
Tools for Management Data Science Course

Master in Big Data, Analytics and
Technology Management

University of Firenze, 2016/2017

Andrea Gigli, PhD Applied Statistics, MSc Big Data & Social Mining

Goals

Congrats, you enrolled in the **Master in Big Data, Analytics and Technology Management @ University of Firenze**, a Master organized by the Departments of Statistics & Computer Science, Engineering and Economics.

I'm Andrea Gigli, PhD in Applied Statistics, MSc in Big Data Analytics & Social Mining. During the Data Science for Management course I'll guide you through Data Science applications for Business.

In these **30 hours Front End lectures, 90 hours Labs, 60 hours Industrial Workshops** you'll understand selected business problems and learn how to use Data Science to solve them, while working on real cases and learning methodologies and techniques during your project works.

Goals

Being in touch with real companies, I've designed this course in order to make your CV appealing for them. This year you'll learn

- R and Python scripting (R & Python)
- How to explore business data (R)
- How to build business dashboards using quantitative, qualitative and geo-referenced data (R)
- How to scrape web pages for business intelligence purposes (Python)
- How to measure customer engagement on Social Networks like facebook or twitter (Python)
- How to use data mining in your customer database (Python & R)
- How to segment and cluster customers (R)

Moreover, during the workshops you'll make practice with other problems and interact with true industry professionals to understand how to solve them with business in mind...

Time to work now...

Before coding...

I've collected some notes here on tools we'll use extensively during course, labs and seminars. You can see how to

1. install R and R Studio
2. install Python 3 and Spyder
3. install Jupyter in Linux Virtual Machine
4. get an access token for Facebook mining and how to use it
5. get an access token for Twitter mining
6. install Scrapy for Python 3.0
7. Xpath basics
8. GitHub basics

Jupyter Notebook

To install IPython Notebook follow these steps

1. Install jupyter for Python 3.X in your Linux

```
$ sudo pip3 install jupyter (enter your password)
```

Now your Jupyter has Python (3) kernel. For using Jupyter with R you need to

2. Install IRKernel in R

```
install.packages(c('repr', 'IRdisplay', 'crayon', 'pbdZMQ',  
'devtools'))
```

```
devtools::install_github('IRkernel/IRkernel')
```

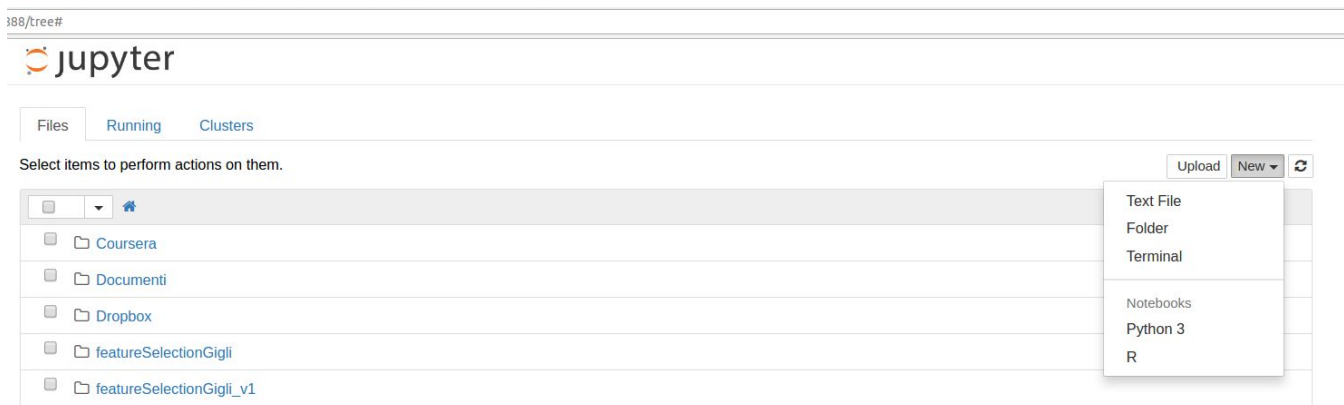
```
IRkernel::installspec() # to register the kernel in the current  
R installation
```

Jupyter Notebook

To launch Jupyter from the Linux command window, type

3. `$ jupyter notebook`

You'll see you browser opening a new session. If you click on “New” you can create a new Python or R notebook



Facebook API

To get access to Facebook API you need a Developer Account

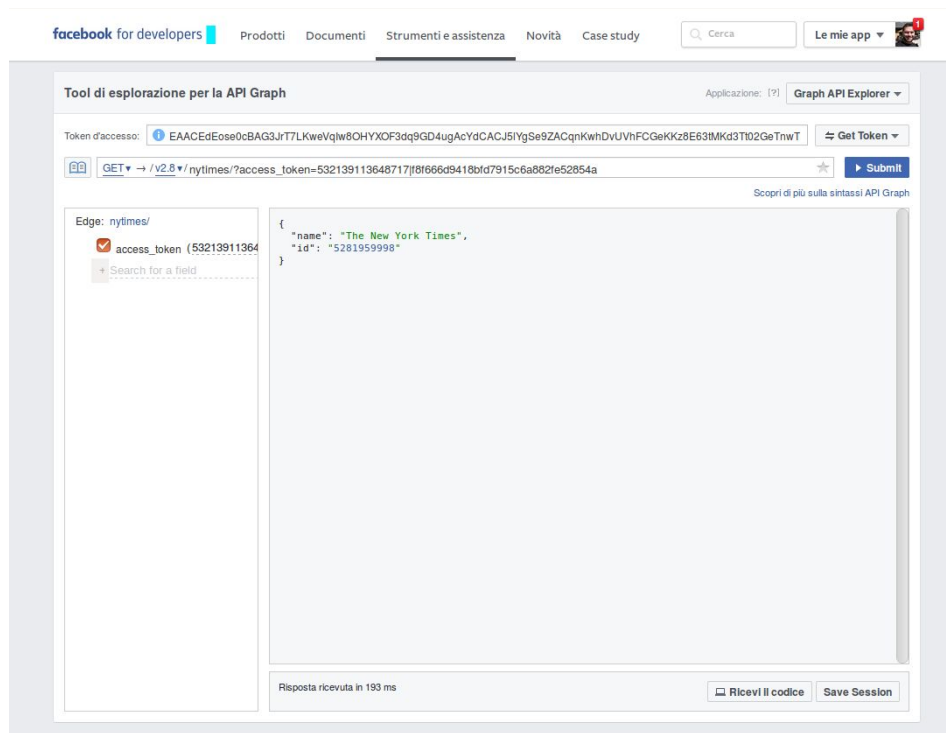
1. Login to Facebook
2. Register to <https://developers.facebook.com>
3. Create a FB dev account and get a temporary (2 hours) token. You can also build a never expiring token by combining
 - app id: 123456789abcdef
 - secret key: f8f666d9418bfd7915c6a882fe52854aand separating them with a “pipe”, “|”
 - never expiring token:
123456789abcdef|f8f666d9418bfd7915c6a882fe52854a

You can find a very useful tutorial @ <https://developers.facebook.com/apps>

Facebook API

To test your access token type

`nytimes/?access_token =
<YOUR ACCESS TOKEN>`



The screenshot shows the Facebook Graph API Explorer interface. At the top, there's a navigation bar with links for 'facebook for developers', 'Prodotti', 'Documenti', 'Strumenti e assistenza', 'Novità', and 'Case study'. A search bar and a 'Le mie app' button are also present. The main area is titled 'Tool di esplorazione per la API Graph'. It shows the 'Token d'accesso' as 'EAACEdEose0cBAG3Jr7TLKweVqIw8OHYXOF3dq9GD4ugAcYdCACJ5iYgSe9ZACqnKwhDvUvHFCGeKKz8E63IMKd3Tt02GeTnwT'. The URL bar shows a GET request to '/v2.8/nytimes/?access_token=532139113648717f8f666d9418bfd7915c6a882e52854a'. The response is a JSON object:

```
{  "name": "The New York Times",  "id": "5281959998"}
```

. The status at the bottom indicates 'Risposta ricevuta in 193 ms'. There are buttons for 'Ricevi il codice' and 'Save Session'.

Facebook API: messages

facebook for developers | Prodotti | Documenti | Strumenti e assistenza | Novità | Case study | Cerca | Le mie app

Tool di esplorazione per la API Graph | Applicazione: [?] Graph API Explorer

Token d'accesso: EAACEdEose0cBAG3Jr77LKwVqIw8OHYXOF3dq9GD4ugAcYdACAJ5IYgSe9ZACqNkWhDVUvHFCGeKKz8E63MKd3Tt02GeTnwT | Get Token

GET → /v2.8/nytimes/feed/?fields=message,link,created_time,type,name,id,likes.limit(1),summary(true),comments.limit(1),summary(tru | Submit

Scopri di più sulla sintassi API Graph

Edge: nytimes/feed/

- ☒ access_token (53213911364)
- ☒ limit (1)
- ☒ message
- ☒ link
- ☒ created_time
- ☒ type
- ☒ name
- ☒ id
- ☒ likes
 - ☒ limit (1)
- ☒ comments
 - ☒ limit (1)
- ☒ shares
 - ☐ Search for a field

1 Debug Message (Show)

```
{
  "data": [
    {
      "message": "It will go by faster than you think.",
      "link": "http://www.nytimes.com/2016/11/26/travel/how-to-plan-an-extended-vacation.html?smid=fb",
      "created_time": "2016-11-26T07:25:01+0000",
      "type": "link",
      "name": "How to Plan an Extended Vacation",
      "id": "5281959998_10150975778669999",
      "shares": {
        "count": 10
      },
      "likes": {
        "data": [
          {
            "id": "921997127815912",
            "name": "Birute Joksaite"
          }
        ],
        "paging": {
          "cursors": {
            "before": "OTIxOTk3MTI3ODU1OTYy",
            "after": "OTIxOTk3MTI3ODU1OTYy"
          },
          "next": "https://graph.facebook.com/v2.8/5281959998_10150975778669999/likes?access_token=EA"
        },
        "summary": {
          "total_count": 39,
          "can_like": true,
          "has_liked": false
        }
      },
      "comments": {
        "data": [
        ]
      }
    }
  ]
}
```

Risposta ricevuta in 246 ms | Ricevi il codice | Save Session

```
nytimes/feed/?fields=message,link,created_time,type,name,id,likes.limit(1).summary(true),comments.limit(1).summary(true),shares&limit=1&access_token=<YOUR ACCESS TOKEN>
```

<https://www.facebook.com/nytimes/posts/10150975887169999>

The screenshot shows a Facebook post from The New York Times. The post is titled "Dylan Devenyi This is the same political party that was caught actively seeking to suppress African American votes, who cut sunday voting to reduce 'souls to the polls' turnout among African American voters, and who lost a lawsuit to the NAACP after illegally striking African American voters from the rolls, and now they dare to allege there was some unfairness against them, or that 'they're' entitled to this race? Please, governor McCrory, retire from public life. North Carolina - not America - want nothing more from you." The post has 1,800 likes and 173 comments. Red circles highlight the "Mi piace" button, the like count, and the comment count. The post is part of a thread of comments, with the first comment by Dylan Devenyi and the second by Leslie Daniel-Jack. The post is also part of a video series, with the first video titled "Will Serena Williams Complete the Grand Slam?" and the second titled "What We're Reading: Great Reads From Dean B...".

The New York Times
@nytimes

Home
Informazioni
Foto
Video
Persone a cui piace
Instagram
Pinterest
Eventi
Note
Post
Crea una Pagina

Mi piace Invia messaggio Condividi Altro

Mi piace 1,8 mila Commenti più in vista

230 condivisioni 173 commenti

Scrivi un commento...

Dylan Devenyi This is the same political party that was caught actively seeking to suppress African American votes, who cut sunday voting to reduce "souls to the polls" turnout among African American voters, and who lost a lawsuit to the NAACP after illegally striking African American voters from the rolls, and now they dare to allege there was some unfairness against them, or that "they're" entitled to this race? Please, governor McCrory, retire from public life. North Carolina - not America - want nothing more from you.
Mi piace · Rispondi · 694 · 20 novembre alle ore 9:57

Nascondi 14 risposte

Emily P. Justingham So you are demanding church mix w/ state- in order to influence voters based on their religious beliefs? I think you need to read the Constitution!!!!
Mi piace · Rispondi · 20 novembre alle ore 12:39

Dylan Devenyi Yeah no churches can't tell people who to vote for but they can encourage people to vote. "Souls to the polls" is perfectly legal. If it weren't, they wouldn't need to have gotten rid of Sunday voting. The "separation of church and state," which is a ... Altro...
Mi piace · Rispondi · 94 · 20 novembre alle ore 12:46

Leslie Daniel-Jack Emily P. Justingham, I think you are too late with that indignation. Republicans have been using the pulpit for decades to influence voters.
Mi piace · Rispondi · 51 · 20 novembre alle ore 12:48

Michelle Sprawls Tadayon And it's actually 7902 that Cooper is in the lead by....
Mi piace · Rispondi · 16 · 20 novembre alle ore 13:03

Lance Link Patricia Conder, Have you ever visited any of the ...
Mi piace · Rispondi · 16 · 20 novembre alle ore 13:03

Guarda il video

1,3 mila 67

263 42 1,1 mila 173

APPLICAZIONI

Instagram

Pinterest

NOTE

Will Serena Williams Complete the Grand Slam?
28 agosto 2015

What We're Reading: Great Reads From Dean B...

Facebook API: reactions

```
5281959998_10150975887169999/?field
s=reactions.type(LIKE).summary(tota
l_count).limit(0).as(like),reaction
s.type(LOVE).summary(total_count).l
imit(0).as(love),reactions.type(WOW
).summary(total_count).limit(0).as(
wow),reactions.type(HAHA).summary(t
otal_count).limit(0).as(haha),react
ions.type(SAD).summary(total_count)
.limit(0).as(sad),reactions.type(AN
GRY).summary(total_count).limit(0).
as(angry)&access_token=<YOUR ACCESS
TOKEN>
```

The screenshot shows the Facebook Graph API Explorer interface. At the top, there's a navigation bar with 'facebook for developers' and links to 'Prodotti', 'Documenti', 'Strumenti e assistenza', 'Novità', and 'Case study'. A search bar and a 'Le mie app' button are also present. The main area is titled 'Tool di esplorazione per la API Graph'. It shows a 'Token d'accesso' field with a long token and a 'Get Token' button. Below that, a query is entered in the 'GET' tab: `/v2.8/5281959998_10150975887169999/?fields=reactions.type(LIKE).summary(total_count).limit(0).as(like),reactions.type(LOVE).summary(total_count).limit(0).as(love),reactions.type(WOW).summary(total_count).limit(0).as(wow),reactions.type(HAHA).summary(total_count).limit(0).as(haha),reactions.type(SAD).summary(total_count).limit(0).as(sad),reactions.type(ANGRY).summary(total_count).limit(0).as(angry)&access_token=EAACEdEose0cBAST4EE7jvIFmeWDbY24y66omuxzmXm3ZBDriZEVATZADZBCdeRWe41asdwnkzC6D2W5pNzQZAUspgnZBUwh`. The response is shown in the 'JSON' tab, displaying a nested JSON structure with counts for each reaction type. At the bottom, there's a 'Ricevi il codice' button and a 'Save Session' button.

facebook for developers | Prodotti | Documenti | Strumenti e assistenza | Novità | Case study

Cerca | Le mie app

Tool di esplorazione per la API Graph | Applicazione: [?] | Graph API Explorer

Token d'accesso: EAACEdEose0cBAST4EE7jvIFmeWDbY24y66omuxzmXm3ZBDriZEVATZADZBCdeRWe41asdwnkzC6D2W5pNzQZAUspgnZBUwh | Get Token

GET → /v2.8/5281959998_10150975887169999/?fields=reactions.type(LIKE).summary(total_count).limit(0).as(like),reactions.type(LOVE).summary(total_count).limit(0).as(love),reactions.type(WOW).summary(total_count).limit(0).as(wow),reactions.type(HAHA).summary(total_count).limit(0).as(haha),reactions.type(SAD).summary(total_count).limit(0).as(sad),reactions.type(ANGRY).summary(total_count).limit(0).as(angry)&access_token=EAACEdEose0cBAST4EE7jvIFmeWDbY24y66omuxzmXm3ZBDriZEVATZADZBCdeRWe41asdwnkzC6D2W5pNzQZAUspgnZBUwh | Submit

Scopri di più sulla sintassi API Graph

Edge: 5281959998_10150975887169999

Search for a field

```
{
  "like": {
    "data": [
    ],
    "summary": {
      "total_count": 432
    }
  },
  "love": {
    "data": [
    ],
    "summary": {
      "total_count": 23
    }
  },
  "wow": {
    "data": [
    ],
    "summary": {
      "total_count": 15
    }
  },
  "haha": {
    "data": [
    ],
    "summary": {
      "total_count": 11
    }
  },
  "sad": {
    "data": [
    ],
    "summary": {
      "total_count": 1
    }
  },
  "angry": {
    "data": [
    ],
    "summary": {
      "total_count": 0
    }
  }
}
```

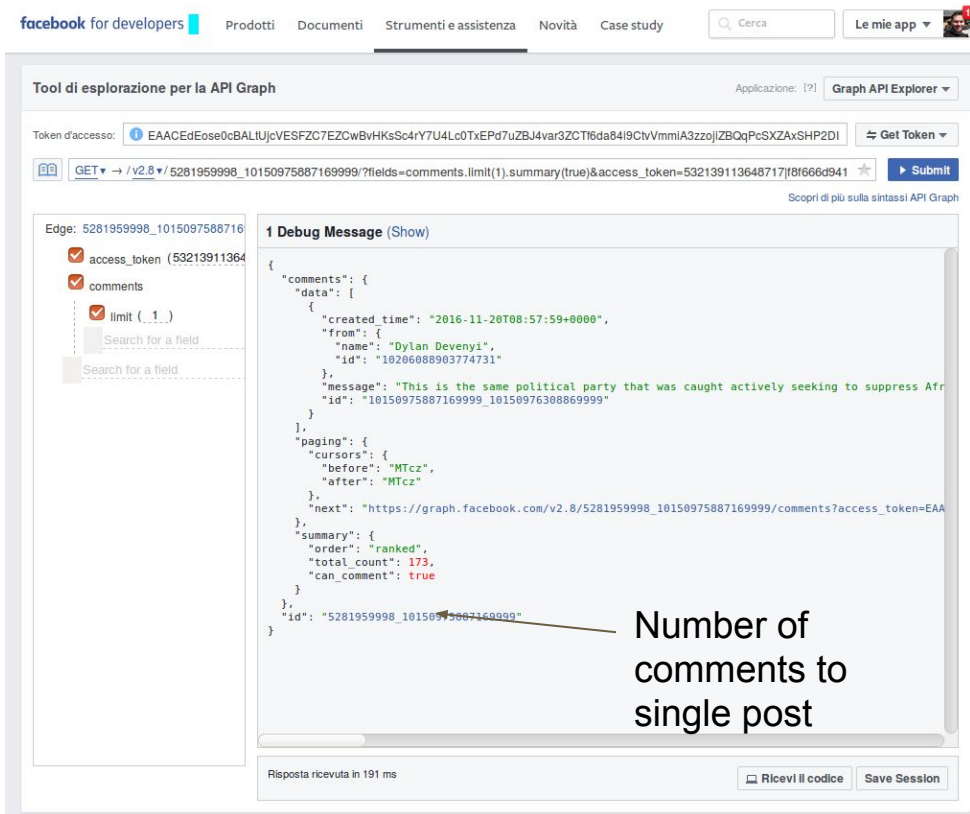
Risposta ricevuta in 184 ms | Ricevi il codice | Save Session

Facebook API: comments to single post

```
5281959998_10150975887169999/?fields=message,created_time,comments.limit(1),summary(true)&access_token=532139113648717|f8f666d9418bfd7915c6a882fe52854a
```

See also

<https://developers.facebook.com/docs/graph-api/reference/v2.8/comment/>



The screenshot shows the Facebook Graph API Explorer interface. The URL bar contains the following query: `GET → /v2.8/5281959998_10150975887169999/?fields=comments.limit(1),summary(true)&access_token=532139113648717|f8f666d9418bfd7915c6a882fe52854a`. The left sidebar shows the selected fields: `access_token`, `comments`, and `limit(1)`. The main area displays a JSON response under the heading "1 Debug Message (Show)". The JSON structure is as follows:

```
{
  "comments": {
    "data": [
      {
        "created_time": "2016-11-20T08:57:59+0000",
        "from": {
          "name": "Dylan Devenyi",
          "id": "10206088903774731"
        },
        "message": "This is the same political party that was caught actively seeking to suppress Afr",
        "id": "10150975887169999_10150976388869999"
      }
    ],
    "paging": {
      "cursors": {
        "before": "MTcz",
        "after": "MTcz"
      },
      "next": "https://graph.facebook.com/v2.8/5281959998_10150975887169999/comments?access_token=EAA"
    },
    "summary": {
      "order": "ranked",
      "total_count": 173,
      "can_comment": true
    }
  },
  "id": "5281959998_10150975887169999"
}
```

An arrow points from the text "Number of comments to single post" to the value "173" in the `total_count` field of the `summary` object.

At the bottom of the interface, it says "Risposta ricevuta in 191 ms" and there are buttons for "Ricevi il codice" and "Save Session".

Facebook API: comments counts to comments

```
5281959998_10150975887169999/?fields=shares,message,created_time,comments.limit(1){comment_count,like_count,message}&access_token=532139113648717|f8f666d9418bfd7915c6a882fe52854a
```

See also

<https://developers.facebook.com/docs/graph-api/reference/v2.8/comment/>

The screenshot shows the Facebook Graph API Explorer interface. The URL bar contains the query: `GET /v2.8/5281959998_10150975887169999/?fields=shares,message,created_time,comments.limit(1){comment_count,like_count,message}&access_token=532139113648717|f8f666d9418bfd7915c6a882fe52854a`. The left sidebar shows the selected fields: `access_token`, `shares`, `message`, `created_time`, `comments`, `limit(1)`, `comment_count`, `like_count`, and `message`. The main panel displays the JSON response, which includes the number of comments (16) and the text of the comment (1). A red arrow points from the text "Number of comments and likes to a comment" to the `comment_count` field in the JSON response.

facebook for developers | Prodotti | Documenti | Strumenti e assistenza | Novità | Case study | Cerca | Le mie app

Tool di esplorazione per la API Graph | Applicazione: [?] Graph API Explorer

Token d'accesso: EAACEdEose0cBAG2Dzq96IUVPkN7dRDGfIGRTmGd1KJ00IDrGvU7vCkCsNCfQhY7RwdVsdZBKIoS8ICZAAL5UIY19MJkb0m6o | Get Token

GET → /v2.8/5281959998_10150975887169999/?fields=shares,message,created_time,comments.limit(1){comment_count,like_count,message}&access_token=532139113648717|f8f666d9418bfd7915c6a882fe52854a | Invia

Scopri di più sulla sintassi API Graph

Edge: 5281959998_10150975887169999

- ☒ access_token (532139113648717|f8f666d9418bfd7915c6a882fe52854a)
- ☒ shares
- ☒ message
- ☒ created_time
- ☒ comments
- ☒ limit (1)
- ☒ comment_count
- ☒ like_count
- ☒ message

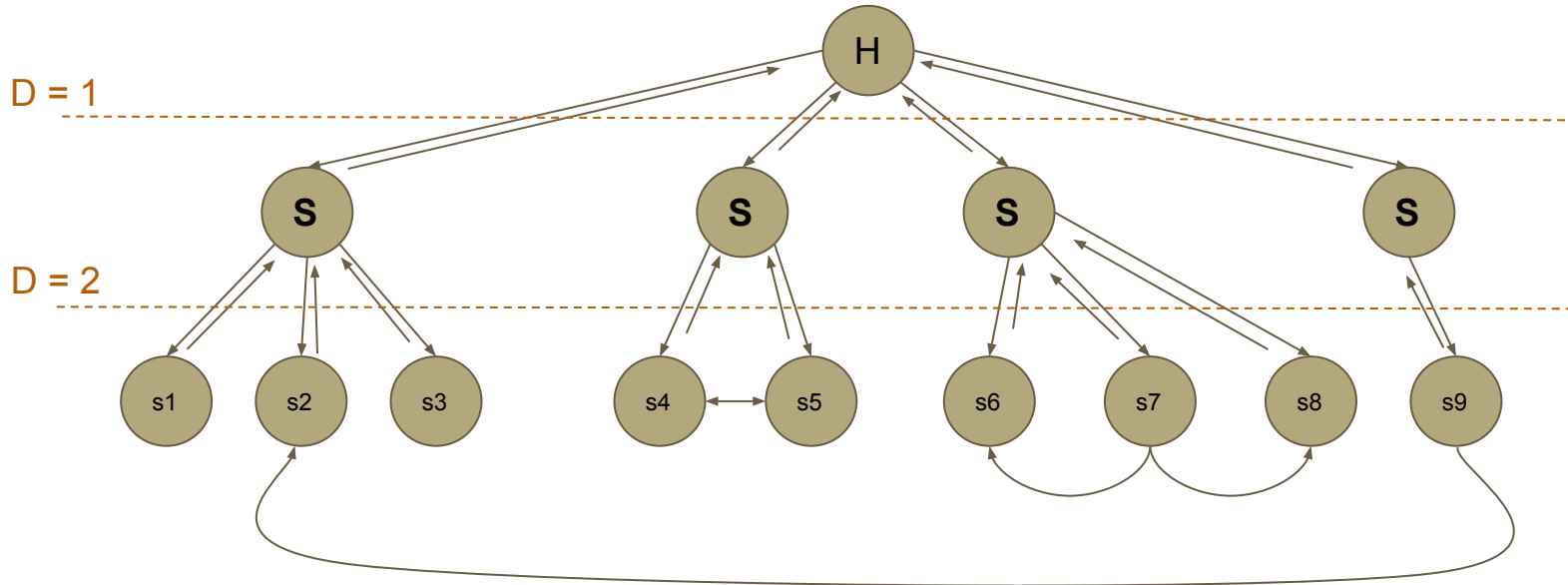
+ Cerca un campo

```
{
  "shares": {
    "count": 230
  },
  "message": "We're supposed to have an inauguration on Jan. 7. ... Are we going to have a governor?",
  "created_time": "2016-11-20T08:55:01+0000",
  "comments": {
    "data": [
      {
        "comment_count": 16,
        "like_count": 694,
        "message": "This is the same political party that was caught actively seeking to suppress Afr",
        "id": "10150975887169999_10150976308860999"
      }
    ],
    "paging": {
      "cursors": {
        "before": "NTcz",
        "after": "NTcz"
      },
      "next": "https://graph.facebook.com/v2.8/5281959998_10150975887169999/comments?access_token=EAA"
    }
  },
  "id": "5281959998_10150975887169999"
}
```

Risposta ricevuta in 202 ms | Ricevi il codice | Save Session

Scrapy

Scrapy is a Python framework for building **web spider**, i.e. robots which are able to visit all the web pages they can reach through links.



Scrapy

To install scrapy in Linux just type

```
sudo pip3 install scrapy
```

To be sure to have the most recent version type again

```
sudo pip3 install --upgrade scrapy
```

IMPORTANT! Sometimes Scrapy can generate conflicts when it is installed system-wide. We recommend to install it in a specific environment.

Look here >> <https://doc.scrapy.org/en/latest/intro/install.html#intro-using-virtualenv>

Alternatives to Scrapy

Scrapy works well but sometimes you simply need **parsing wisely web pages** (I say wisely...) or to interact with javascript of **web pages**.

In these cases you should look at

- BeautifulSoup (web page parsing Python library, [start from here](#))
- Selenium (tool for automating web application testing, [official Python doc here](#))

Xpath logic is the same.

XPath

Once you have downloaded the web pages you have to extract information from them.

Xpath is a syntax used to navigate through elements and attributes in an XML document.

XPath uses path expressions to select nodes or node-sets in an XML document.

These path expressions look very much like the expressions you use with a traditional computer file system ----->



XPath

In XPath, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document nodes.

XML documents are treated as trees of nodes. The topmost element of the tree is called the **root element** and **atomic values** are nodes with no children or parent.

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore> → Root node  
  <book>  
    <title lang="en">Harry Potter</title> → Attribute node  
    <author>J K. Rowling</author>  
    <year>2005</year>  
    <price>29.99</price> → Element node  
  </book>  
</bookstore>
```

For more details, please start with

http://www.w3schools.com/xml/xpath_intro.asp

and play with

<http://codebeautify.org/Xpath-Tester>

... and enjoy!

GitHub

Git is a **version control software**, which means it manages changes to a project without overwriting any part of that project.

This means you and your coworker(s) can each upload your revisions to the project (for example a web page, a paper, a program, ...) and Git will save more copies. Later, you can merge your changes together without losing any work along the way. You can even revert to an earlier version at any time, because Git keeps a “snapshot” of every change ever made.

Git was designed with a big project like Linux in mind, there are a lot of Git commands. However, to use the basics of Git, you'll only need to know a few terms. They all begin the same way, with the word “**git**”

GitHub: very frequent commands

git init	Initializes a new Git repository.
git clone	Clone a Git repository and create a local repository
git status	Check the status of your repository.
git pull	“pull” the changes down from GitHub to your local repository.
git add	This does <i>not</i> add new files to your repository. Instead, it brings new files to Git’s attention.
git commit	After you make any sort of change, you input this in order to take a “snapshot” of the repository. Usually it goes git commit -m “Message here”
git push	“push” the changes up to GitHub with this command.
git help	Forgot a command? Type this into the command line to bring up the 21 most common git commands.

Your First GitHub

- [Sign up for an account](#) on GitHub.com.
- [Install Git](#) on your computer
- Introduce yourself to Git. Open a terminal (Windows users have to start the Git Bash app you just installed) and type in the following code:

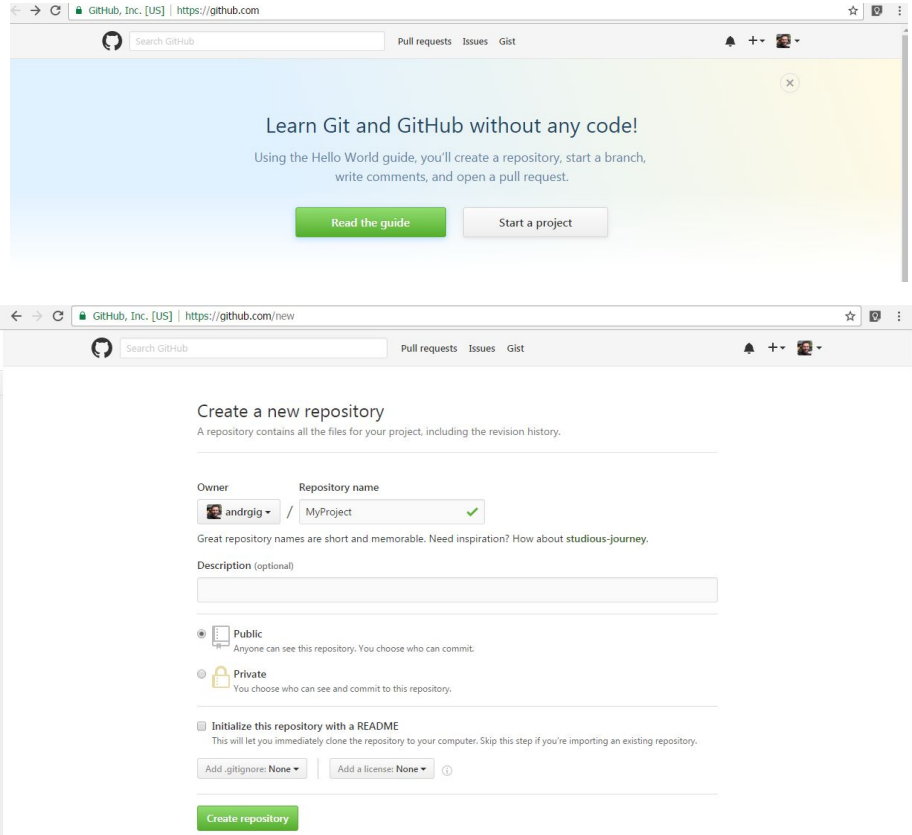
```
git config --global user.name "Your Name Here"
```

- tell it your email and make sure it's the same email you used when you signed up for a GitHub.com account

```
git config --global user.email "your_email@youremail.com"
```

Your First GitHub

- Go to GitHub.com and create a repository (a digital directory where you can access your project, its files, and all the versions of its files that Git saves) clicking on “Start a project”.
- Give your repository a short, memorable name and click the green “Create Repository” button
- You’re set. You now have an online space for your project to live in.



Your First GitHub

Mirror that online repository we just made as a local directory:

- First type:

```
mkdir ~/MyProject
```

(this is a general navigational command from the time before visual computer interfaces. The ~/ ensures that we're building the repository at the top level of your computer's file structure, instead of stuck inside some other directory that would be hard to find later)

- Then type:

```
cd ~/MyProject
```

(cd stands for change directory, and it's also a navigational command) Once we type this command, we are transported inside the local folder **MyProject**.

Your First GitHub

- For your next line, type:

```
git init
```

(You're using a Git command because it always begins with `git`. `init` stands for “initialize”, it tells the computer to recognize this directory as a local Git repository)

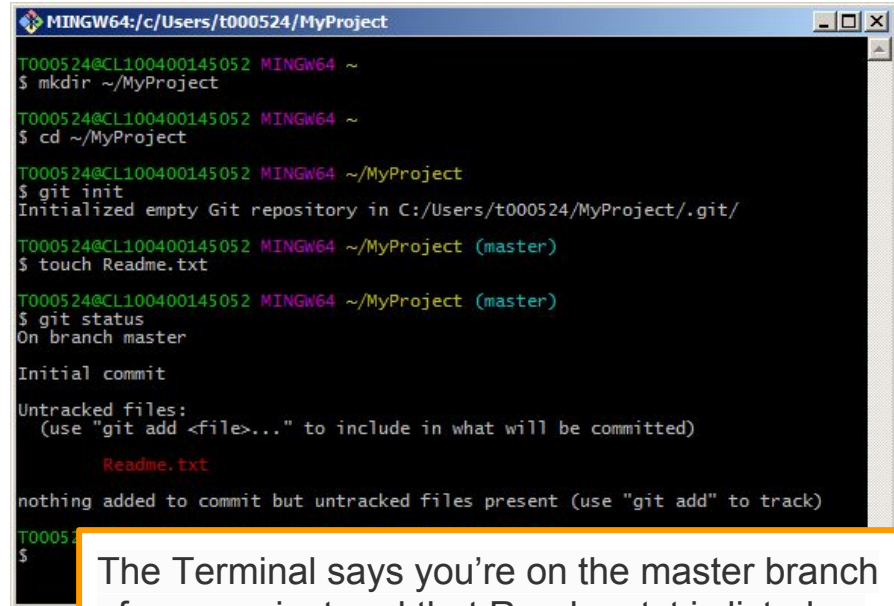
- Now create a simple txt file on the local MyProject folder

```
touch Readme.txt
```

(touch is a navigational command meaning “create”)

- To check if Git can see your file write

```
git status
```

A screenshot of a Windows Command Prompt window titled "MINGW64: c:/Users/t000524/MyProject". The window shows the following commands and output:

```
T000524@CL100400145052 MINGW64 ~
$ mkdir ~/MyProject
T000524@CL100400145052 MINGW64 ~
$ cd ~/MyProject
T000524@CL100400145052 MINGW64 ~/MyProject
$ git init
Initialized empty Git repository in C:/Users/t000524/MyProject/.git/
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ touch Readme.txt
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    Readme.txt

nothing added to commit but untracked files present (use "git add" to track)
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$
```

The Terminal says you're on the master branch of your project and that Readme.txt is listed as an “untracked” file, which means Git is ignoring it for now.

Your First GitHub

- To make Git notice that the file is there, type:

```
git add Readme.txt
```

- It's time to take a "snapshot" of the project so far, or "commit" it:

```
git commit -m "Add Readme.txt"
```

(The -m flag indicates that the following text should be read as a message.)

- Tell Git that a remote repository actually exists somewhere online

```
git remote add origin https://github.com/andrgig/MyProject.git
```

(remote is a descriptor of origin, to indicate the origin is not on the computer, but somewhere online)

- Git now knows there's a remote repository and it's where you want your local repository changes to go.
To confirm, type this to check:

```
git remote -v
```

```
MINGW64:/c/Users/t000524/MyProject
T000524@CL100400145052 MINGW64 ~
$ mkdir ~/MyProject
T000524@CL100400145052 MINGW64 ~
$ cd ~/MyProject
T000524@CL100400145052 MINGW64 ~/MyProject
$ git init
Initialized empty Git repository in C:/Users/t000524/MyProject/.git/
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ touch Readme.txt
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    Readme.txt

nothing added to commit but untracked files present (use "git add" to track)
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ git add Readme.txt
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ git commit -m "Add Readme.txt"
[master (root-commit) b5b3d2a] Add Readme.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Readme.txt
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ git remote add origin https://github.com/andrgig/MyProject.git
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ git remote -v
origin https://github.com/andrgig/MyProject.git (fetch)
origin https://github.com/andrgig/MyProject.git (push)
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$
```

Your First GitHub

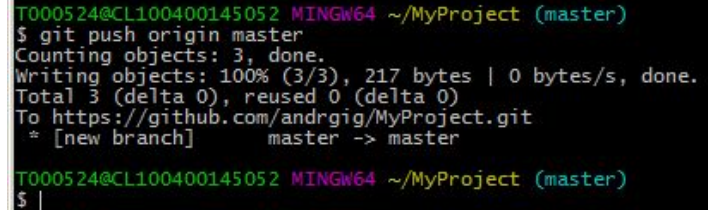
- Now we want to upload, or “push,” our changes up to the GitHub remote repo. That’s easy. Just type:

```
git push origin master
```

And insert your username and password for GitHub.

Now try again

```
cd ~/MyFirstGit
git init
git status
"Create a file <FileName>"
git add <FileName>
git commit -m "Add <FileName>"
git remote add origin https://github.com/<YourUsername>/myfirstgit.git
git remote -v
git push origin master
```

A terminal window with a black background and green text. The prompt is 'T000524@CL100400145052 MINGW64 ~/MyProject (master)'. The user enters '\$ git push origin master'. The output shows 'Counting objects: 3, done.', 'Writing objects: 100% (3/3), 217 bytes | 0 bytes/s, done.', 'Total 3 (delta 0), reused 0 (delta 0)', and 'To https://github.com/andrgig/MyProject.git'. A summary line shows '* [new branch] master -> master'. The prompt returns to '\$ |'.

```
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 217 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/andrgig/MyProject.git
 * [new branch]      master -> master
T000524@CL100400145052 MINGW64 ~/MyProject (master)
$ |
```

Git Commands

git init: Initializes a new Git repository. Until you run this command inside a repository or directory, it's just a regular folder. *Only after you input this does it accept further Git commands.*

git config: Short for “configure,” this is most useful when you're setting up Git for the first time.

git help: Forgot a command? Type this into the command line to bring up the 21 most common git commands. You can also be more specific and type “git help init” or another term to figure out how to use and configure a specific git command.

git status: Check the status of your repository. See which files are inside it, which changes still need to be committed, and which branch of the repository you're currently working on.

git add: This does *not* add new files to your repository. Instead, it brings new files to Git's attention. After you add files, they're included in Git's “snapshots” of the repository.

git branch: Working with multiple collaborators and want to make changes on your own? *This command will let you build a new branch, or timeline of commits, of changes and file additions that are completely your own.* Your title goes after the command. If you wanted a new branch called “cats,” you'd type **git branch cats**.

GitHub

git commit: Git's most important command. After you make any sort of change, you input this in order to take a “snapshot” of the repository. Usually it goes `git commit -m “Message here.”` The -m indicates that the following section of the command should be read as a message.

git checkout: Literally allows you to “check out” a repository that you are not currently inside. This is a navigational command that lets you move to the repository you want to check. You can use this command as `git checkout master` to look at the master branch, or `git checkout cats` to look at another branch.

git merge: When you're done working on a branch, you can *merge your changes back to the master branch*, which is visible to all collaborators. `git merge cats` would take all the changes you made to the “cats” branch and add them to the master.

git push: If you're working on your local computer, and *want your commits to be visible online on GitHub* as well, you “push” the changes up to GitHub with this command.

git pull: If you're working on your local computer and want *the most up-to-date version of your repository to work with*, you “pull” the changes down from GitHub with this command.

Regular Expression