CODEMOTION MILAN - SPECIAL EDITION
10 – 11 NOVEMBER 2017

{codemotion}

{Python Milano Meetup

Christian 'Strap' Strappazzon

Python Milano

Aperto a tutti, sviluppatrici, sviluppatori, appassionate e appassionati di Python

{codemotion}

Python Milano Meetup

Who We Are

What we do,
why, how, when, where...

Carlo Miron @cm
Cesare Placanica @keobox
Christian Strappazzon @cstrap
Flavio Percoco @flaper87
Lorenzo Mele @greenkey
Pietro Brunetti @gunzapper
Simone Picciani @zanza00

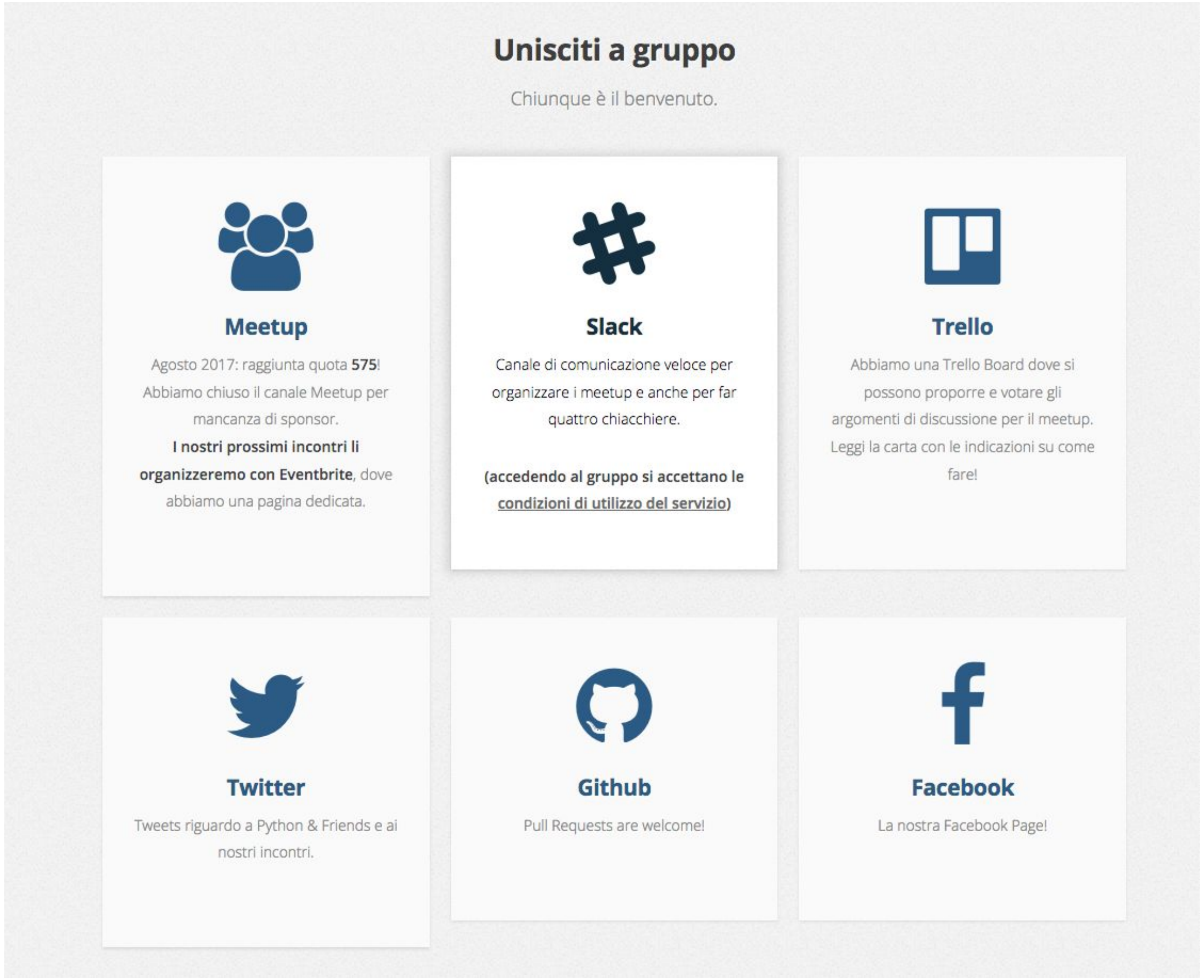
http://milano.python.it/

Slack

http://pythonmilano.herokuapp.com/

Join us!

Python Milano Meetup



Unisciti a gruppo

Chiunque è il benvenuto.

Meetup

Agosto 2017: raggiunta quota **575**! Abbiamo chiuso il canale Meetup per mancanza di sponsor. **I nostri prossimi incontri li organizzeremo con Eventbrite**, dove abbiamo una pagina dedicata.

Slack

Canale di comunicazione veloce per organizzare i meetup e anche per far quattro chiacchiere.

(accedendo al gruppo si accettano le condizioni di utilizzo del servizio)

Trello

Abbiamo una Trello Board dove si possono proporre e votare gli argomenti di discussione per il meetup. Leggi la carta con le indicazioni su come fare!

Twitter

Tweets riguardo a Python & Friends e ai nostri incontri.

Github

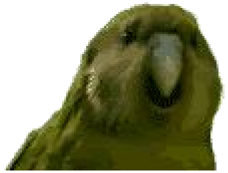
Pull Requests are welcome!

Facebook

La nostra Facebook Page!

Python Milano Meetup

Python Milano Meetup



Let’s Start!

http://cultofthepartyparrot.com/

Python Milano Meetup

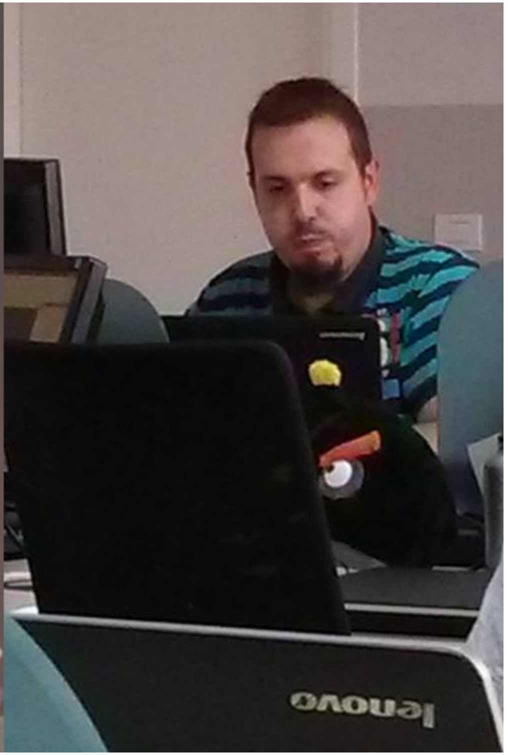
Python Blueprint

The tools, the packages and the ecosystem

{codemotion}

Python Milano Meetup

I'm not the dev every team needs, but I'm the dev every team deserves.
Perhaps I'm not the most productive programmer out there, but you sure contribute to increasing the team's overall productivity (I'm likely still more productive than most though).
I'm often learning about new things, asking the right questions and getting everyone up to speed.
I'm usually the person junior devs look up to and everyone likes working with.
Any team is lucky having me!
Last but not least I'm dad and family addicted.



Python Milano Meetup

Pay attention! I’m not the guy that play the guitar!

Agenda

- Which Python?
- pip
- virtualenv and virtualenvwrapper
- package patterns
- cookiecutter
- bumpversion & punch
- PyPI
- Linting, coverage and testing frameworks
- IDEs

Which Python?

Python 2 or Python 3?
Python 2 ending support on 2020.

Install different interpreters:

- https://github.com/yyuu/pyenv lets you easily switch between multiple versions of Python. It's simple, unobtrusive, and follows the UNIX tradition of single-purpose tools that do one thing well.

Compatibility solutions:

- http://python-future.org is the missing compatibility layer between Python 2 and Python 3. It allows you to use a single, clean Python 3.x-compatible codebase to support both Python 2 and Python 3 with minimal overhead.
- `six` provides simple utilities for wrapping over differences between Python 2 and Python 3. It is intended to support codebases that work on both Python 2 and 3 without modification. six consists of only one Python file, so it is painless to copy into a project. https://pythonhosted.org/six/

pip

`pip` is a tool for installing and managing Python packages, such as those found in the Python Package Index (PyPI).

`pip` is already installed if you're using Python 2 >=2.7.9 or Python 3 >=3.4 downloaded from python.org, but you'll need to upgrade pip.

https://pip.pypa.io/en/stable/

```
$ pip install SomePackage

$ pip install --upgrade package-name

$ pip uninstall package-name

$ pip search "query" # yolk or pipdeptree are two
packages to view dependencies

$ pip freeze > requirements.txt

$ pip install -r requirements.txt
```

```
pip install /path/to/SomePackage.1.1.1.tar.gz

pip install http://myrepo.it/SomePakage-1.4.0.zip

pip install -e \
     git+http://github.com/django/django.git#egg=
django

pip install -e path/to/SomePackage

pip install django>=1.7 # ==, >=, >, <, <=
```

Python Milano Meetup

Pipenv: Python Development Workflow for Humans

The officially recommended Python packaging tool from Python.org, free (as in freedom).

Pipenv is a tool that aims to bring the best of all packaging worlds (bundler, composer, npm, cargo, yarn, etc.) to the Python world. Windows is a first–class citizen, in our world.

It automatically creates and manages a virtualenv for your projects, as well as adds/removes packages from your Pipfile as you install/uninstall packages.

It also generates the ever–important Pipfile.lock, which is used to produce deterministic builds.

https://github.com/kennethreitz/pipenv

virtualenv

`virtualenv` is a tool to create isolated Python environments. Every virtualenv has pip installed in it automatically. Does not require root access or modify your system.

Why?

Isolation - Python packages and even version live in their own space
Permissions - No sudoers, the environment is mine!
Organization - each project can maintain its own requirements file of Python packages
No-Globalization - don't require installing stuff globally on the system.

https://virtualenv.readthedocs.org/en/latest/

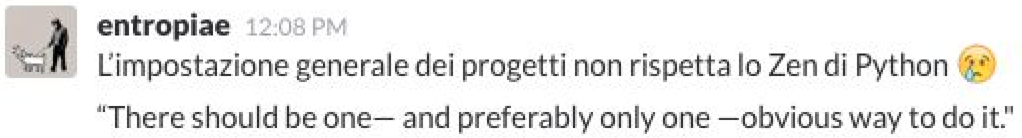
```
$ virtualenv ENV
     # This creates a folder ENV in the $PWD
     # You'll find python packages on
     #    ENV/lib/pythonX.X/site-packages

$ virtualenv ENV --python=/path/to/bin/python
     # Doesn't inherit global site-packages
     # Use a different Python interpreter

$ cd ENV ; . bin/activate
(ENV) $ pip install django
 Downloading/unpacking django
     ...
```

`virtualenvwrapper` is is a set of extensions to `virtualenv` tool.
https://virtualenvwrapper.readthedocs.org/en/latest/

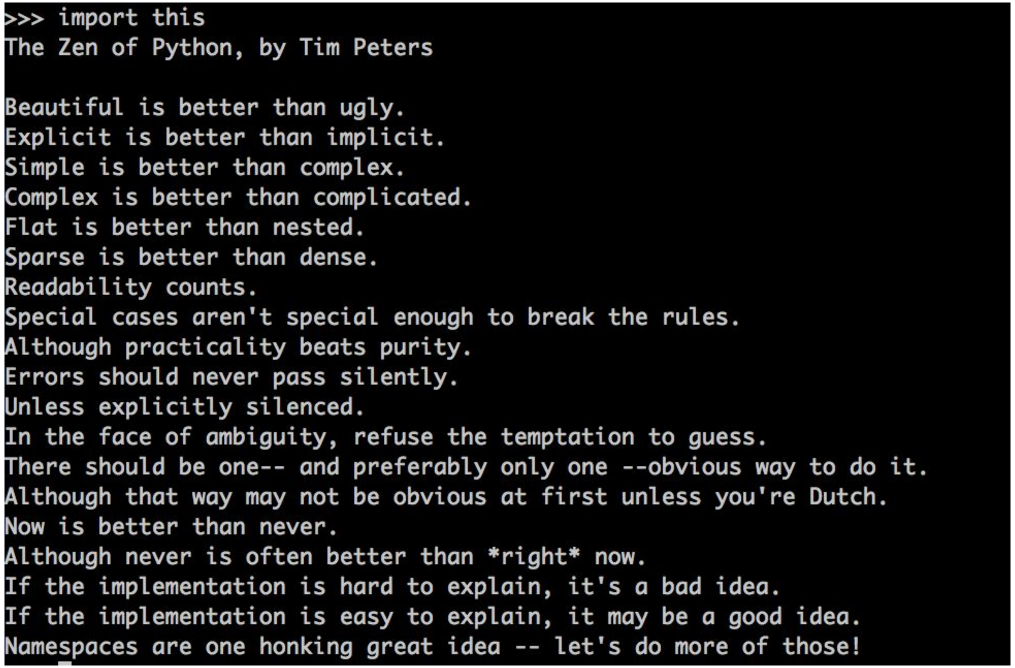
- Organizes all of your virtual environments in one place.
- Wrappers for managing your virtual environments (create, delete, copy).
- Use a single command to switch between environments.
- Tab completion for commands that take a virtual environment as argument.
- User-configurable hooks for all operations.
- Plugin system for more creating sharable extensions.



entropiae 12:08 PM

L'impostazione generale dei progetti non rispetta lo Zen di Python 😢

"There should be one— and preferably only one —obvious way to do it."



```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Package Patterns

keep test code in a separate root directory, commonly used for standalone modules as it allows us to distribute the code and tests together

```
root
|
+- package
|   |
|   +- file1
|   +- file2
|
+- tests
    |
    +- test_file1
    +- test_file2
```

```
python_boilerplate
|
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
|
+- tests
    |
    +- __init__.py
    +- test_python_boilerplate.py
```

keep test code as a submodule of the main code and has an advantage when the application has to be packaged without the test code (e.g. deploying to production servers or distributing to customers)

```
root
|
+- package
    |
    +- file1
    +- file2
    +- tests
        |
        +- test_file1
        +- test_file2
```

```
src
|
+- python_boilerplate
    |
    +- __init__.py
    +- python_boilerplate.py
    +- tests
        |
        +- __init__.py
        +- test_python_boilerplate.py
```

Excerpt From: “Test-Driven Python Development.”

cookiecutter

`cookiecutter` is a command-line utility that creates projects from **cookiecutters** (project templates).

Some features:

Cross-platform: Windows, Mac, and Linux are officially supported.
Works with Python 2.7, 3.3, 3.4, 3.5, and PyPy. (But you don't have to know/write Python code to use Cookiecutter.)
Project templates can be in any programming language or markup format: Python, JavaScript, Ruby,...
Simple command line usage.

https://github.com/audreyr/cookiecutter

```
$ pip install cookiecutter
(env) $ cookiecutter gh:audreyr/cookiecutter-pypackage
...

$ cat ~/.cookiecutterrc
default_context:
     full_name: "Antani Tatablinda"
     email: "come.se.fosse@ntani.org"
     github_username: "antani"
cookiecutters_dir: "~/.cookiecutters/"
```

There're some **cookiecutters** (aka Cookiecutter project templates) available for Python, Python-Django, C, C++, C#, Common Lisp, JavaScript, Kotlin, LaTeX/XeTeX, Berkshelf-Vagrant, HTML, Scala, 6502 Assembly

Similar projects: Paste, Diecutter, Django's startproject and startapp commands, python-packager, Yeoman, Pyramid's pcreate command, mr.bob, grunt-init, scaffolt, init-skeleton, Cog, Skaffold and maybe others :-)

Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Let’s analyze all files

created by **cookiecutter**

Dissecting a package

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Dissecting a package

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

Include, exclude files and directories

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Python Milano Meetup

Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

Include, exclude files and directories

Python code

Python Milano Meetup

Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

Include, exclude files and directories

Python code

Develop dependencies

Python Milano Meetup

Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

Include, exclude files and directories

Python code

Develop dependencies

`setup.py` serves for project configuration, contains global `setup()` function and it's a command line interface for running commands related to packaging task. `setup.cfg` is an ini file that contains option defaults for `setup.py` commands.

Python Milano Meetup

Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|   |
|   +- __init__.py
|   +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|   |
|   +- __init__.py
|   +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

Include, exclude files and directories

Python code

Develop dependencies

`setup.py` serves for project configuration, contains global `setup()` function and it's a command line interface for running commands related to packaging task. `setup.cfg` is an ini file that contains option defaults for `setup.py` commands.

Test suite

Python Milano Meetup

Dissecting a package

```
python_boilerplate
+- AUTHORS.rst
+- CONTRIBUTING.rst
+- HISTORY.rst
+- LICENSE
+- MANIFEST.in
+- Makefile
+- README.rst
+- python_boilerplate
|  |
|  +- __init__.py
|  +- python_boilerplate.py
+- requirements_dev.txt
+- setup.cfg
+- setup.py
+- tests
|  |
|  +- __init__.py
|  +- test_python_boilerplate.py
+- tox.ini
+- travis_pypi_setup.py
```

Info on authors, how to contribute, history and license. Some infos provided by cookiecutter.

Include, exclude files and directories

Python code

Develop dependencies

setup.py serves for project configuration, contains global setup() function and it's a command line interface for running commands related to packaging task. setup.cfg is an ini file that contains option defaults for setup.py commands.

Test suite

Because we want write docs

```
+- docs
|  |
|  +- Makefile
|  +- authors.rst
|  +- conf.py
|  +- contributing.rst
|  +- history.rst
|  +- index.rst
|  +- installation.rst
|  +- make.bat
|  +- readme.rst
|  +- usage.rst
```

bumpversion

`bumpversion` is a small command line tool to simplify releasing software by updating all version strings in your source code by the correct increment. Also creates commits and tags.

Features:

- version formats are highly configurable
- works without any VCS, but happily reads tag information from and writes commits and tags to Git and Mercurial if available
- just handles text files, so it's not specific to any programming language.

https://github.com/peritus/bumpversion/

```
$ pip install --upgrade bumpversion

$ bumpversion patch
$ bumpversion minor
$ bumpversion major
```

Python Milano Meetup

punch

`punch.py` is a small command line tool to simplify releasing software by updating all version strings in your source code by the correct increment. Also creates commits and tags.

It's a replacement for bumpversion.

https://github.com/lgiordani/punch

```
$ pip install --upgrade punch.py

$ punch --init
     # which will create the
     # punch_config.py
     # punch_version.py
     # in the current directory

$ punch --part patch
...
```

Python Milano Meetup

Some other useful links

- Welcome to The Hitchhiker's Guide to Packaging
  https://the-hitchhikers-guide-to-packaging.readthedocs.io/en/latest/
- The Hitchhiker's Guide to Python!
  http://docs.python-guide.org/en/latest/
- A Human's Ultimate Guide to setup.py.
  https://github.com/kennethreitz/setup.py
- Alice in Python projectland
  http://veekaybee.github.io/2017/09/26/python-packaging/
- Packaging and Distributing Projects
  https://packaging.python.org/tutorials/distributing-packages/
- Packaging Python software with pbr
  https://julien.danjou.info/blog/2017/packaging-python-with-pbr

PyPI

`PyPI` is a repository of software for Python and currently count more than 76000 packages.

The PyPI repository provides alternative locations that store the packages.

You can create your own mirror, following the PEP 381 or using a tool such as `pep381client`

You can host your own private repository: e.g. `pypiserver` can easly do the job.

https://pypi.python.org/pypi

```
$ virtualenv cheeseshop
$ cd cheeshop ; . bin/activate
(cheeseshop) $ pip install pypiserver passlib
...
(cheeseshop) $ mkdir packages
(cheeseshop) $ pypi-server -p 8080 ~/packages
```

```
(env) $ pip install  --extra-index-url
http://localhost:8080/simple/...
```

```
$ cat .pypirc
[distutils]
index-servers =
      pypi
      myPyPi

[pypi]
repository=https://pypi.python.org/pypi
username=
password=

[myPyPi]
repository=http://csgs1.pythonanywhere.com
username=foobar
password=barfoo
```

Linting & Coverage

- Pylint: static checker, provide statistics about code complexity, check code errors, style errors, code smells and print reports
  => https://pylint.org
- Pyflakes: perform logic checks in code
  => https://launchpad.net/pyflakes
- McCabe: script that check McCabe
  plugin
  => https://pypi.python.org/pypi/mccabe
- Pycodestyle: code style guide, previously known a
  => https://pypi.python.org/pypi/pycodestyle
- Flake8: wrapper around Pyflakes, McCabe and pycodestyle
  => http://flake8.pycqa.org/en/latest/
- Coverage.py: measuring code coverage with reports
  => https://coverage.readthedocs.io/en/

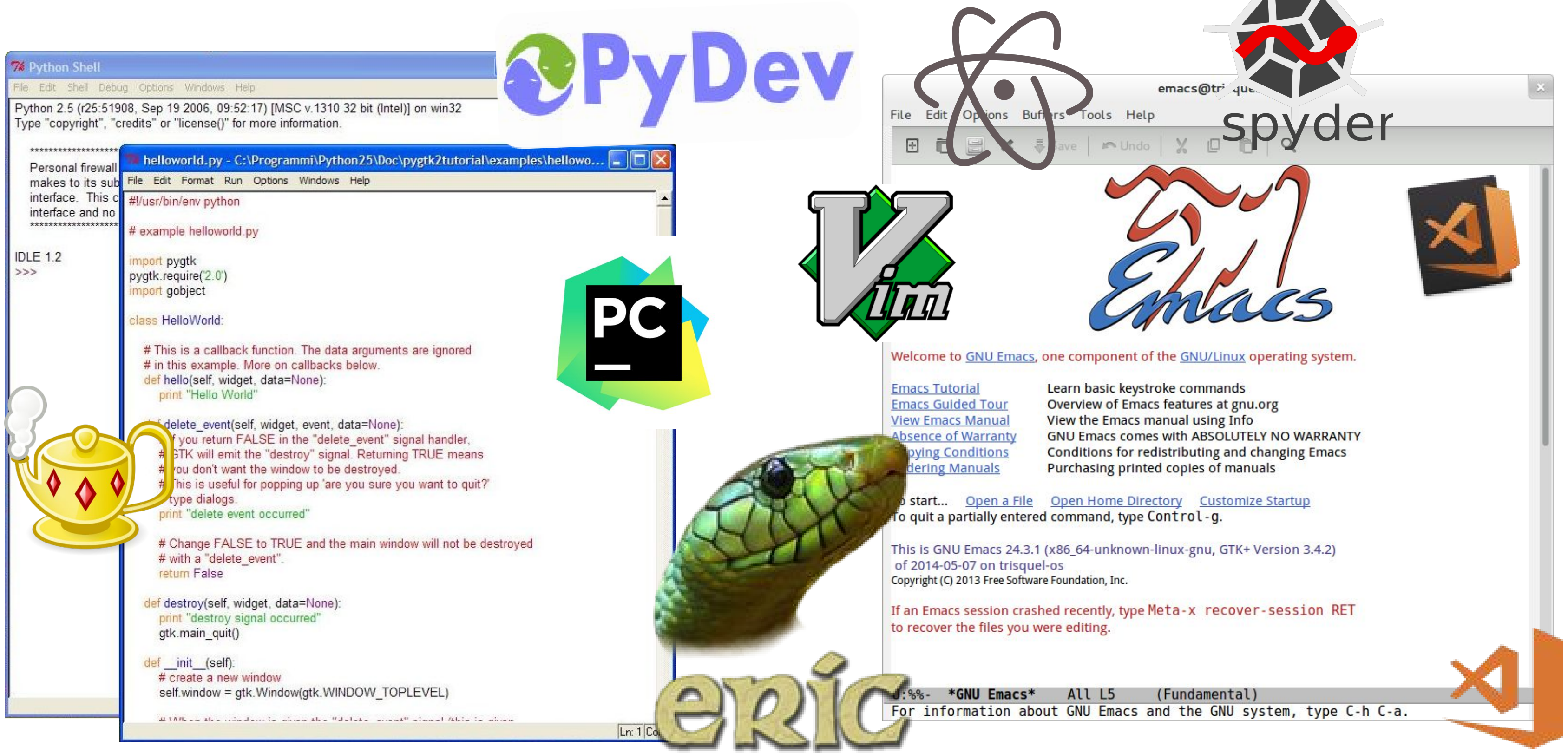
M = E - N + 2P
E => Number of edges in graph
N => Number of nodes in graph
P => Number of connected components in graph

Python Milano Meetup

Testing Frameworks

- unittest: included into the standard library, inspired from jUnit
  => https://docs.python.org/3.6/library/unittest.html
- doctest: included into the standard library, a special way to test with “comments”
  => https://docs.python.org/3.6/library/doctest.html
- py.test: makes it easy to write small tests, yet scales to support complex functional testing for applications and libraries
  => https://docs.pytest.org/en/latest/
- nose2: test runner based on unittest2
  => https://github.com/nose-devs/nose2
- Frameworks could be the own test suite

IDEs… WAR! Make code not wars, please! Codemotion rulez!


Python Shell
File Edit Shell Debug Options Windows Help
PyDev
spyder
Vim
Emacs
PC
eric
File Edit Options Buffers Tools Help
Undo
File Edit Format Run Options Windows Help
Ln: 1 Co

```
Python 2.5 (r25:51908, Sep 19 2006, 09:52:17) [MSC v.1310 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

********************
Personal firewall
makes to its sub
interface. This c
interface and no
********************

IDLE 1.2
>>>
```

helloworld.py - C:\Programmi\Python25\Doc\pygtk2tutorial\examples\hellowo…

```
#!/usr/bin/env python

# example helloworld.py

import pygtk
pygtk.require('2.0')
import gobject

class HelloWorld:

    # This is a callback function. The data arguments are ignored
    # in this example. More on callbacks below.
    def hello(self, widget, data=None):
        print "Hello World"

    def delete_event(self, widget, event, data=None):
        # f you return FALSE in the "delete_event" signal handler,
        # GTK will emit the "destroy" signal. Returning TRUE means
        # you don't want the window to be destroyed.
        # This is useful for popping up 'are you sure you want to quit?'
        # type dialogs.
        print "delete event occurred"

        # Change FALSE to TRUE and the main window will not be destroyed
        # with a "delete_event".
        return False

    def destroy(self, widget, data=None):
        print "destroy signal occurred"
        gtk.main_quit()

    def __init__(self):
        # create a new window
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
```

emacs@tri que

```
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Emacs Tutorial          Learn basic keystroke commands
Emacs Guided Tour       Overview of Emacs features at gnu.org
View Emacs Manual       View the Emacs manual using Info
Absence of Warranty     GNU Emacs comes with ABSOLUTELY NO WARRANTY
pying Conditions        Conditions for redistributing and changing Emacs
dering Manuals          Purchasing printed copies of manuals

o start...   Open a File   Open Home Directory   Customize Startup
To quit a partially entered command, type Control-g.

This is GNU Emacs 24.3.1 (x86_64-unknown-linux-gnu, GTK+ Version 3.4.2)
 of 2014-05-07 on trisquel-os
Copyright (C) 2013 Free Software Foundation, Inc.

If an Emacs session crashed recently, type Meta-x recover-session RET
to recover the files you were editing.

-:%%-  *GNU Emacs*    All L5     (Fundamental)
For information about GNU Emacs and the GNU system, type C-h C-a.
```

Python Milano Meetup

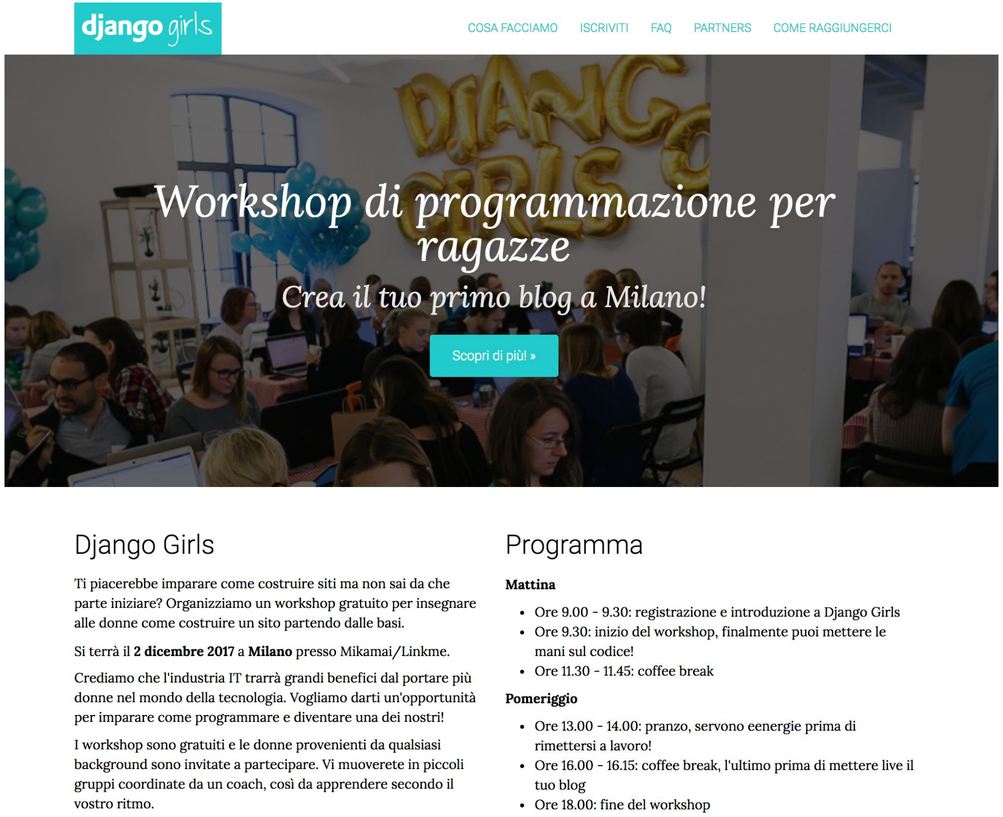
Next Events

Python e la Fotografia Astronomica Automatizzata

30/11/2017

https://pymi.eventbrite.com

Python Milano Meetup



Workshop di programmazione per ragazze

Crea il tuo primo blog a Milano!

Django Girls

Ti piacerebbe imparare come costruire siti ma non sai da che parte iniziare? Organizziamo un workshop gratuito per insegnare alle donne come costruire un sito partendo dalle basi.

Si terrà il **2 dicembre 2017** a **Milano** presso Mikamai/Linkme.

Crediamo che l'industria IT trarrà grandi benefici dal portare più donne nel mondo della tecnologia. Vogliamo darti un'opportunità per imparare come programmare e diventare una dei nostri!

I workshop sono gratuiti e le donne provenienti da qualsiasi background sono invitate a partecipare. Vi muoverete in piccoli gruppi coordinate da un coach, così da apprendere secondo il vostro ritmo.

Programma

Mattina

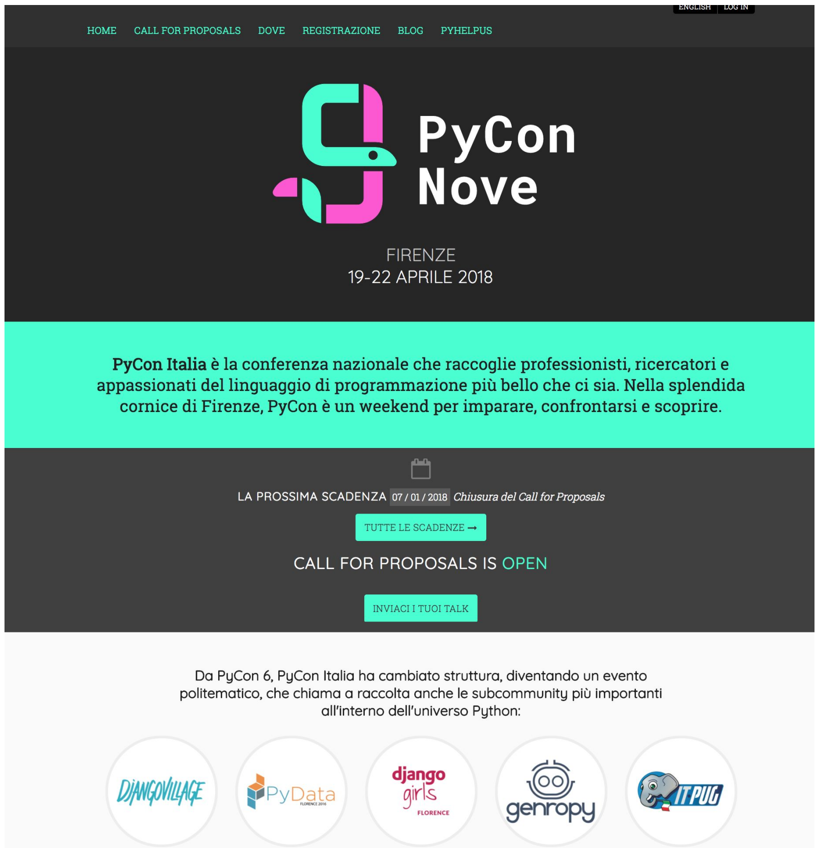
- Ore 9.00 - 9.30: registrazione e introduzione a Django Girls
- Ore 9.30: inizio del workshop, finalmente puoi mettere le mani sul codice!
- Ore 11.30 - 11.45: coffee break

Pomeriggio

- Ore 13.00 - 14.00: pranzo, servono eenergie prima di rimettersi a lavoro!
- Ore 16.00 - 16.15: coffee break, l'ultimo prima di mettere live il tuo blog
- Ore 18.00: fine del workshop

https://djangogirls.org/milano/

Python Milano Meetup



FIRENZE
19-22 APRILE 2018

PyCon Italia è la conferenza nazionale che raccoglie professionisti, ricercatori e appassionati del linguaggio di programmazione più bello che ci sia. Nella splendida cornice di Firenze, PyCon è un weekend per imparare, confrontarsi e scoprire.

LA PROSSIMA SCADENZA 07 / 01 / 2018 *Chiusura del Call for Proposals*

CALL FOR PROPOSALS IS OPEN

Da PyCon 6, PyCon Italia ha cambiato struttura, diventando un evento politematico, che chiama a raccolta anche le subcommunity più importanti all'interno dell'universo Python:

https://www.pycon.it/

Thanks!

"Python's a drop-in replacement for BASIC in the sense that Optimus Prime is a drop-in replacement for a truck."
– Cory Dodt

Follow me!
@cstrap on Twitter, GitHub, Bitbucket, LinkendIn, Trello...

Python Milano Meetup