Università degli Studi di Cagliari
Corso di Laurea DSBAI

Web Analytics e Analisi Testuale

http://agilegroup.eu

A.A. 2017/2018

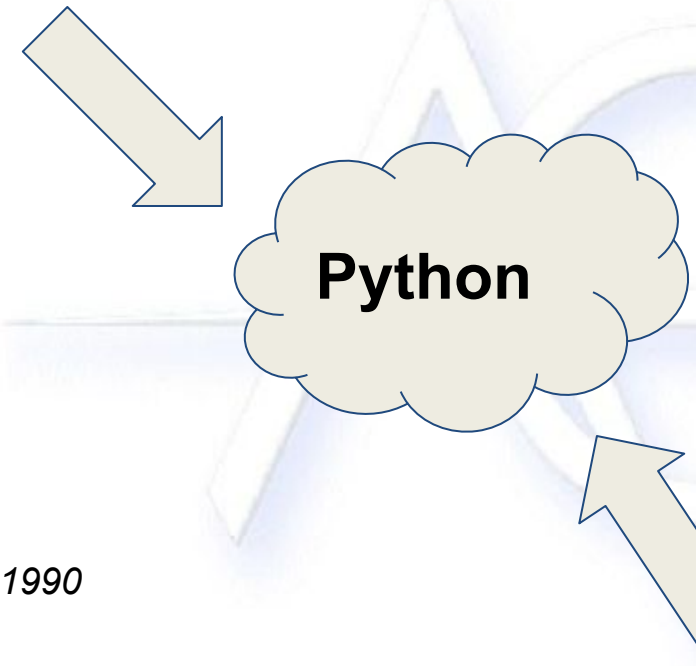
Ing. Marco Ortu
Via Porcell 4, primo piano
mail: marco.ortu@diee.unica.it

Python

Cos'è Python

Object Oriented Programming
(C++, Modula-3, ABC, Icon)

Scripting Languages
(Perl, Tcl)



Author:
Guido Van Rossum, 1990

Functional Programming
(Scheme)

Cos'è Python: Zen

```
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.

...
```

Cos’è Python: Zen

```
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.

...
```

Cos’è Python: Zen

```
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!
```

Cos’è Python: https://www.python.org

- Python is an easy to learn, powerful programming language.
- It has efficient high-level data structures and a simple but effective approach to object-oriented programming.
- Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Cos’è Python:

Il Python è un linguaggio che supporta diversi paradigmi/stili di programmazione:

- procedurale
- funzionale
- a oggetti
- meta-programmazione
- scripting

Installazione: Python

Guida completa

- ➔ guida completa alla installazione di Python
- ➔ come installare Pip si windows
- ➔ Installare Python 2.7.14
- ➔ Installare Python 3.6.4
  - ◆ **N.B!!** spuntare l'opzione "ADD PYTHON TO PATH"

Installazione: Git

Scegliere la versione **Windows**:

➔ https://git-scm.com/download/win

Installazione: Pycham Community

Scegliere la versione **Community**:

- ➔ https://www.jetbrains.com/pycharm/download
- ➔ Opzioni:
  - ◆ Associa file .py
  - ◆ Scarica jre
- ➔ Al primo avvio installare plugin:
  - ◆ Markdown

Pycharm: nuovo progetto

Primo avvio:

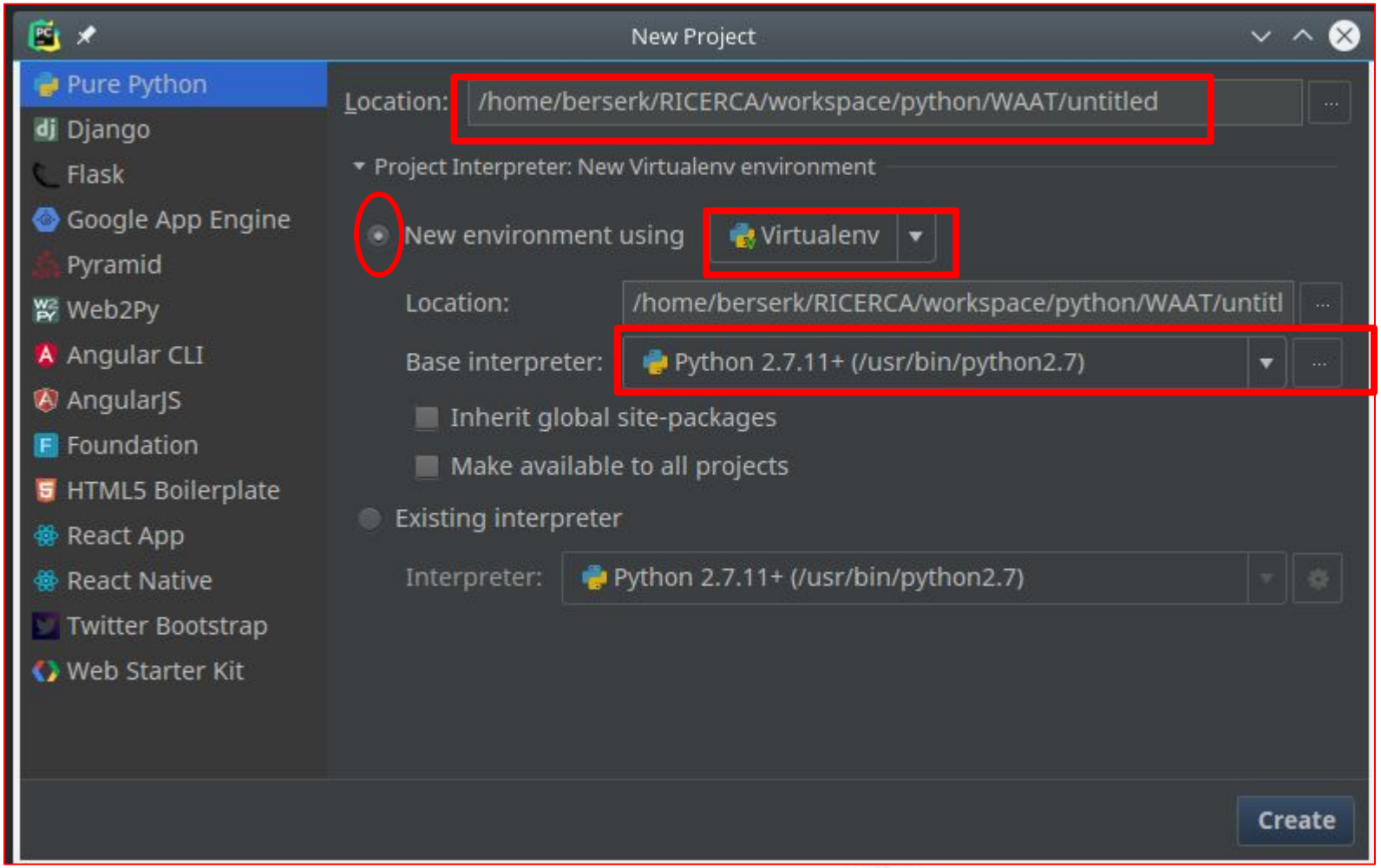
➔ Create New Project

Dal menu File:

➔ File -> New Project

- In "***Location***" -> selezionare il percorso dove salvare il progetto
- Selezionare "***New Virtual Environment with" -> VirtualEnv***
- Base interpreter -> python2.7 (oppure python3.6)

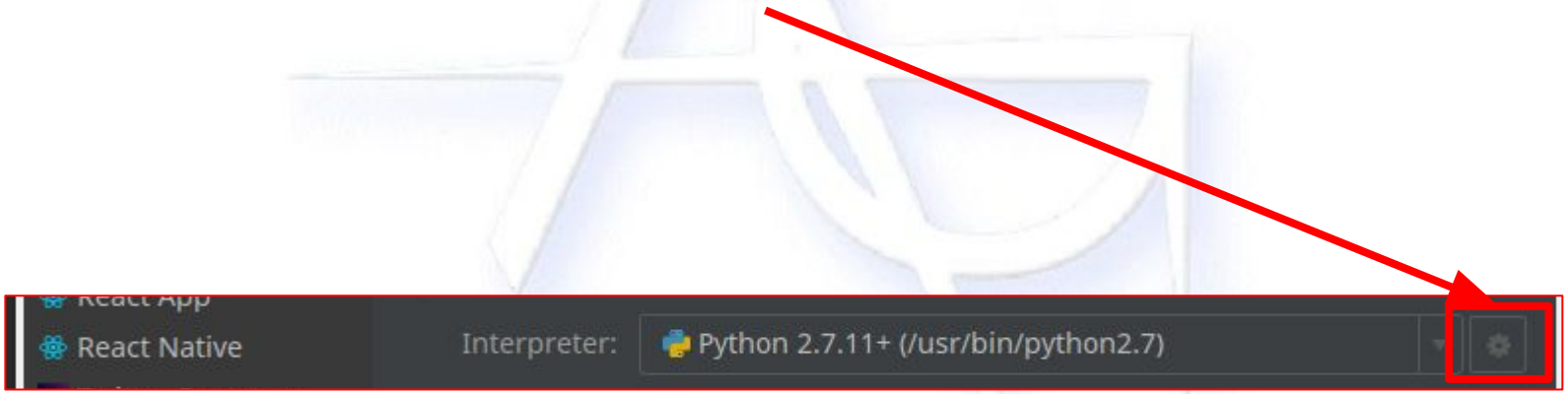
Pycharm: nuovo progetto



Pycharm: nuovo progetto

In alcuni casi potrebbe non comparire la precedente interfaccia grafica per la selezione dell'interprete Python.

In questo caso cliccare sull'ingranggio, selezionare "***add local***" per mostrare la precedente interfaccia grafica.



Pycharm: hello world

Creare un nuovo file:

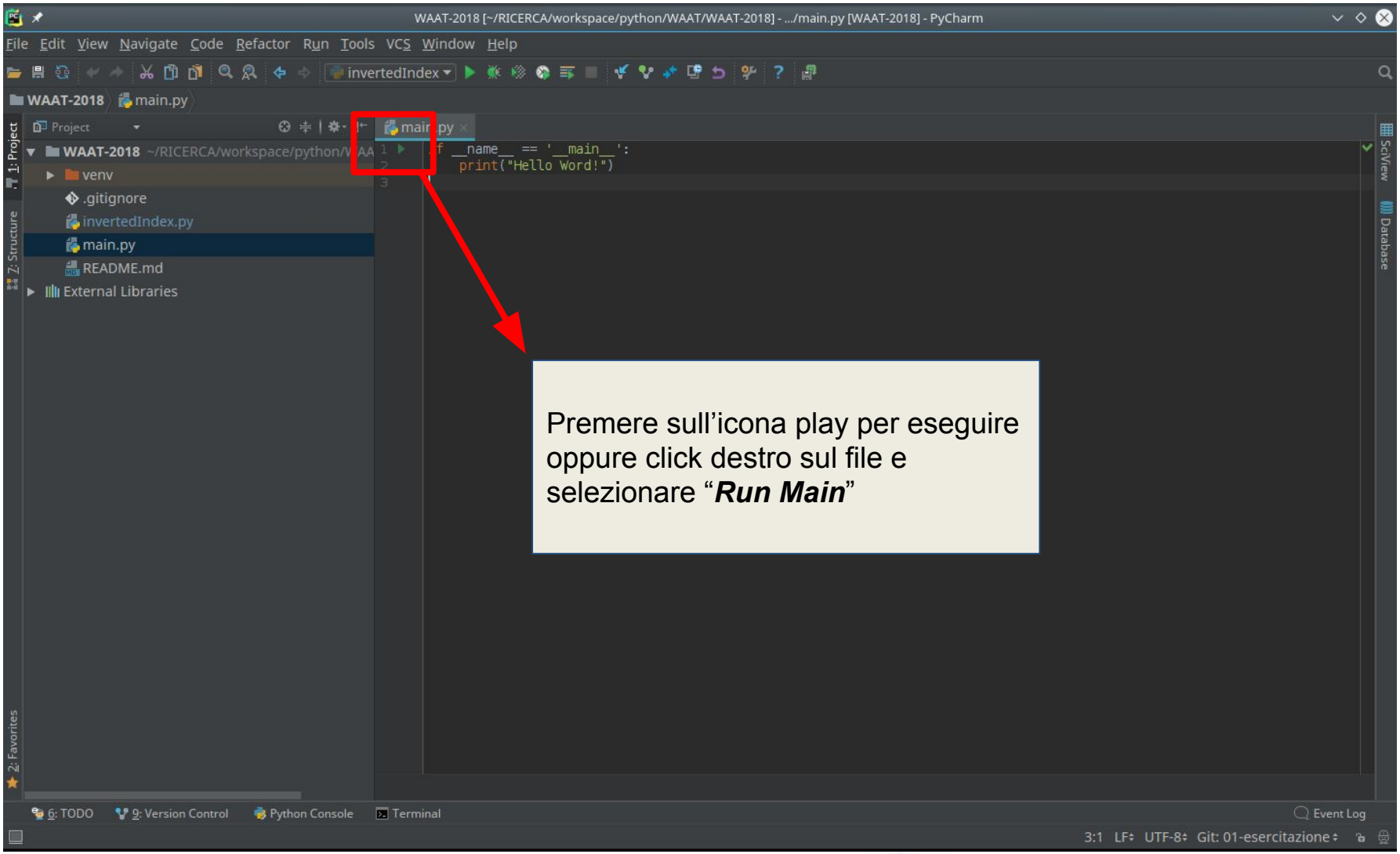
➔ File -> New -> Python file

- chiamatelo ***“main”***

main.py

```
if __name__ == ‘__main__’:
    print(‘Hello World’)
```

Pycharm: hello world



Python: Set

```
# initialize my_set
my_set = {1, 3}
print(my_set)
# if you uncomment line 9,
# you will get an error
# TypeError: 'set' object does not support indexing
# my_set[0]
# add an element
# Output: {1, 2, 3}
my_set.add(2)
print(my_set)
# add multiple elements
# Output: {1, 2, 3, 4}
my_set.update([2, 3, 4])
print(my_set)
# add list and set
# Output: {1, 2, 3, 4, 5, 6, 8}
my_set.update([4, 5], {1, 6, 8})
print(my_set)
# initialize my_set
my_set = {1, 3, 4, 5, 6}
print(my_set)
```

Python: Set

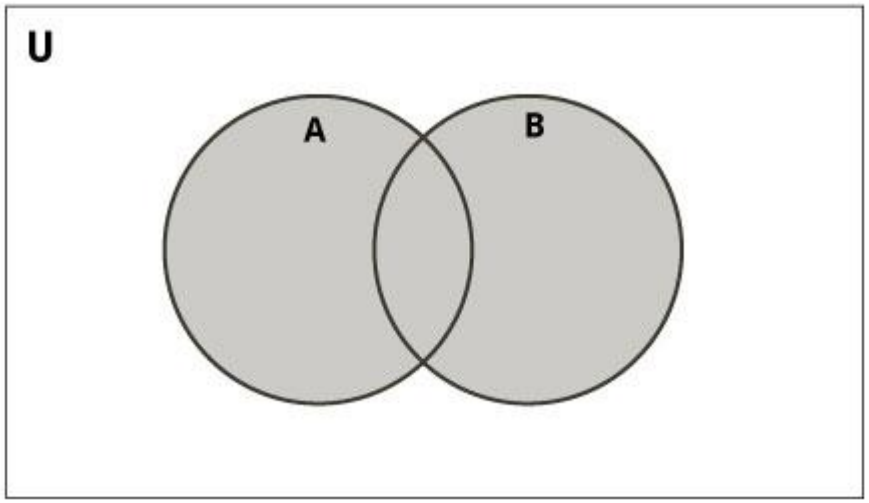
```
# discard an element
# Output: {1, 3, 5, 6}
my_set.discard(4)
print(my_set)

# remove an element
# Output: {1, 3, 5}
my_set.remove(6)
print(my_set)

# discard an element
# not present in my_set
# Output: {1, 3, 5}
my_set.discard(2)
print(my_set)

# remove an element
# not present in my_set
# If you uncomment line 27,
# you will get an error.
# Output: KeyError: 2

# my_set.remove(2)
```

Python: Set

```
# initialize my_set
# Output: set of unique elements
my_set = set("HelloWorld")
print(my_set)

# pop an element
# Output: random element
print(my_set.pop())

# pop another element
# Output: random element
my_set.pop()
print(my_set)

# clear my_set
# Output: set()
my_set.clear()
print(my_set)
```

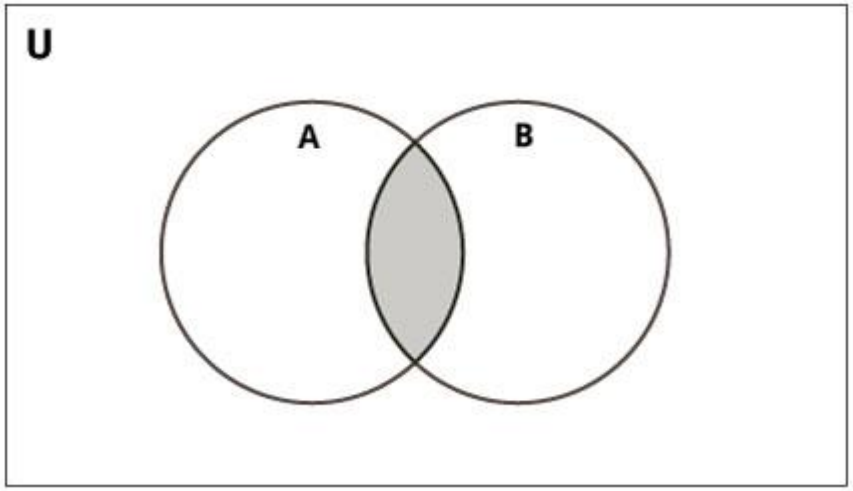
Python: Set

```
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use | operator
# Output: {1, 2, 3, 4, 5, 6, 7, 8}
print(A | B)
print(A.union(B))
print(B.union(A))
```



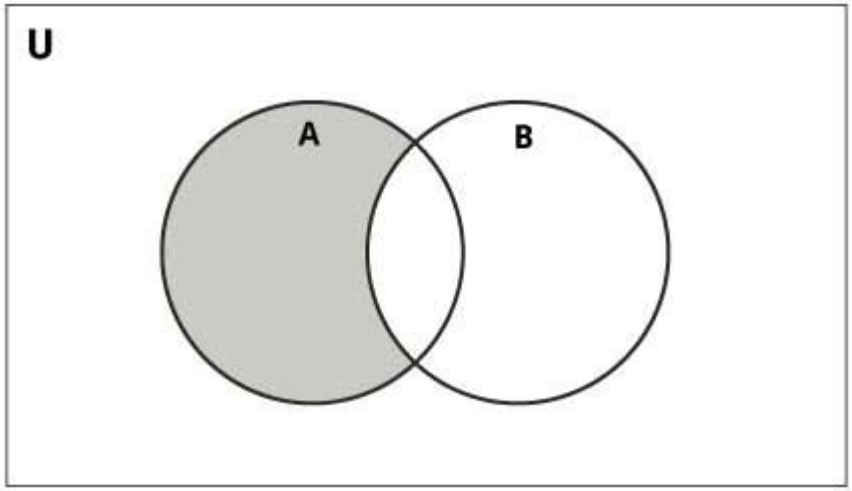
Python: Set

```
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use & operator
# Output: {4, 5}
print(A & B)
print(A.intersection(B))
print(B.intersection(A))
C = {4, 7, 10, 11}
print(A & B & C)
print(A.intersection(B, C))
# If can pass a runtime list to intersection
print(A.intersection(*[B, C]))
```



Python: Set

```
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use - operator on A
# Output: {1, 2, 3}
print(A - B)
print(A.difference(B))
```



Python: Set

```
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use ^ operator
# Output: {1, 2, 3, 6, 7, 8}
print(A ^ B)
```