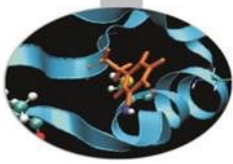
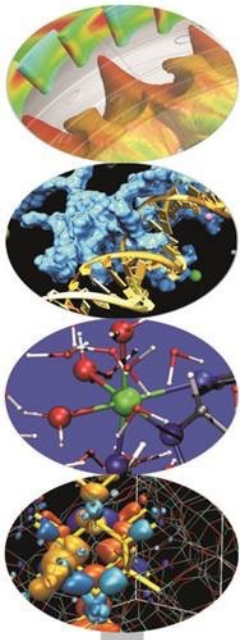
CINECA SCAI
SuperComputing Applications and Innovation



Benvenuti

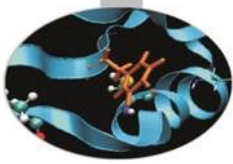


Materiale del corso

http://j.mp/cineca_scpy

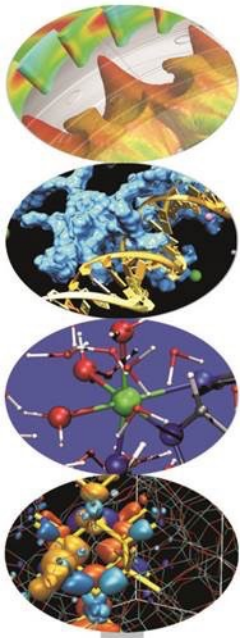
Paolo D'Onorio De Meo
p.donoriodemeo@cineca.it

CINECA

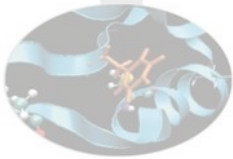


Avvio

Utilizzare i computer portatili
all'interno di questa sala corsi



Utilizzo dei portatili

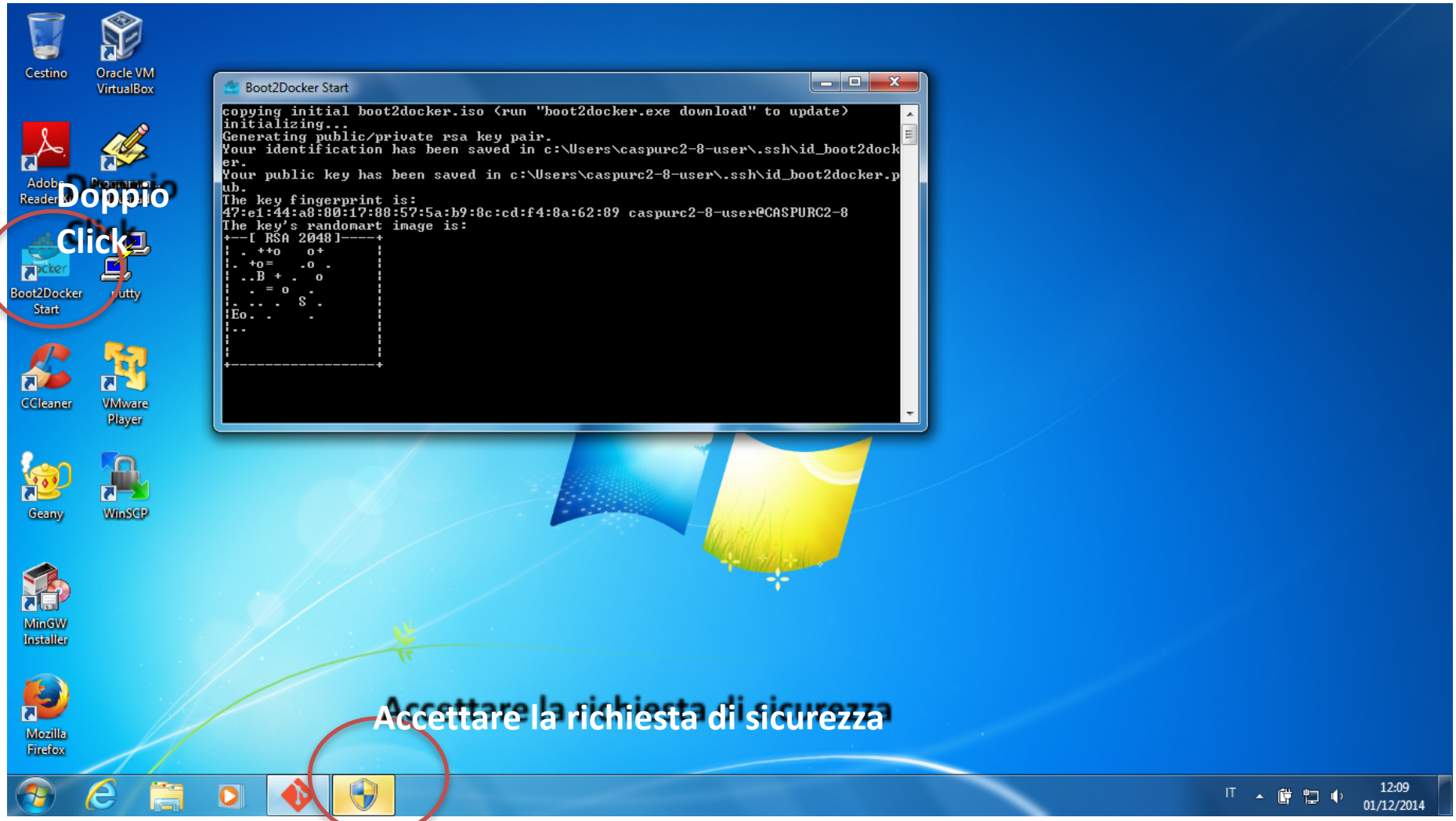


- Avviare il computer mantenendo premuto il pulsante power
- Non selezionare nessun sistema operativo al momento della richiesta interattiva
  - Lasciare eseguire Windows 7
  - Selezionare nella schermata di login l'utenza per cui avete ricevuto la password
  - Verificare una volta acceduto che la rete sia connessa
    - es. lanciare Mozilla Firefox e connettersi a un sito a piacere

Successivamente…. ->

Materiale del corso
http://j.mp/cineca_scpy

Utilizzo dei portatili



Utilizzo dei portatili: avvio completato

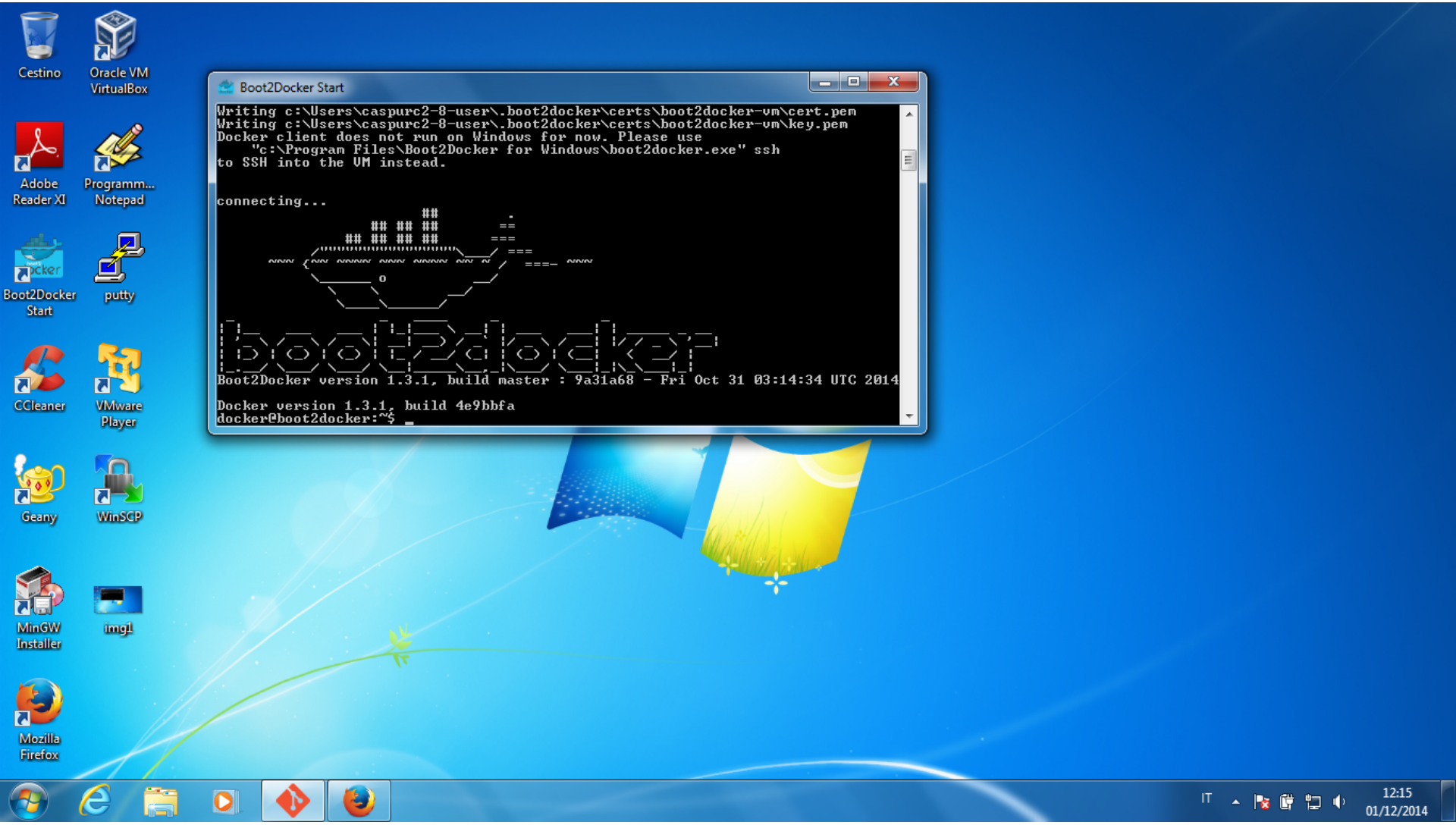
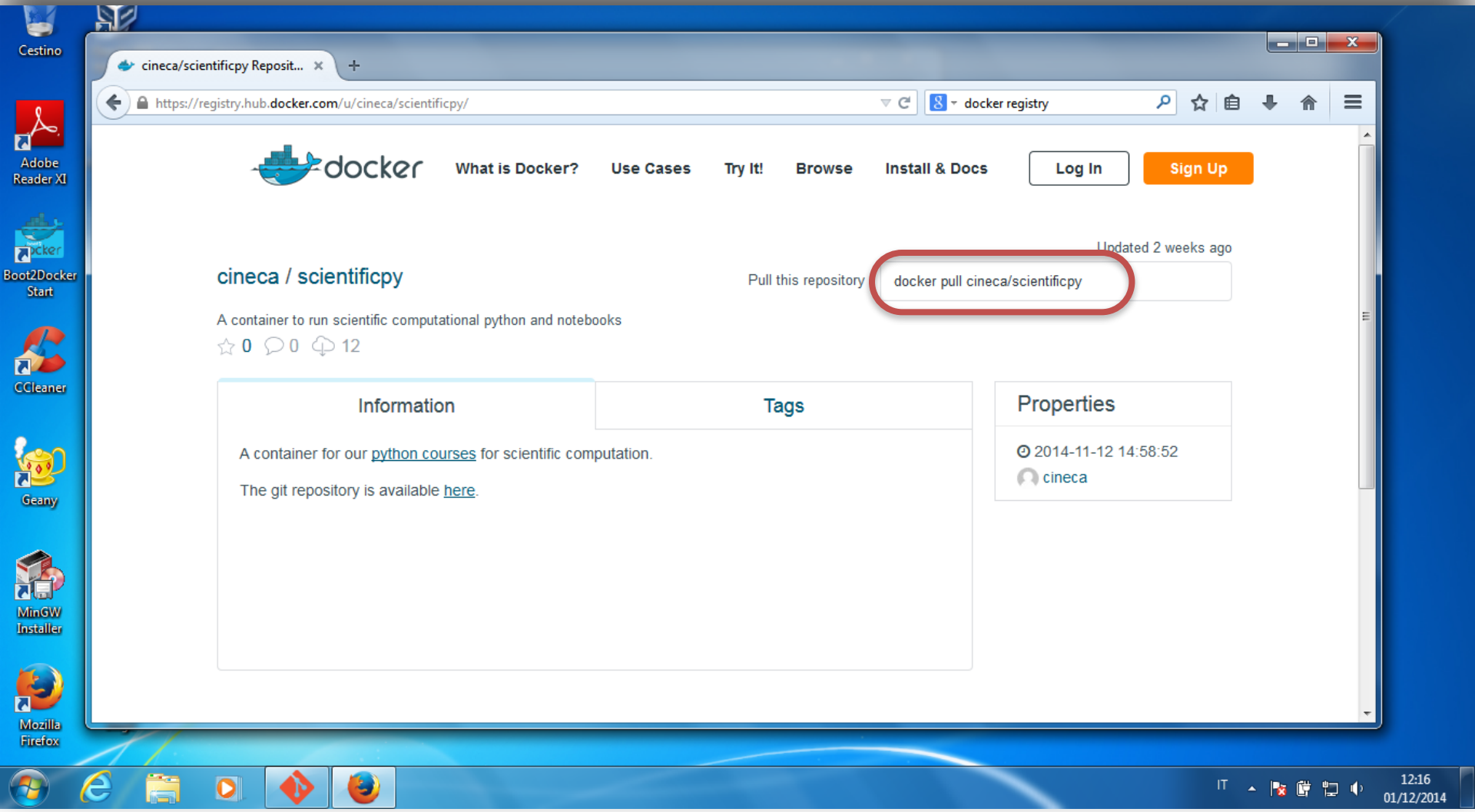
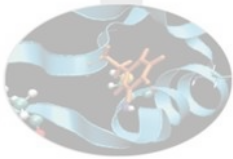


Immagine virtuale che utilizzeremo



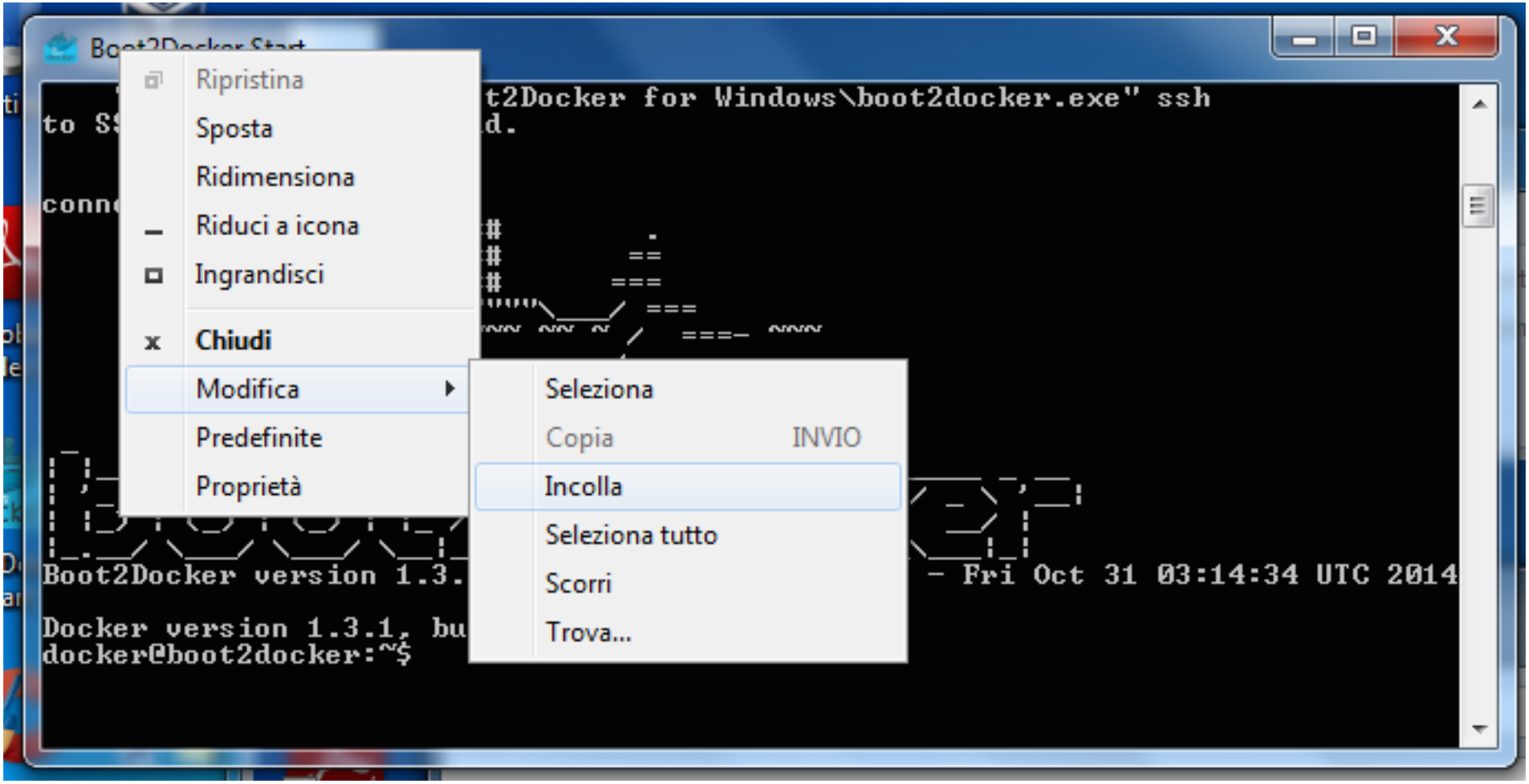


Installazione in locale

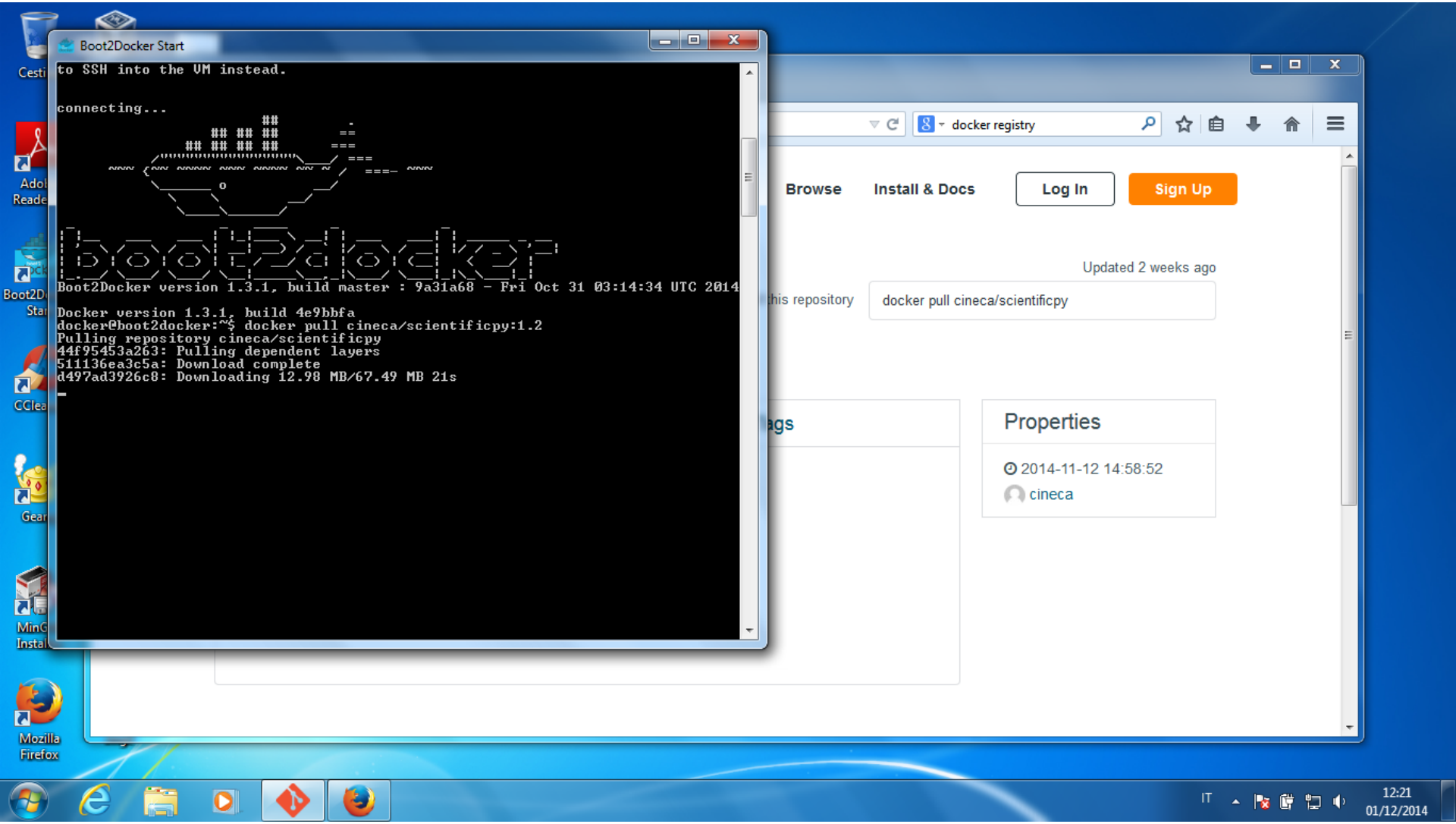
Il comando da eseguire (copia/incolla) nel terminale:

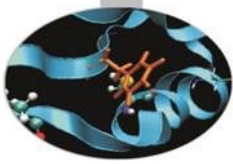
```
docker pull cineca/scientificpy
```

Incollare nel terminale

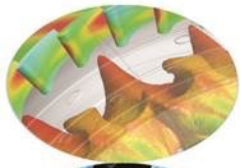
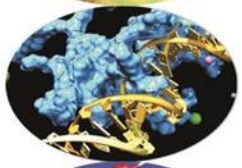
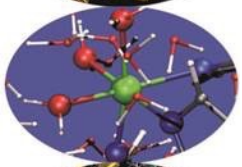
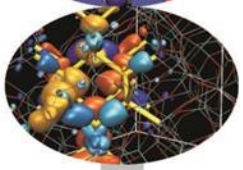


Scaricamento immagine





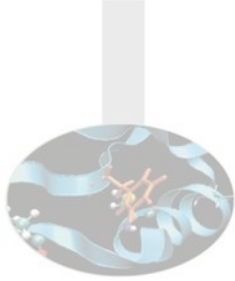
Cosa spiegheremo

Materiale del corso

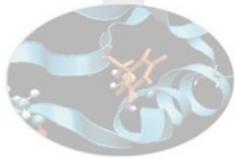
http://j.mp/cineca_scpy

Prima parte: Sommario



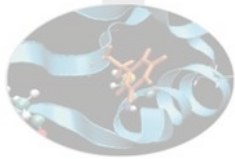
1. Introduzione al linguaggio
2. Interprete e Modalità batch
3. Strutture dati
    1. Stringhe
    2. Liste e Tuple
    3. Dizionari
    4. Insiemi
4. Controllo del flusso
    1. If then else
    2. For
    3. While
5. Accesso ai file
6. Funzioni e moduli

Prima parte: Agenda



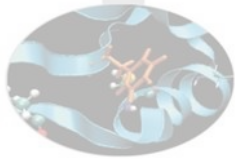
Martedì 02/12/2014 Ore 10-13	Introduzione a Python
Martedì 02/12/2014 Ore 14-17	Strutture dati e flussi di controllo, I/O, funzioni e moduli
Mercoledì 03/12/2014 Ore 10-13	Docker, ipython, notebooks

Prima parte: Sommario



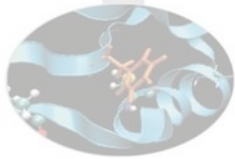
1. Introduzione al linguaggio
2. Interprete e Modalità batch
3. Strutture dati
    1. Stringhe
    2. Liste e Tuple
    3. Dizionari
    4. Insiemi
4. Controllo del flusso
    1. If then else
    2. For
    3. While
5. Accesso ai file
6. Funzioni e moduli

Seconda parte: Agenda



Mercoledì 04/12/2014 Ore 14-17	Introduzione a NumPy Matplotlib: il modulo pylab
Giovedì 05/12/2014 Ore 10-13	Introduzione a SciPy
Giovedì 05/12/2014 Ore 14-17	mixed language programming

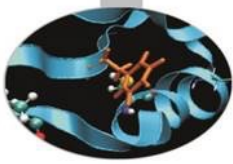
Seconda parte: Sommario



1. Introduzione a NumPy prima parte (l'oggetto NDarray)
2. Matplotlib: il modulo pylab
3. Introduzione a NumPy seconda parte (operazioni su array)

4. Introduzione a SciPy
5. Performance in Python: mixed language programming
    1. Introduzione
    2. F2py
    3. Cython
    4. 2D wave equation: un caso reale

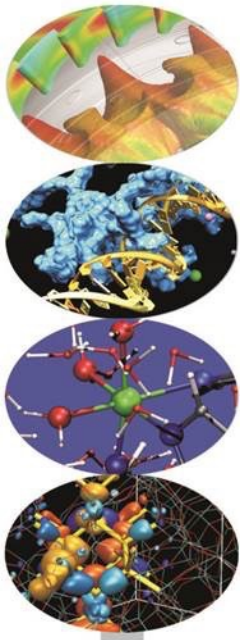
CINECA SCAI
SuperComputing Applications and Innovation



Python: il linguaggio

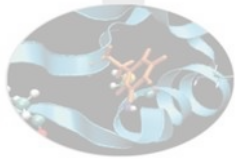
Benvenuti!

Materiale del corso
http://j.mp/cineca_scpy



Paolo D'Onorio De Meo
p.donoriodemeo@cineca.it

CINECA



Python

Un linguaggio di programmazione

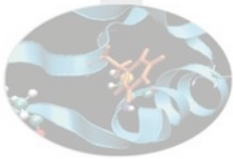
moderno

che sta imponendo

il suo potenziale

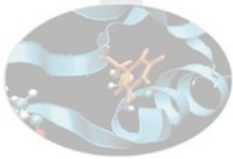
all'attenzione

degli sviluppatori e della comunità scientifica



PYTHON: principali caratteristiche

- Leggibilità
- Librerie standard
  - includono: xml, url and web browsing, zip, email
- Moduli di terze parti
  - plotting 2/3D, PDF, giochi, DB
- Strutture dati
- Multi-paradigma
  - funzionale e/o programmazione ad oggetti
- Estensibilità
- Open source, Comunità
- Multi-piattaforma

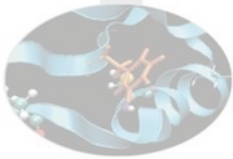


Comparazione con altri linguaggi

"You may be wondering why you should use Python,
and not more well known languages like C, Perl or JAVA.
It is a good question."

Python could be almost fully understood
by just knowing English.

Comparazione: leggibilità



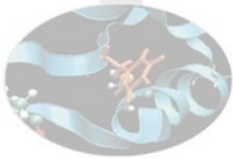
PYTHON

```
print("Hello world!")
```

JAVA

```
public class Hello
{
     public static void
     main(String[] args) {
          System.out.printf("Hello
     world!");
         }
}
```

Comparazione: leggibilità



PYTHON

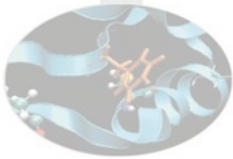
```
in = open("input.txt")
out = open("output.txt",
   "w")
out.writelines(in)
in.close()
out.close()
```

OR in one line (!):

```
open("output.txt", "w").writelines(open("input.txt"))
```

C

```
#include <stdio.h>
int main(int argc, char **argv) {
     FILE *in, *out;
     int c;
     in = fopen("input.txt", "r");
     out = fopen("output.txt", "w");
     while ((c = fgetc(in)) != EOF)
     {
          fputc(c, out);
     }
     fclose(out);
     fclose(in);
}
```

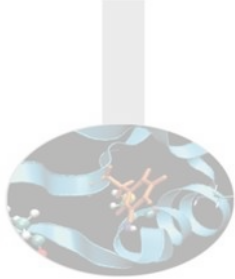


Comparazione: velocità di esecuzione

- Un programma interpretato che impiegava una settimana a girare, adesso richiede 10 secondi
  - Quello compilato un secondo
  - Meno rilevante se consideriamo il tempo di sviluppo

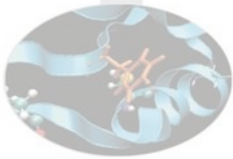
Comunque:

- Speed up 10X può essere cruciale
  - lunghi calcoli scientifici
  - si può coprire parte del gap scrivendo codice ottimizzato



PYTHON: per cosa si usa

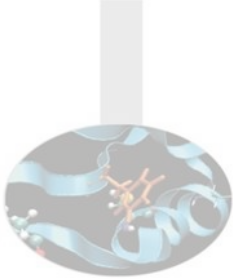
- Non è specifico per una nicchia
  - es. perl/php applicazioni web. Java per desktop
- Usato nei grandi progetti
  - Google, YouTube (!)
  - DropBox
  - Wikipedia
  - Open-Office
  - Yahoo
  - Sito web Nasa.org
  - BitTorrent
- Comunità scientifica
  - SciPy



PYTHON: flavors

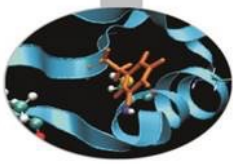
Python is actually a language definition

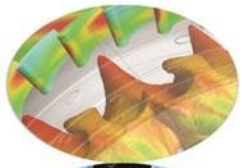
Implementazione standard: cPython

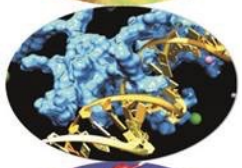


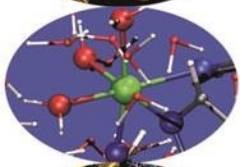
PYTHON: flavors

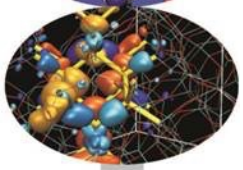
- cPython
  - Versione standard, la più usata
- PyPy
  - sperimentale: python scritto in python
- Stackless
  - sperimentale: Custom design
- Jython
  - Scritto in Java, gira nella JVM
  - Aggiunge librerie Jython al sistema Java
- IronPython
  - Adattato alla piattaforma .Net/.mono e Silverlight
    - Microsoft
- Bundles





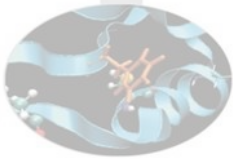






Corso python

IL PROBLEMA DELLE VERSIONI

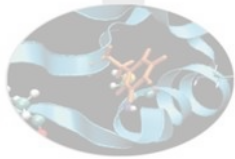


Python e le versioni recenti

Looking for a specific release?

Python releases by version number:

Release version	Release date	
Python 3.4.2	2014-10-13	Download
Python 3.3.6	2014-10-12	Download
Python 3.2.6	2014-10-12	Download
Python 2.7.8	2014-07-2	Download
Python 2.7.7	2014-06-1	Download
Python 3.4.1	2014-05-19	Download
Python 3.4.0	2014-03-17	Download



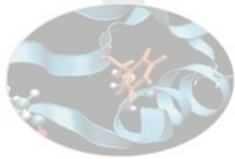
Python 2 e Python 3

https://wiki.python.org/moin/Python2orPython3

What are the differences?

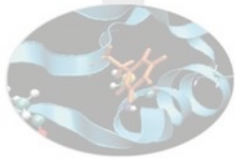
Short version:

- Python 2.x is the status quo,
- Python 3.x is the present and future of the language



Python 2 e Python 3

- Guido van Rossum ha deciso di ripulire Python 2.x in maniera appropriata...
  - ….senza preoccuparsi INTENZIONALMENTE della **retro-compatibilità**!
- Versione 3: Differenze drastiche
  - Per rendere ancora più facile imparare il linguaggio
  - Più consistenza in tutto il linguaggio
  - Più efficiente
    - strutture ottimizzate
    - Supporto Unicode
    - Separazione bytes
  - Print e exec sono funzioni! :)



Python 2 e Python 3

Porting e/o coesistenza?

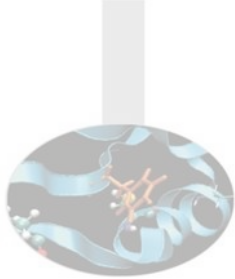
- Porting molto complicato
- Scrivere un codice compatibile per entrambe le versioni …
  - …è troppo contorto!
- In compenso le due versioni possono coesistere
  - nello stesso sistema operativo

Python 2

“pragmatici”

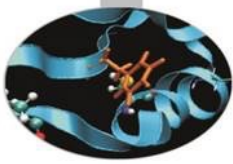
Python 3

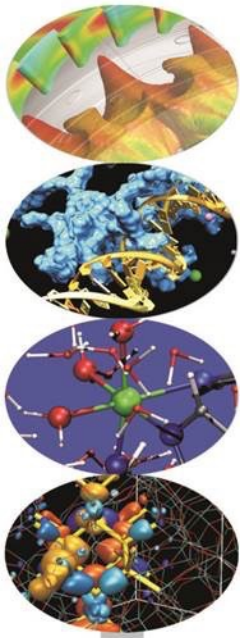
“idealisti”



Python 2 o Python 3?

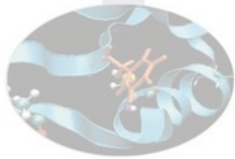
- Python 2 sarà il riferimento di questo corso
  - Standard attuale de facto
  - Riferimento per la comunità scientifica
  - Compatibilità al 100% con tutte le librerie matematiche e di plotting





Corso python

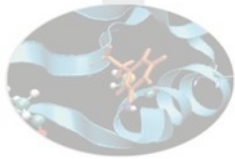
INSTALLAZIONE



Installazione

"In Windows
you have to download the Windows installer
from the Python download page "

- http://www.python.org/download
- Installation is pretty straightforward

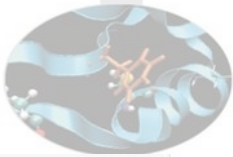


Installazione

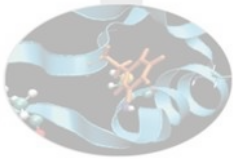
Per Linux e Mac OS X Python solitamente si trova già installato tra i pacchetti/ programmi previsti di default

```
paulie@workstation:~$ which python
   /usr/bin/python
paulie@workstation:~$ python -V
   Python 2.7.8
```

Docker (just a glimpse)

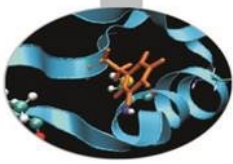


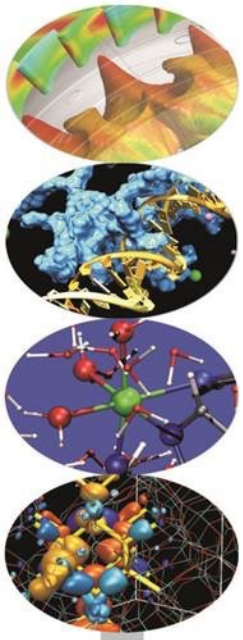
```
FROM ubuntu:14.04
# Install ubuntu packages (compilers and other tools)
RUN apt-get update && apt-get install -y expect build-essential gfortran
# Install anaconda light
RUN mkdir -p /opt
WORKDIR /opt
ADD Miniconda*sh /opt/installer.sh
ADD expect.sh /opt/
# execute and clean
RUN cd /opt && ./expect.sh
# Add to path
ENV PATH $PATH:/opt/miniconda/bin
# Install minimal packages - reply with yes to install
RUN yes | conda create -n scientific ipython-notebook numpy scipy matplotlib cython
# Set up scripts for your needs
RUN echo "ipython notebook --ip=0.0.0.0 –port=8888 --no-browser" > /opt/start_notebook.sh
# notebook port
EXPOSE 8888
# Activate virtualenv
RUN rm /bin/sh && ln -s /bin/bash /bin/sh
RUN echo "source /opt/miniconda/bin/activate scientific" >> /root/.bashrc
```



Test di installazione

```
boot2docker:~$ docker run -it cineca/scientificpy bash
discarding /opt/miniconda/bin from PATH
prepending /opt/miniconda/envs/scientific/bin to PATH
(scientific):~$ python
(scientific)root@2abcd7c92c05:/opt# python
Python 2.7.8 |Continuum Analytics, Inc.| (default, Aug 21 2014,
   18:22:21)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more
   information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
>>>
```





Corso python

PRIMI PASSI: **INTERPRETE**

Il miglior modo per imparare il python è usarlo!

Interprete interattivo

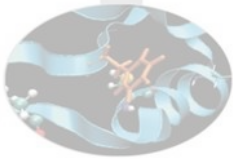
Avviato l'interprete appaiono tre caratteri "maggiore di" (>>>)

Questo è il prompt interativo dell'interprete di Python

Dagli esempi forniti non deve essere copiato

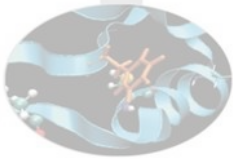
Il simbolo >>> indica che il Python è pronto per interpretare comandi

```
$ python
  Python 2.7.4
>>>
```



Ciao Mondo!

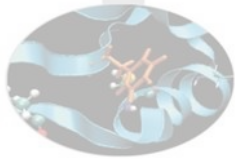
```
>>> print "Hello World!"
Hello World!
```



Ciao Mondo!

```
 >>> print "Hello World!"
Hello World!

>>> print "Hello World!";
Hello World!

>>> print("Hello World!")
Hello World!

>>> print("Hello World!");
Hello World!
```

Metodi di stampa



In **Python 2.x** la procedura di stampa (print) è considerata una semplice istruzione

Funziona passando un parametro (es. stringa, variabile) dopo la parola chiave print

```
>>> print "Hello World!"
Hello World!

>>> print "Hello World!" “test”
Hello World!test

>>> print "Hello World!","test"
Hello World! test
```

Metodi di stampa

In **Python 3.x** print è diventata a tutti gli effetti una funzione

Più flessibile

Ha un comportamento più generico

```
>>> print("Hello World!")
Hello World!

>>> print("Hello","World!")
Hello World!

>>> print("Hello","World!",sep=",")
Hello,World!
```

Richiedere un input

Funzione raw_input

Prende una stringa in interattivo

```
>>> name = raw_input("Enter your name: “)
Enter your name: Seba

>>> name
’Seba’
```

Richiedere un input

Funzione input

Richiede una stringa, ma cerca di valutarla come se fosse un oggetto Python

```
>>> name = input("Enter your name: ")
Enter your name: Seba
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "<string>", line 1, in <module>
NameError: name 'Seba' is not defined
```

Richiedere un input

Funzione input

Richiede una stringa, ma cerca di valutarla come se fosse un oggetto Python

```
>>> name = input("Enter your name: ")
Enter your name: "Seba"
>>> name
'Seba'
```

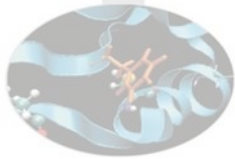
Richiedere un input

Funzione input

Richiede una stringa, ma cerca di valutarla come se fosse un oggetto Python

```
>>> test = "Prova“
>>> name = input("Enter your name: ")
Enter your name: test
>>> name
'Prova‘
>>>
```

Richiedere un input: Python 3



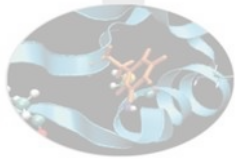
Non esiste più la funzione raw_input

La funzione input è adesso invece equivalente alla “vecchia” raw_input

Per emulare la “vecchia” input si può utilizzare la funzione eval

```
>>> name = input("Enter your name: ")
Enter your name: Seba
>>> name
'Seba'


>>> input("Operation: ")
Operation: 2+2
'2+2'


>>> eval(input("Operation: "))
Operation: 2+2
4
```

La calcolatrice in interattivo

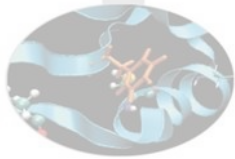


L'interprete interattivo è una comoda calcolatrice

```
>>> 1+1
    2
>>> 3*2
    6
>>> 4-7
    -3
>>> 9/4
    2


>>> 2**8/2+100
228
```

p.s. ** è il simbolo dell'elevamento a potenza

Duplice funzionalità



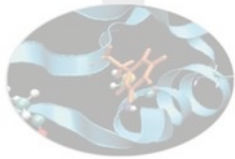
L'operatore + fa anche "somma di stringhe" ovvero la concatenazione

Singoli apici e doppi apici sono indifferenti per incapsulare stringhe

Il tipo dei dati coinvolti nell'operazione deve essere lo stesso!

```
>>> '1'+'1'
'11'
>>> "A string of " + 'characters'
'A string of characters'
>>> 'The answer is ' + 42
Traceback (most recent call last):
File "<stdin>", line 1, in ?
TypeError: cannot concatenate 'str' and
   'int' objects
>>> 'The answer is ' + str(42)
'The answer is 42'
```

Formattare stringhe



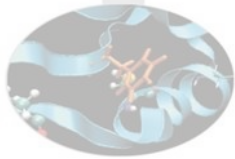
Sullo stile del linguaggio C esistono in Python delle operazioni di formattazione attraverso l'operatore "%" all'interno delle stringhe stesse

I valori possono essere assegnati a delle variabili.

Un pò come assegnare un nome a una cosa!

```
>>> 'The answer is ' + str(42)
'The answer is 42'

>>> 'The answer is %s'%42
'The answer is 42'

>>> n = 42
>>> 'The answer is %s'%n
'The answer is 42'
```

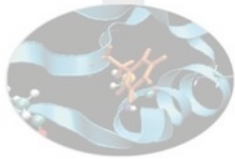
La divisione: Python 2



Come notato prima, la vecchia versione dell'operatore di divisione ha un atteggiamento diverso da quello atteso

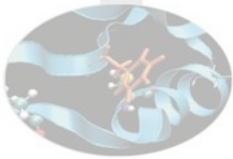
```
>>> 10/3
3


>>> 10.0/3
3.3333333333333335
>>> 10/3.
3.3333333333333335
```

La divisione: Python 3



Anche questa strana modalità è stata corretta nella nuova versione

```
>>> 10/3
3.3333333333333335
>>> 10/2
5.0


>>> 10//3
3
>>> 10//2
5
```

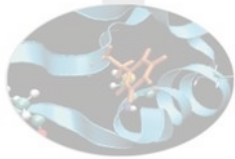


Esercizio su utilizzo interprete

- Richiedere in input un numero.
- Sommare uno al numero ottenuto.
- Moltiplicare il risultato per quattro.

(Effettuare le operazioni matematiche su una stessa riga.)

Exit



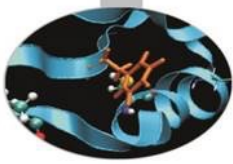
Per interrompere l'interprete possiamo usare la chiamata di sistema a exit()

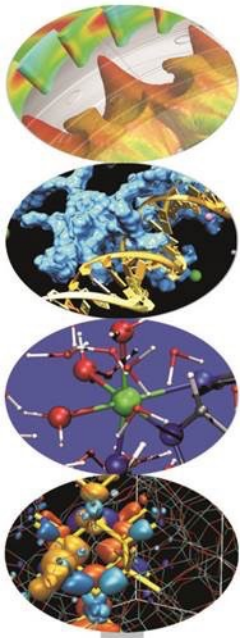
Non è necessaria dentro i file .py

Exit termina la nostra "sessione"

```
paulie@workstation:~$ python
Python 2.7.8
[GCC 4.7.3] on linux2

>>> exit()

paulie@workstation:~$
```

CINECA SCAI
SuperComputing Applications and Innovation



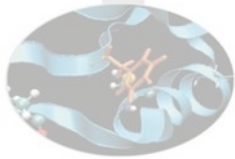


Corso python

PRIMI PASSI: **BATCH MODE**

Il miglior modo per imparare il python è usarlo!

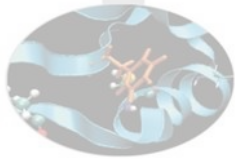
CINECA



Batch mode

- La modalità interattiva è chiaramente limitata
- I programmi di solito vengono salvati nei files
- Il codice interattivo è valido…
  - …solo finchè la sessione è aperta
  - Quando chiudiamo la sessione, il precedente codice viene perduto
- Per consentire la persistenza del codice i programmi vengono salvati in files di testo
- Un programma Python eseguito a partire da un file invece che dall'interprete fa uso della modalità batch.

Il nostro primo script



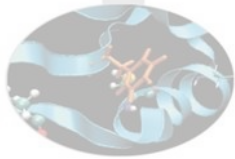
- Cartella *C:\Users\test* della vostra versione di Windows 7
- equivale a */Users/test* di Linux (notazione shell Unix)
- montata come */data* nel vostro container

C:\Users\test <-> /data

Windows <-> Container

Geane <-> Python interpreter

Il nostro primo file Python



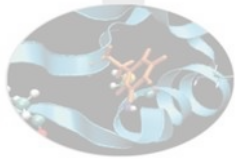
Contenuto del file

```
print("Hello World!")
```

Interprete

```
$ python hello.py
Hello World!
```

Il nostro primo file Python



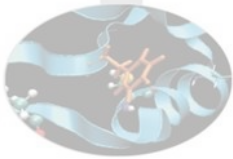
Contenuto del file

```
print("Hello World!")
```

Interprete

```
$ ./hello.py
./hello.py: line 1: syntax error
near unexpected token <=
''Hello world!''
./hello.py: line 1: 'print('Hello
world!')'
```

Il primo file Python standalone



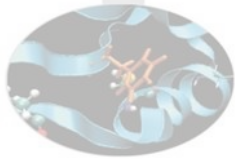
Contenuto del file

```
#!/usr/bin/python2.7
print("Hello World!")
```

Interprete

```
$ which python2.7
/usr/bin/python2.7

$ ./hello.py
Hello World!
```

Encoding comment



Contenuto del file

```
#!/usr/bin/python2.7
# -*- coding: latin1 -*-
print("Hello World!")

# Without encoding comment,
# Python's parser will assume #
ASCII
# It's the default encoding)
```

Interprete

```
$ ./hello.py
Hello World!
```

I commenti

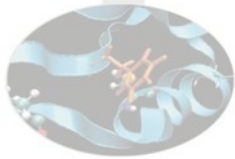
Contenuto del file

```
print("Hello World!")
#Stampa di una stringa
```

Interprete

```
$ python hello.py
Hello World!
```

Indentazione e leggibbilità



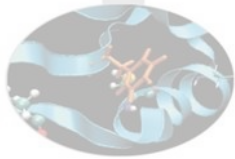
Non indentato

```
if (attr == -1){while (x<5){
printf("Waiting...\n");
wait(1);x = x+1;} printf("OK\n");}
else {printf("Error\n");}
```

Con indentazione

```
if (attr == -1) {
    while (x<5) {
        printf("Waiting...\n");
        wait(1);
        x = x+1;
    }
    printf("OK\n");
} else {
    printf("Error\n");
}
```

Indentazione e leggibbilità

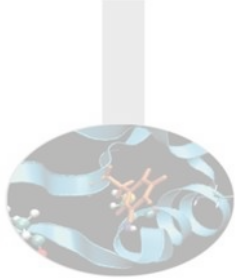


C

```
if (attr == -1) {
    while (x<5) {
        printf("Waiting...\n");
        wait(1);
        x = x+1;
    }
    printf("OK\n");
} else {
    printf("Error\n");
}
```

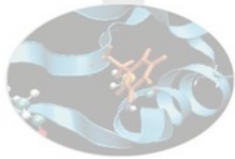
Python

```
if attr==-1:
        while x<5:
        print("Waiting...")
        wait(1)
    x = x + 1
    print("OK")
else
    print("Error")
```



Scegliere l'editor di testo

- Editor normali
  - Notepad
  - Mousepad
- Editor che conoscono la sintassi Python
  - Kate
  - Vim
  - Eclipse
- Editor pensati appositamente per il Python
  (auto completamento, debugger integrato, etc)
  - Dr Python
  - Eric
  - SPE
- Editor scritti in linguaggio Python :)
  - Sublime text 3

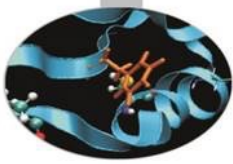


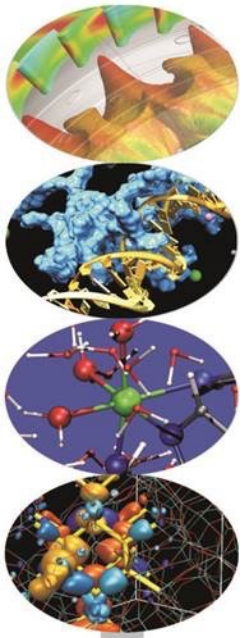
Esercizio su batch mode

- Richiedere in input un numero.
- Dividere il numero in input per cinque
- Moltiplicare il risultato per quattro.

(Richiesto il dettaglio dei numeri decimali dopo la virgola)

CINECA SCAI
SuperComputing Applications and Innovation

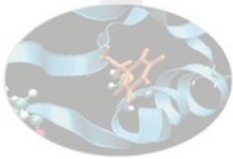




Corso python

Primi accenni sui moduli

CINECA



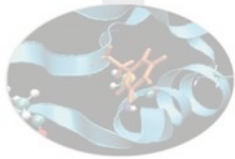
I moduli in Python

- La maggior parte delle funzionalità del Python sono fornite dai moduli
- La Python Standard Library è una collezione di moduli con le implementazioni multi-piattaforma delle operazioni più comuni
  - file I/O
  - stringhe

References

- The Python Language Reference: http://docs.python.org/2/reference/index.html
- The Python Standard Library: http://docs.python.org/2/library/

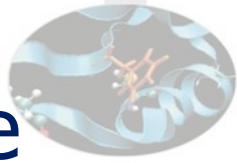
Importare un modulo



Per usare un modulo la prima cosa da fare è importarlo.

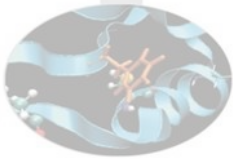
Per farlo utilizziamo l'operazione import

```
>>> import math

>>> x = math.cos(2 * math.pi)

>>> print(x)
1.0
```



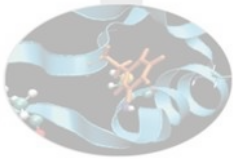
Import dei moduli: varie casistiche

```
>>> import sound.effects.echo.echofilter
>>> sound.effects.echo.echofilter(...)

>>> import sound.effects.echo
>>> sound.effects.echo.echofilter(...)

>>> from sound.effects import echo
>>> echo.echofilter(...)

>>> from sound.effects.echo import echofilter
>>> echofilter(...)
```



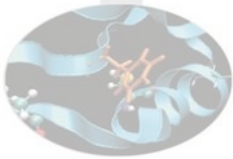
Moduli: help e documentazione

```
>>> help
   Type help() for interactive help, or help(object) for help
   about object.
   >>> help("string")


Help on module string:
   NAME
       string - A collection of string operations (most are no
   longer used).

   FILE
       /usr/lib/python2.7/string.py

   MODULE DOCS
       http://docs.python.org/library/string
```



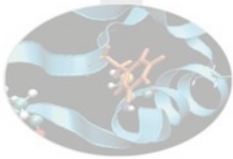
Moduli: help e documentazione

```
>>> help()

   Welcome to Python 2.7!  This is the online help utility.


help> keywords

   Here is a list of the Python keywords.  Enter any keyword to
   get more help.
[...]

   help> quit
>>>
```



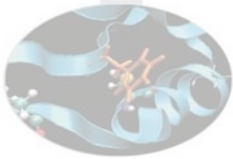
Moduli e introspezione

```
# it isn't as easy to remember
# what each module contains.
>>> import math
>>> dir(math)
   ['__doc__', '__name__', '__package__', 'acos', 'acosh',
   'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
   'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc',
   'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
   'frexp', 'fsum', 'gamma', 'hypot', 'isinf', 'isnan', 'ldexp',
   'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow',
   'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
>>>
```



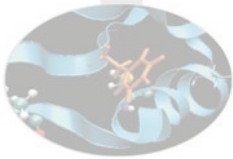
Moduli e introspezione

```
# What did i load so far?


>>> import math


>>> dir()
   ['__builtins__', '__doc__', '__name__', '__package__',
   'math']
```



Moduli e introspezione

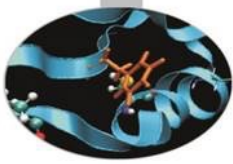
```
# What to do with math?
>>> import math
>>> dir(math)
   [... 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan',
   'tanh', 'trunc‘]
>>> help ("math.trunc")
math.trunc = trunc(...)
      trunc(x:Real) -> Integral

      Truncates x to the nearest Integral toward 0. Uses the
   __trunc__ magic method.


>>> math.trunc(5.676876)
   5
```

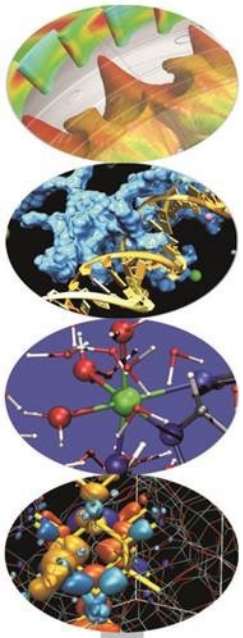


Esercizio su introspezione

Applicare il seno del logaritmo di 4

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

I TIPI DI DATO: **stringhe**

CINECA

Essere o non essere una stringa

Per apprendere informazioni su di un tipo possiamo definire una variabile e verificarla con la funzione type

```
>>> sconosciuto = "Una stringa"
>>> help("type")
Help on class type in module __builtin__:

    class type(object)
     |  type(object) -> the object's type
     |  type(name, bases, dict) -> a new type
     |

>>> type(sconosciuto)
   <type 'str'>


>>> type(42)
   <type 'int'>
```

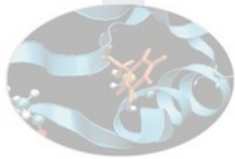
Essere o non essere una stringa

Una stringa è una sequenza di simboli delimitata da:

- single quote (’)
- double quotes (”)
- single triple quotes (”’)
- double triple quotes (”””)

```
#Sono equivalenti:
"This is a string in Python"
'This is a string in Python'
'''This is a string in Python'''
"""This is a string in Python"""


#Attenzione a quali apici usiamo
"A single quote (') inside a double quote"
'Here we have "double quotes" inside single
    quotes'
>>> "Mixing quotes leads to the dark side'
File "<stdin>", line 1
"Mixing quotes leads to the dark side'
^
SyntaxError: EOL while scanning single-quoted
    string
```

Codifiche in Python



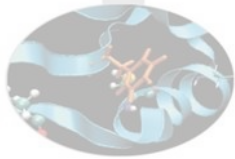
Python 2

```
#Byte strings
>>> s = u'Sebastiàn'
>>> s.encode('utf-8')
'Sebasti\xc3\xa1n'

#Unicode strings
>>> b=s.encode('utf-8')
>>> x=unicode(b,'utf-8')
>>> print x
Sebastiàn

#Equivalence
>>> x==s
True
```

Python 3

```
#Unicode by default
>>> 'Python 3, strings are
Unicode: ~n'
'Python 3, strings are Unicode:
~n'

#Byte strings
>>> b'Bytes in Python 3'
b'Bytes in Python 3'
```

Manipolazione di stringhe

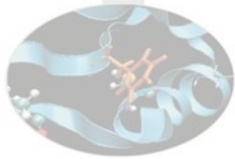


Facendo introspezione possiamo verificare le operazioni disponibili su una stringa.

Ecco alcuni esempi.

```
>>> maiuscole = "SONO UNA STRINGA"
>>> dir(maiuscole) #Cosa trovo?


>>> maiuscole.lower()
   'sono una stringa'
>>> maiuscole.lower().swapcase()
   'SONO UNA STRINGA'


>>> DNAseq="TTGCTAG"
>>> mRNAseq=DNAseq.replace("T","U")
>>> mRNAseq
'UUGCUAG'
```

Manipolazione di stringhe



Dividere una stringa,

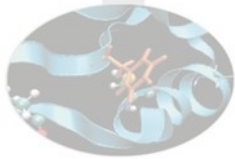
oppure ottenerne una

sotto-porzione

sono le due operazioni più comuni su questo tipo di dato!

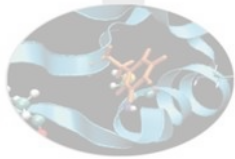
```
>>> maiuscole = "Sono Una Stringa“
>>> maiuscole.lower().split(" ")
   ['sono', 'una', 'stringa']
>>> x = "Hello World!“
>>> x[2:]
   'llo World!‘
>>> x[:2]
   'He‘
>>> x[:-2]
   'Hello Worl‘
>>> x[-2:]
   'd!‘
>>> x[2:-2]
   'llo Worl'
```

Manipolazione di stringhe



Cercate di capire cosa sta accadendo a questa stringa

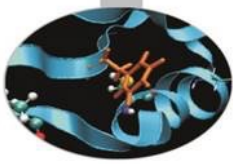
```
>>> stringa="dividimiintantipezzi"

>>> list(stringa)
   ['d', 'i', 'v', 'i', 'd', 'i',
   'm', 'i', 'i', 'n', 't', 'a', 'n',
   't', 'i', 'p', 'e', 'z', 'z', 'i']


>>> "".join(list(stringa))
   'dividimiintantipezzi'
```

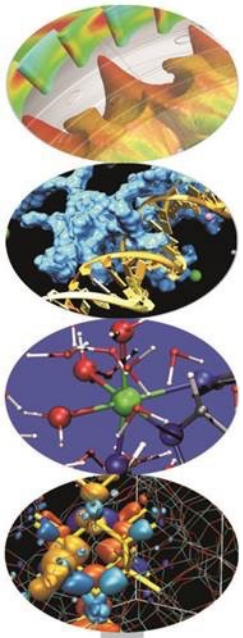


Esercizio su stringhe

- Creare una variabile stringa contenente tre spazi
- Dividere la stringa sul separatore spazio “ “
- Comporre dai pezzi ottenuti una nuova stringa usando le ultime due lettere di ogni pezzo

CINECA SCAI
SuperComputing Applications and Innovation





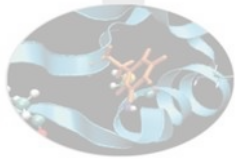
Corso python

I TIPI DI DATO: **liste**

“List Is the Workhorse Datatype in Python”

CINECA

Cosa contiene una lista



La lista è in Python l'equivalente del vettore

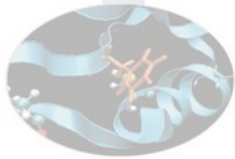
(o array) dei comuni linguaggi di programmazione.

Le liste possono contenere tipi differenti,

anche altre liste annidate

```
>>> elementi = "test stringa".split(" ")
>>> elementi
    ['test', 'stringa']
    >>> type(elementi)
    <type 'list'>
>>> elementi = [ 1, "uno" ]
>>> type(elementi)
    <type 'list'>


>>> elementi = [ 1, "uno", elementi ]
>>> elementi
    [1, 'uno', [1, 'uno']]


>>> lista_vuota = []
>>> lista_vuota
    []
```

Operazioni su una lista



Esempi di moltiplicazione con diversi obiettivi

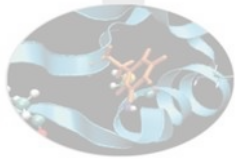
```
#Moltiplicare una lista
>>> ripetizione = ["Uno"] * 5
•  >>> ripetizione
      ['Uno', 'Uno', 'Uno', 'Uno', 'Uno']


#Moltiplicare una lista arimeticamente
>>> A = [0,1,2,3,4,5]
>>> [3*x for x in A]
   [0, 3, 6, 9, 12, 15]
```

Accedere agli elementi di una lista

Un elemento singolo può essere acceduto attraverso un indice numerico.

```
>>> lista = [ 1,2,3,4,5 ]


>>> lista[0]
   1
>>> lista[1]
   2


>>> lista[-1]
   5
>>> lista[-3]
   3
```

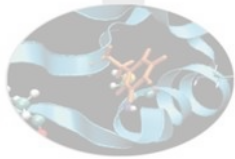
Modificare una lista



Inserire e rimuovere oggetti nella lista

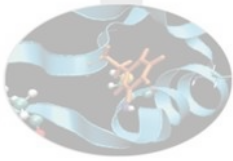
```
>>> lista.append("stringa")
>>> lista
   [1, 2, 3, 4, 5, 'stringa']
>>> lista.insert(3,3.5)
>>> lista
   [1, 2, 3, 3.5, 4, 5, 'stringa']


>>> lista.pop()
   'stringa‘
>>> lista.extend([6,7,8])
>>> lista
   [1, 2, 3, 3.5, 4, 5, 6, 7, 8]
>>> lista.remove(3)
   >>> lista
   [1, 2, 3.5, 4, 5, 6, 7, 8]
```

Copiare una lista



Attenzione al concetto di oggetto originale e di un suo riferimento

```
#Reference
>>> a=[1,2,3]
>>> b=a
>>> b.pop()
3
>>> a
[1, 2]
#Real copy
>>> a=[1,2,3]
>>> b=a[:]
>>> b.pop()
3
>>> a
[1, 2, 3]
```



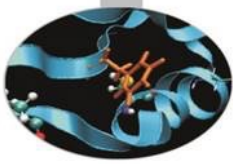
Esercizio su liste

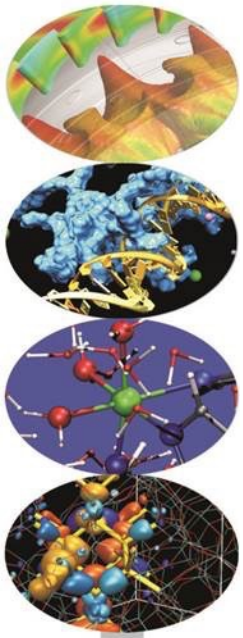
1. Definire la seguente lista

```
lista = [ 1, [7, 6], [8, [9, 4, 5] ] ,6, [ 0, 2] ]
```

Accedere all'elemento di valore '5'

2. Generare una lista fatti di dieci 1000 e un cinque nel mezzo

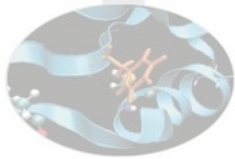




Corso python

I TIPI DI DATO: **tuple**

Cosa contiene una tupla

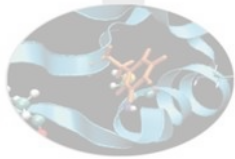


La tupla è un tipo di lista particolare.

Si definisce usando le parentesi tonde al posto delle quadre.

```
>>> point=(3,5,-6)


>>> point
(3, 5, -6)


>>> type(point)
<type 'tuple'>


>>>
```

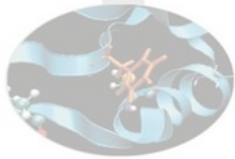
Differenze tra tupla e lista



La tupla non consente la modifica sul numero di elementi contenuti

```
>>> point.append(3)
   Traceback (most recent call last):
     File "<stdin>", line 1, in
   <module>
   AttributeError: 'tuple' object has
   no attribute 'append‘
>>> point.remove(5)
   Traceback (most recent call last):
     File "<stdin>", line 1, in
   <module>
   AttributeError: 'tuple' object has
   no attribute 'remove‘


>>> dir(point)
#Cosa ci trovo?
```

A cosa serve allora una tupla?



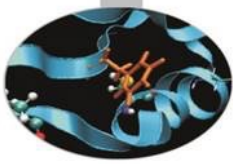
Esempio tipico:

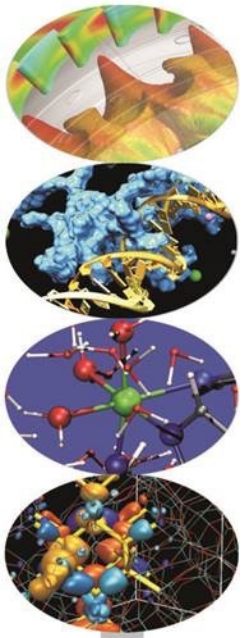
un sistema di coordinate.

Anche nei valori di ritorno di funzioni risultano molto comode.

```
>>> point = (3,5,-6)


>>> x, y, z = point
>>> print("x =", x)
   ('x =', 3)
>>> print("y =", y)
   ('y =', 5)
>>> print("z = ", z)
   ('z = ', -6)
```

CINECA SCAI
SuperComputing Applications and Innovation



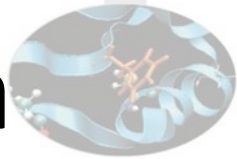


Corso python

I TIPI DI DATO: **sequenze**

CINECA

Cosa intendiamo con sequenza



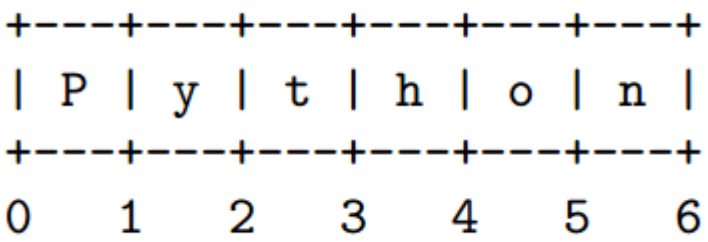
Possiamo applicare alcune operazioni (come l'indexing) ugualmente a stringhe, liste o tuple.

In questo caso li consideriamo insieme come uno stesso tipo chiamato sequenza.

```
>>> point=(23,56,11)
>>> point[0]
23
>>> point[1]
56
>>> sequence="MRVLLVALALLALAASATS"
>>> sequence[5]
'V'
>>> sequence[-3]
    'A'
>>> parameters = ['UniGene','dna','Mm.
    248907',5]
>>> parameters[2]
'Mm.248907'
```

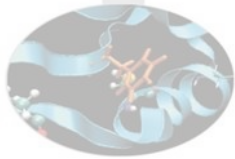
Sequenze e slicing

Pezzetti della sequenza “Python”




```
>>> my_sequence="Python"


>>> my_sequence[0:2]
    'Py‘
>>> my_sequence[:2]
    'Py‘
>>> my_sequence[4:6]
    'on‘
>>> my_sequence[4:]
    'on‘
>>> my_sequence[1:5]
    'ytho'
>>> my_sequence[1:5:2]
    'yh'
#Rovesciare la sequenza
>>> my_sequence[::-1]
    'nohtyP'
```

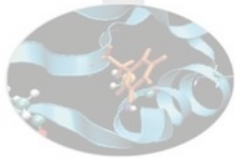
Sequenze e verifiche



La keyword in permette di verificare se un elemento è presente nella sequenza.

```
>>> point=(23,56,11)
>>> 11 in point
   True


>>> sequenza
   'MRVLLVALALLALAASATS‘
>>> 'X' in sequenza
   False
>>> 'Y' in sequenza
   False
>>> 'A' in sequenza
   True
```

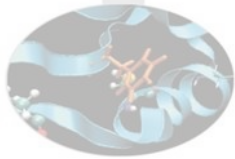
Concatenazione



Uniamo sequenze attraverso l'operatore +

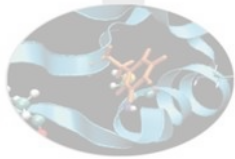
```
>>> point=(23,56,11)
>>> point2=(2,6,7)
>>> point+point2
   (23, 56, 11, 2, 6, 7)
>>> DNAseq = "ATGCTAGACGTCCTCAGATAGCCG"
>>> TATAbox = "TATAAA"
>>> TATAbox+DNAseq
   'TATAAAATGCTAGACGTCCTCAGATAGCCG‘
>>> point+TATAbox
   Traceback (most recent call last):
     File "<stdin>", line 1, in <module>
   TypeError: can only concatenate tuple (not
   "str") to tuple
```

Dimensioni delle sequenze



len, max e min consentono di gestire le dimensioni delle sequenze

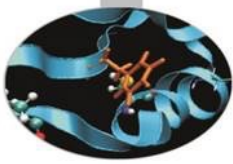
```
>>> len(sequenza)
   19
>>> len(point)
   3
>>> max(sequenza)
   'V‘
>>> max(point)
   56
>>> min(sequenza)
   'A‘
>>> min(point)
   11
```

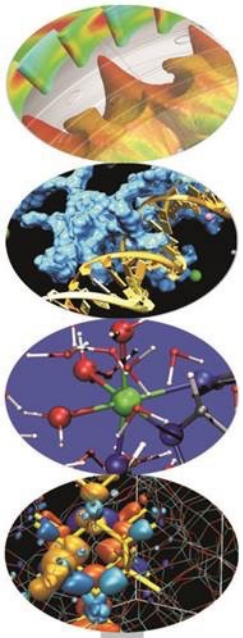


Esercizio sulle sequenze

- Definire una stringa e un punto di un asse cartesiano
- Creare e stampare una nuova stringa composta da:
  - La lettera più piccola della stringa
  - La stringa rovesciata
  - Il valore più grande del punto

CINECA SCAI
SuperComputing Applications and Innovation

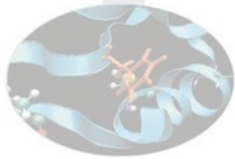




Corso python

I TIPI DI DATO: **dizionari**

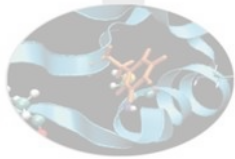
CINECA



I dizionari

Il dizionario è una collezione che associa coppie di tipo chiave : valore

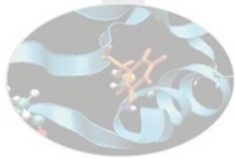
```
>>> grammatica={ 'A':'Verbo', 'B':'Sostantivo', 'C':'Aggettivo'}


>>> grammatica
   {'A': 'Verbo', 'C': 'Aggettivo', 'B': 'Sostantivo'}


>>> grammatica['C']
   'Aggettivo'


>>> type(grammatica)
   <type 'dict'>
```



I dizionari

- Chiamare ogni valore della collezione con un nome (al posto di un indice numerico).
- Non mi preoccupo dell'ordinamento.
- In altri linguaggi può essere chiamato Hash, oppure Array Associativo.
- E' uno dei tipi più potenti
  - ma non necessariamente quello adatto ad ogni casistica!

Azioni semplici su dizionari

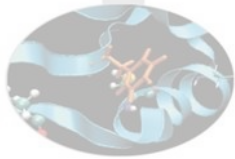


I dizionari sono coppie di chiavi e valori.

Possiamo operari sui due componenti separatamente.

```
>>> grammatica={ 'A':'Verbo',
   'B':'Sostantivo', 'C':'Aggettivo'}


>>> grammatica.keys()
   ['A', 'C', 'B‘]
>>> grammatica.values()
   ['Verbo', 'Aggettivo', 'Sostantivo']
>>> 'A' in grammatica
   True
   >>> grammatica.has_key('A')
   True
>>> 'Verbo' in grammatica
   False
>>> 'Verbo' in grammatica.values()
   True
```

Azioni semplici su dizionari

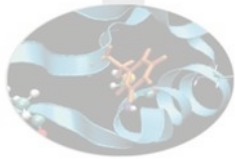


I dizionari sono coppie di chiavi e valori.

Possiamo operare sulle coppie singolarmente con la funzione items.

```
>>> grammatica.items()
   [('A', 'Verbo'), ('C', 'Aggettivo'),
   ('B', 'Sostantivo')]


>>> grammatica_lista = grammatica.items()


>>> grammatica_lista[0]
   ('A', 'Verbo')
>>> grammatica_lista[2]
   ('B', 'Sostantivo')
```

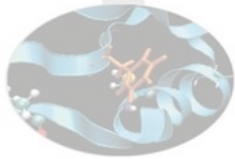
Gestione di un elemento



La funzione get permette di ottenere un valore a partire da una chiave.

```
#Equivalenza di accesso
>>> grammatica['B']
   'Sostantivo'
>>> grammatica.get('B')
   'Sostantivo'


#Get e' più potente
>>> grammatica['Z']
   Traceback (most recent call last):
     File "<stdin>", line 1, in <module>
   KeyError: 'Z'
>>> grammatica.get('Z‘)
>>> grammatica.get('Z','Nessun elemento')
   'Nessun elemento'
```

Gestione di un elemento



La funzione del permette di rimuovere una coppia a partire da una chiave del dizionario.

```
>>> grammatica
   {'A': 'Verbo', 'C': 'Aggettivo', 'B':
   'Sostantivo'}


>>> del grammatica['A']


>>> grammatica
   {'C': 'Aggettivo', 'B': 'Sostantivo'}
```

Proprietà dei dizionari: un elenco

Proprietà	Descrizione
len(a)	Number of elements of *a*
a[k]	The element from *a* that has a *k* key
a[k] = v	Set a[k] to *v*
del a[k]	Remove a[k] from *a*
a.clear()	Remove all items from *a*
a.copy()	A copy of *a*
k in a	True if a has a key *k*, else False
k not in a	Equivalent to not *k* in *a*
a.has_key(k)	Equivalent to *k* in *a*, use that form in new code
a.items()	A copy of *a*'s list of (key, value) pairs
a.keys()	A copy of *a*'s list of keys
a.update([b])	Updates (and overwrites) key/value pairs from *b*
a.fromkeys(seq[, value])	Creates a new dictionary with keys from *seq* and values set to *value*
a.values()	A copy of *a*'s list of values
a.get(k[, x])	a[k] if *k* in *a*, else *x*
a.setdefault(k[, x])	a[k] if *k* in *a*, else *x* (also setting it)
a.pop(k[, x])	a[k] if *k* in *a*, else *x* (and remove *k*)
a.popitem()	Remove and return an arbitrary (key, value) pair

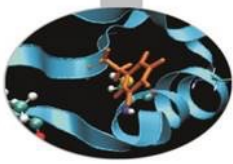
Dizionari in python 3: dict views

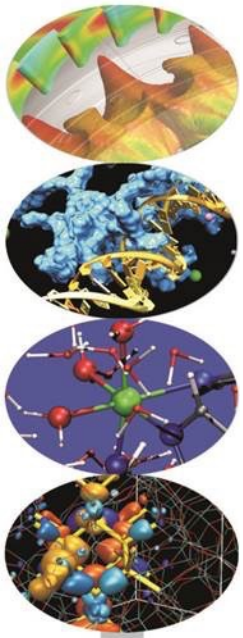
Non abbiamo più la lista come output di default nella gestione dei dizionari.

Le dict_views sono sincronizzate con l'oggetto originale

```
>>> grammatica={ 'A':'Verbo', 'B':'Sostantivo',
    'C':'Aggettivo'}

#Nuovo tipo di oggetto dict_
    >>> grammatica.values()
    dict_values(['Verbo', 'Sostantivo', 'Aggettivo'])

>>> grammatica.keys()
    dict_keys(['A', 'B', 'C'])

#Iterazione come una lista

>>> for i in grammatica:
    ...     print(i)
    A
    B
    C

#Possiamo ottenere sempre una lista se serve

>>> list(grammatica.values())
    ['Verbo', 'Sostantivo', 'Aggettivo']
```

CINECA SCAI
SuperComputing Applications and Innovation

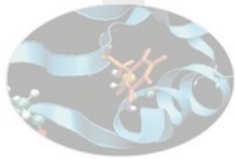




Corso python

I TIPI DI DATO: **gli insiemi**

CINECA

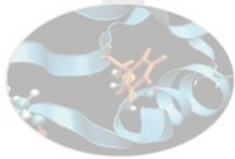


Gli insiemi

Struttura comunemente trovata in matematica.

Non molto tipica dei linguaggi di programmazione.

```
>>> cestino = ['mela', 'pera', 'mela', 'arancia', 'arancia','mela']


>>> frutta = set(cestino)
#Un insieme non contiene elementi duplicati
#e non ha un ordine
>>> frutta
   set(['pera', 'mela', 'arancia'])
>>> type(frutta)
   <type 'set'>
```



Gli insiemi

Operazioni base

```
>>> frutta = set()                    #INIZIALIZZAZIONE
    >>> frutta.add('mela')
    >>> frutta.add('pera')
    >>> frutta.add('arancia')
    >>> frutta
    set(['pera', 'mela', 'arancia'])
>>> frutta.add('mela')      #NIENTE DUPLICATI
    >>> frutta
    set(['pera', 'mela', 'arancia'])
>>> frutta.remove('mela')  #RIMOZIONE
    >>> frutta
    set(['pera', 'arancia'])
    >>> frutta.remove('mela')
    Traceback (most recent call last):
      File "<stdin>", line 1, in <module>
    KeyError: 'mela'
```

Insiemi: esistenza di un elemento

L'insieme consente anche delle operazioni gia' incontrate in altre strutture dati.

```
#Lettere univoche di una stringa
>>> a = set('abracadabra')
>>> a
   set(['a', 'r', 'b', 'c', 'd'])
>>> list(a)
   ['a', 'r', 'b', 'c', 'd']


>>> b = set('alacazam')
   >>> b
   set(['a', 'c', 'z', 'm', 'l'])
>>> 'b' in a
   True
>>> 'b' in b
   False
>>> 'b' not in b
```

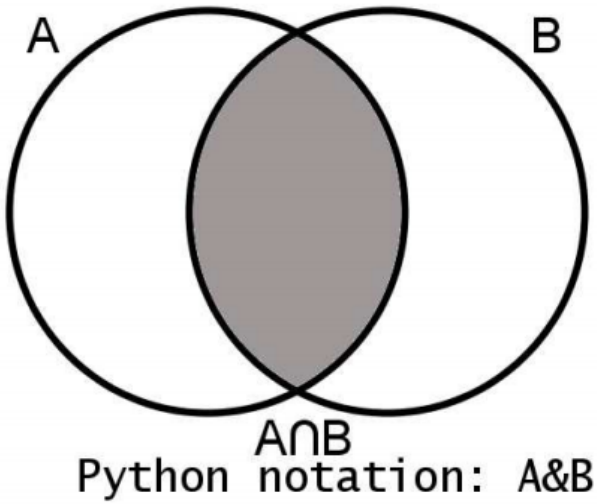
Insiemi: le operazioni fondamentali

L'insieme ha le potenzialità che conosciamo nella stessa matematica.

Questo tipo di operazioni non sono facili su strutture dati più comuni.

```
>>> a = set('abracadabra')
>>> a
    set(['a', 'r', 'b', 'c', 'd'])
    >>> b = set('alacazam')
    >>> b
    set(['a', 'c', 'z', 'm', 'l'])
>>> a - b
    set(['r', 'b', 'd'])
    >>> b - a
    set(['z', 'm', 'l'])
>>> a | b          #OR
    set(['a', 'c', 'b', 'd', 'm', 'l', 'r', 'z'])
>>> a & b          #AND
    set(['a', 'c'])
>>> a ^ b
    set(['b', 'd', 'm', 'l', 'r', 'z'])
```

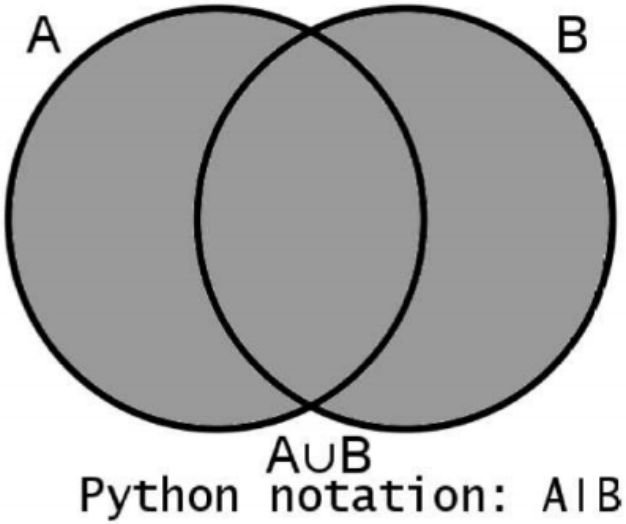
Insiemi: intersezione



Python notation: A&B

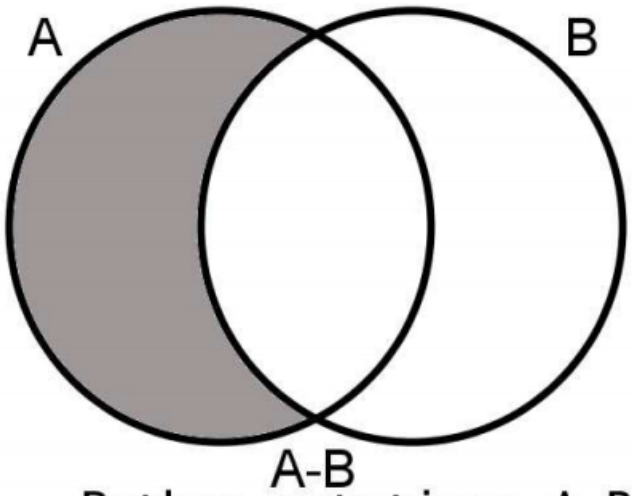
```
>>> a = set('abracadabra')
>>> a
   set(['a', 'r', 'b', 'c', 'd'])
   >>> b = set('alacazam')
   >>> b
   set(['a', 'c', 'z', 'm', 'l'])


>>> a & b
   set(['a', 'c'])
```

Insiemi: unione




```
>>> a = set('abracadabra')
>>> a
   set(['a', 'r', 'b', 'c', 'd'])
>>> b = set('alacazam')
>>> b
   set(['a', 'c', 'z', 'm', 'l'])


>>> a | b
set(['a', 'c', 'b', 'd', 'm', 'l', 'r', 'z'])
```

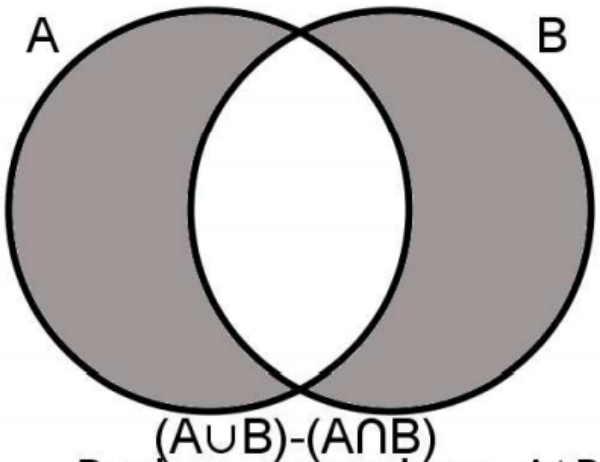
Insiemi: differenza



Python notation: A-B

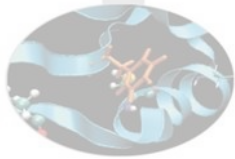
```
>>> a = set('abracadabra')
>>> a
   set(['a', 'r', 'b', 'c', 'd'])
>>> b = set('alacazam')
>>> b
   set(['a', 'c', 'z', 'm', 'l'])


>>> a.difference(b)
   set(['r', 'b', 'd'])
>>> a - b
   set(['r', 'b', 'd'])
```

Insiemi: differenza simmetrica



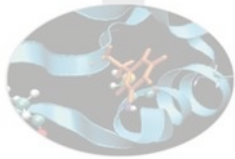

```
>>> a = set('abracadabra')
>>> a
   set(['a', 'r', 'b', 'c', 'd'])
>>> b = set('alacazam')
>>> b
   set(['a', 'c', 'z', 'm', 'l'])


>>> a ^ b
   set(['b', 'd', 'm', 'l', 'r', 'z'])
>>> (a | b) - (a & b)
   set(['b', 'd', 'm', 'l', 'r', 'z'])
```

Insiemi in python 3



Essenzialmente cambia soltanto la sintassi di rappresentazione, attraverso le parentesi graffe.

```
>>> frutta = {'mela','pera','arancia'}


>>> frutta
   {'arancia', 'mela', 'pera'}


>>> type(frutta)
   <class 'set'>


#Da una lista
>>> frutta = ['mela','pera','arancia']
>>> set(frutta)
   {'arancia', 'mela', 'pera'}
```



Esercizio sugli insiemi

Trovare le lettere che si trovano

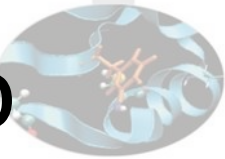
sia nella stringa “fruttato” che in “affondare”

e che *contemporaneamente* **non** siano presenti

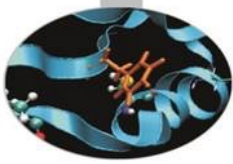
nelle lettere per cui differiscono le stringhe

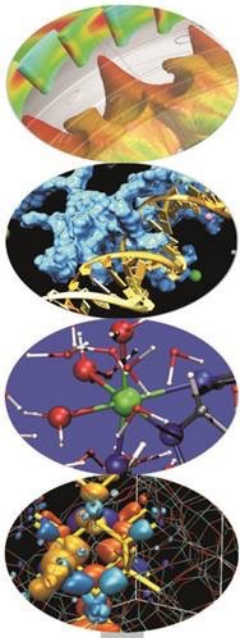
“maturando” e “famelico”.

Python 2: sommario tipi di dato



1. Stringhe `'stringa'`
2. Liste `[0, 1, 2]`
3. Tuple `(0, 1, 2)`
4. Dizionari `{'A': 0, 'B': 1}`
5. Insiemi `set([0, 1, 2])`

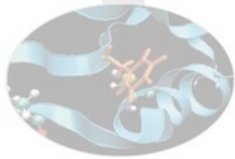




Corso python

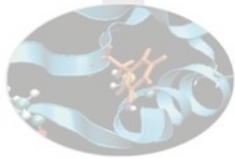
Assegnazione variabili
VS
Binding di un nome ad un oggetto

Passaggio per valore

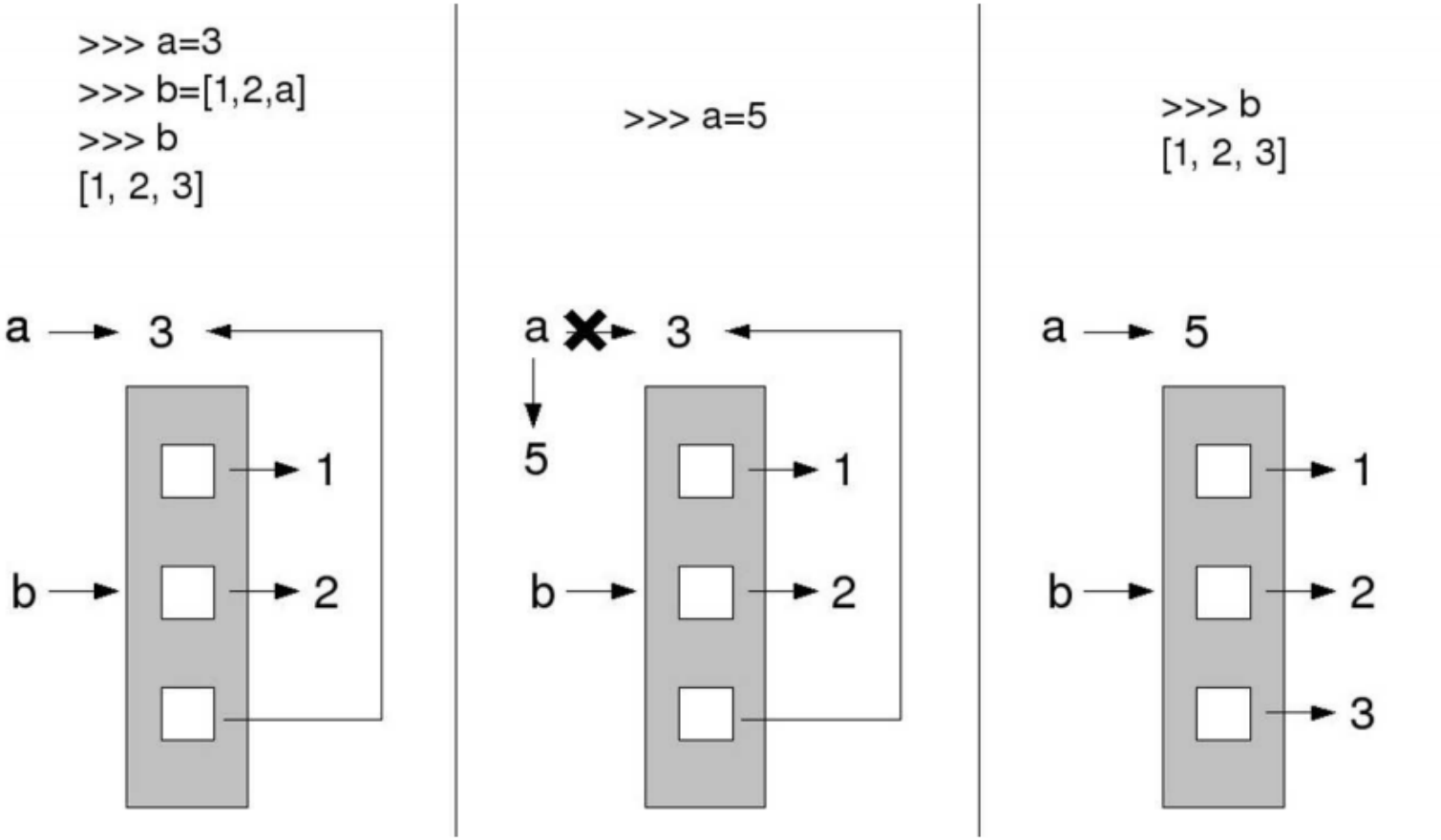


Salvataggio del valore di a nella lista b

```
>>> a = 3
>>> b = [ 1, 2, a ]
>>> b
[1, 2, 3]

>>> a = 5
>>> b
[1, 2, 3]
```



Passaggio per valore



Binding di un nome a un oggetto

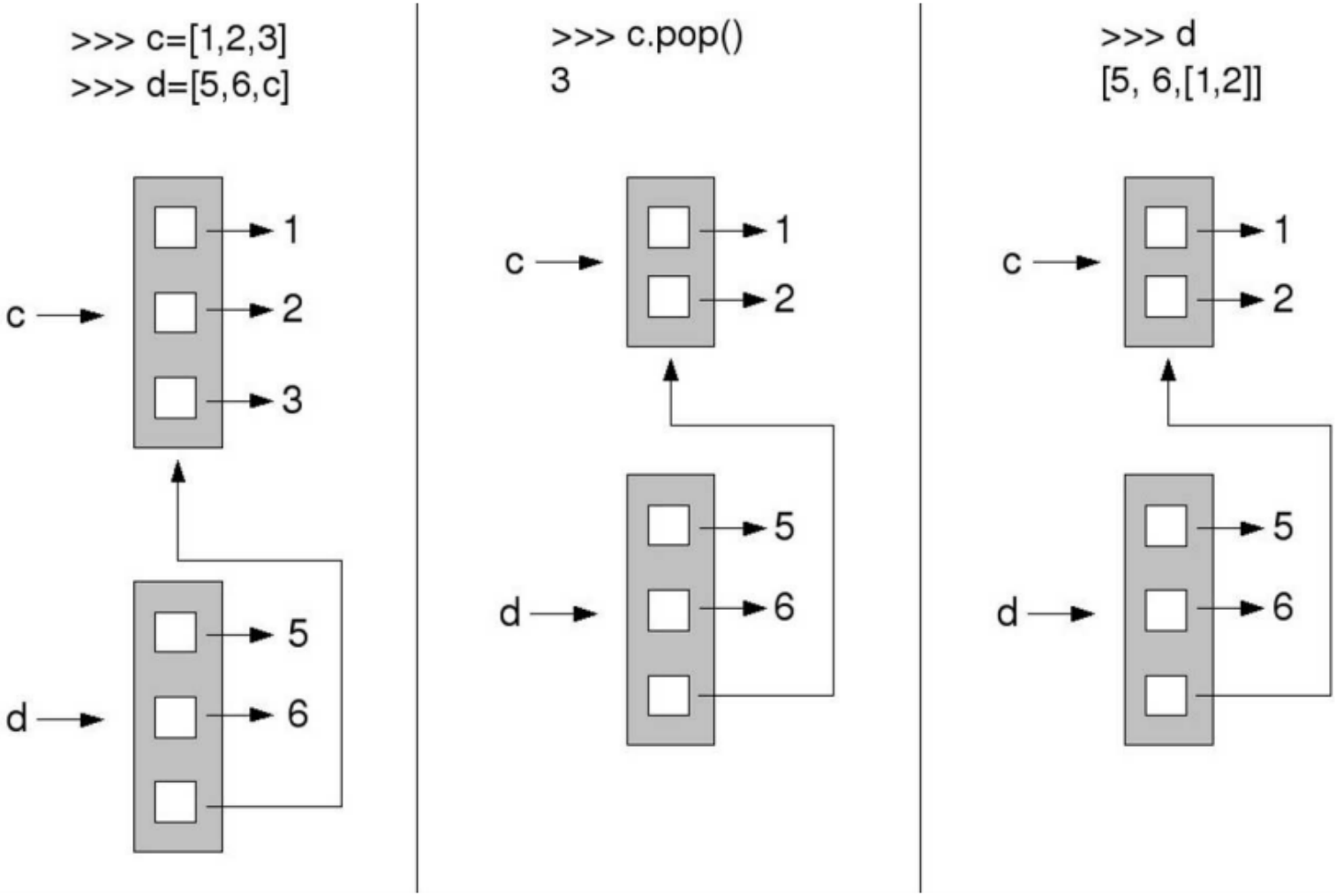
Salvataggio

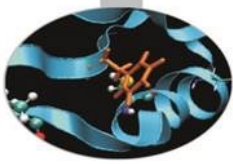
della lista c

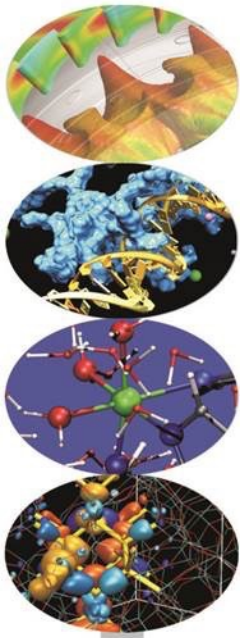
nella lista d

```
>>> c = [ 1, 2, 3 ]
   >>> d = [ 5, 6, c ]
   >>> d
   [5, 6, [1, 2, 3]]


>>> c.pop()
   3
>>> c
   [1, 2]


>>> d
   [5, 6, [1, 2]]
```

Binding di un nome a un oggetto

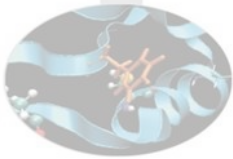






Corso python

Controllo del flusso



Controllo del flusso 1: *if then else*

Esecuzione di un blocco adeguato in base alla condizione che si viene a verificare.

```
if EXPRESSION1:
      Block1
elif EXPRESSION2:
      Block2
elif EXPRESSION3:
      Block3
else:
      Block4
```

Controllo del flusso 1: if then else

Effettuare una scelta in base a una condizione.

La condizione è indicata da una espressione di python.

```
>>> test = "pippo“
>>> "p" in test
   True
>>> "z" in test
   False
>>> if "p" in test:
...     print("Contiene almeno una 'p'")
... else:
...     print("Non contiene nessuna 'p'")
...
Contiene almeno una 'p'
```

Controllo del flusso 1: if then else

Possiamo indicare anche due o più condizioni nello stesso blocco.

Esse vengono eseguite in ordine.

```
>>> test = "pippo"
>>> if "z" in test:
...     print("Contiene una 'z'")
... elif "p" in test:
...     print("Contiene una 'p'")
... else:
...     print("Non contiene 'z' o 'p'")
...


Contiene una 'p'
```

Controllo del flusso 1: if then else

Le condizioni possono essere annidate

```
>>> test = "pippo"

>>> if "z" in test:
...     print("Contiene una 'z'“)
... elif "p" in test:
...     print("Contiene una 'p'“)
...     if "i" in test:
...             print("Contiene una 'i' oltre a 'p'“)
... else:
...     print("Non contiene ‘p’ o 'z'“)
...
Contiene una 'p'
Contiene una 'i' oltre a 'p'
```

Controllo del flusso 1: if then else

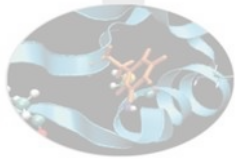
Le condizioni possono essere multiple nello stesso sotto-blocco.

```
>>> test = "pippo"


#N.B. else non e' obbligatorio
>>> if "z" in test or "p" in test:
...     print("Contiene 'p' o 'z'")
...
Contiene 'p' o 'z'
>>> if "z" in test and "p" in test:
...     print("Contiene sia 'p' che 'z'“)
... Else:
...     print("Non contiene sia 'p' che 'z'“)
...
Non contiene sia 'p' che 'z'
```

Controllo del flusso 1: if then else

Quando non è previsto l'else, possiamo usare la keyword pass per rendere il codice più leggibile

```
>>> test = "pippo"

>>> if "z" in test or "p" in test:
...     print("Contiene 'p' o 'z'“)
... else:
...     pass
Contiene 'p' o 'z'

>>> if "z" in test and "p" in test:
...     print("Non contiene sia 'p' che 'z'“)
... else:
...     pass
#non stampa nulla
```



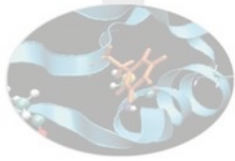
Controllo del flusso 2: *for*

Ripetizione di un blocco in base alle occorrenze di un oggetto iterabile.

```
for VAR in ITERABLE:
    BLOCK
```

Ad ogni iterazione VAR diventa l'elemento corrente dell'oggetto iterabile.

Controllo del flusso 2: for

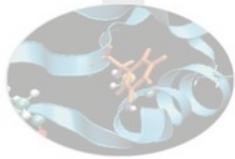


Un esempio semplice:

effettuare una stampa di una lista.

```
>>> for i in [ 1,2,3,4 ]:
...     print i
...
   1
   2
   3
   4
>>> temp = [ 1,2,3,4 ]
>>> for i in temp:
...     print i
...
   1
   2
   3
   4
```

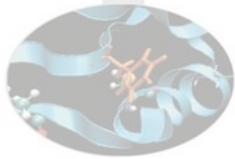
Controllo del flusso 2: for



Per iterare una lista su un intervallo di interi esiste la funzione range

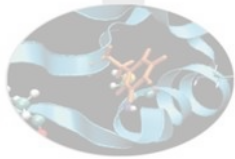
```
>>> for i in range(4):
...     print i
...
   0
   1
   2
   3
>>> for i in range(1,5):
...     print i
...
   1
   2
   3
   4
```

Controllo del flusso 2: for



Una stringa è comunque una lista di caratteri. Perciò:

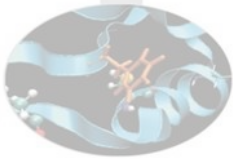
```
>>> for i in "abracadabra“:
...     print i
...
   a
   b
   r
   a
   c
   a
   d
   a
   b
   r
   a
```



Esercizio sui flussi

Stampare gli interi compresi tra 10 e 100

che sono divisibili contemporaneamente per 7 e per 2.



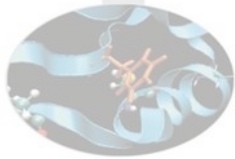
Controllo del flusso 3: *while*

Ripetizione di un blocco finché una condizione è vera.

```
while EXPRESSION:
      BLOCK
```

L'espressione indica la condizione da verificare.

Controllo del flusso 3: while

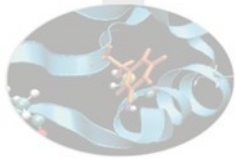


Un esempio semplice:

effettuare una stampa di numeri.

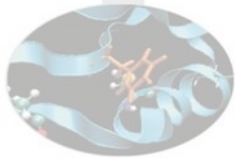
```
>>> i = 0


>>> while i < 5:
...     i = i + 1
...     print i
...
   1
   2
   3
   4
   5
```

Controllo del flusso 3: while



Un ciclo può essere interrotto con la keyword break

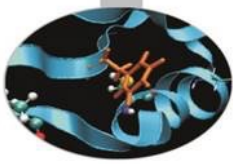
```
>>> i = 0
>>> while True:
...     i = i +1
...     print i
...     if i >= 5:
...             Break
...     Else:
...             Pass
   1
   2
   3
   4
   5
```



Controllo del flusso

Riassunto:

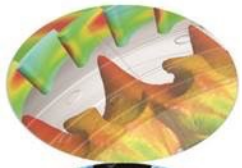
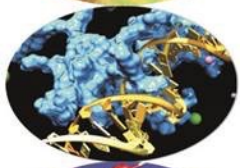
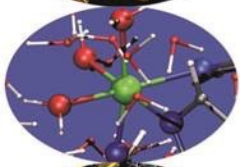
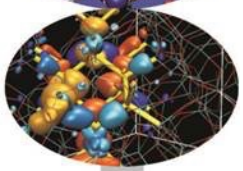
1. Condizioni verificabili con If then else
2. Ciclare su iterazioni attraverso il for
3. Elaborare operazioni fino a cambiare una certa condizione attraverso il while

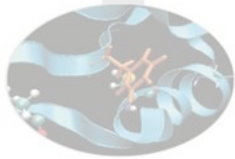


Primo giorno: Esercitazione finale

La massima del giorno:

più introspezione

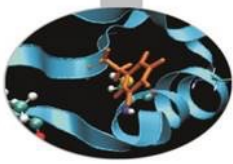







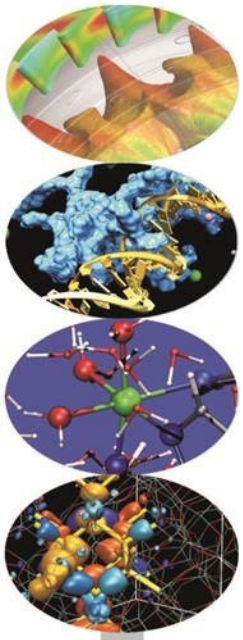
Esercizio su… tutto!

Generare numeri casuali compresi tra 100 e 500 finché non se ne trova uno divisibile per 13 e la cui seconda cifra sia un 5.

Hint: casuale in inglese si scrive random



Introduzione: fine prima parte



Bravi