumId)
    - Artist
    - Customer
    - Employee
    - Genre
    - Invoice
    - InvoiceLine
    - MediaType
    - Playlist
    - PlaylistTrack
    - Track

- Code Hosting Platform
  - Version Control, Bug Tracking & Todo list, Wiki, Collaboration, . . .
- Public + Private Projects
- Cloud-based or Private Storage
- Alternatives:
  - BitBucket, SourceFourge, Team Foundation Server, SVN, CVS

First steps on Github

- Repository-oriented Family of Services
  - Repository: group of files relevant to a specific project.
  - Not necessarily related to coding.
- Each member of the project needs a separate account.
- Repositories are owned by an account.
  - Organizations are also allowed to own repositories.
- Repositories are created via the Website.
- Repositories can be browsed/modified via the Web or via broad range of client applications.

Creating a new Repository



PUBLIC

Owner: hubot / Repository name: hello-world

Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

Description (optional)

Just another repository

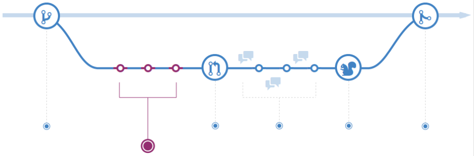
[✓ **Public**] Anyone can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

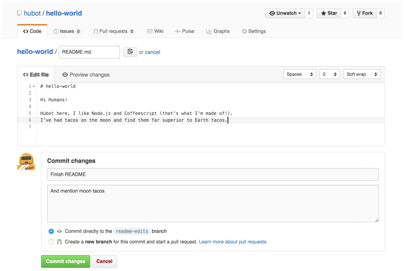
[✓ **Initialize this repository with a README**] This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None**

Make and commit changes



- Whenever you add, edit, delete.
- Keeps track of progress.
- Easy to roll-back to previous states.



```
# hello-world

Hi Humans!

Hubot here, I like Node.js and Coffeescript (that's what I'm made of!).
I've had tacos on the moon and find them far superior to Earth tacos.
```

Commit changes

Finish README

And mention moon tacos

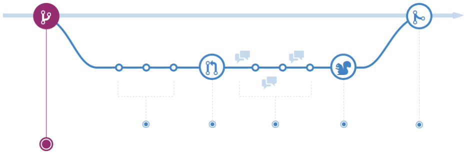
[✓ Commit directly to the `readme-edits` branch]

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

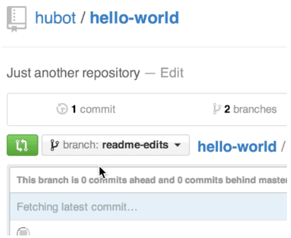
Real power of Github: Branching

- The most over-stressed functionality.
- Branching: work on different versions of a repository at one time.
- By default each repository has 1 branch: master
- When create a new branch off the master:
  - Make a copy of all contents.
  - Changes on new repository are separated.
  - Can pull changes from master at any point.
  - Can push changes to master at any point.

Branching



- Starting from the MASTER branch.
- We create the FEATURE branch.
- The new branch progresses independently.
- Eventually, it MERGES into MASTER.


hubot / hello-world

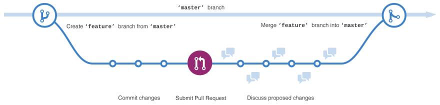
Just another repository — Edit

1 commit 2 branches

branch: readme-edits hello-world /

This branch is 0 commits ahead and 0 commits behind maste

Fetching latest commit…

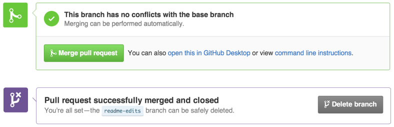


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

- Communicating changes to the other members of the team is done via PULL REQUESTS.
- Pull Requests are the heart of collaboration on GitHub.
- As soon as you make a commit:
  - open a pull request,
  - start a discussion!

Merge Pull Requests

- The final step of bringing changes together.
- Merging 2 brunches.
- After confirming the merge, other branches can be deleted.


This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Pull request successfully merged and closed
You're all set—the `readme-edits` branch can be safely deleted.

Introduction to AWS S3

- S3 = Simple Storage Service
  - From 0 bytes to 5 Tbytes.
- Provides a secure, durable, highly-scalable storage space.
  - AWS secures content with encryption, ACL and bucket policies.
  - AWS guarantees 99.999999999% durability (11 $\times$ 9s).
  - AWS guarantees 99.99% availability.
- We can access items stored:
  - Using the web.
  - Using the Web Console.
  - Using the Smartphone App.
  - From the Command line AWS tool.
  - Programmatically through the AWS S3 API.

S3 Basics

- Object-based storage.
  - Files = Objects.
  - Not suitable to install an operating system or host a database.
- Files/Objects are organized in Buckets.
- Bucket names must be unique – S3 is a universal namespace.
  - `http://sapienza2020adm.s3.amazonaws.com/`
  - When you create a new S3 bucket, AWS creates a new web address.
- Objects (Files) have the following properties:
  - Key: the name of the object.
  - Value: the actual contents.
  - Version ID: used by the versioning system.
  - Metadata: tags that we can attach to objects.
  - ACL: who can access the object.

S3 Storage Classes

- Free Tier – new AWS accounts
  - 5GB of S3 storage.
  - 20,000 GET – 2,000 PUT/COPY/POST/LIST
  - 15GB of Data Transfer Out each month for one year
- S3 Standard
  - \$0.0245 per GB
  - \$0.0054 per 1000 PUT/COPY/POST/LIST
  - \$0.00043 per 1000 GET/SELECT/all other requests.
- S3-IA Infrequent Access
  - \$0.0135 per GB – a minimum storage duration of 30 days.
  - \$0.01 per 1000 PUT/COPY/POST/LIST
  - \$0.001 per 1000 GET/SELECT/all other requests.
- S3 Glacier
  - \$0.0045 per GB – a minimum storage duration of 90 days.
  - \$0.06 per 1000 PUT/COPY/POST/LIST
  - \$0.00043 per 1000 GET/SELECT/all other requests.

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.

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.

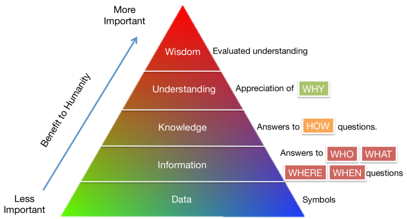
Why do we need open data?

- 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:

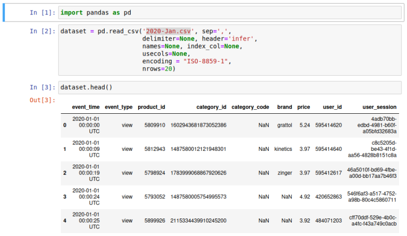
```
pip3 install pandas
```

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

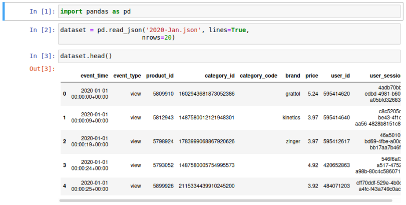
```
{
  "event_time":"2020-01-01 00:00:00 UTC",
  "event_type":"view",
  "product_id":5809910,
  "category_id":1602943681873052386,
  "category_code":"",
  "brand":"grattol",
  "price":5.24,
  "user_id":595414620,
  "user_session":"4adb70bb-edbd-4981-b60f-a05bfd32683a"
}
```

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