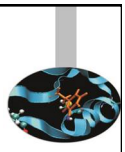
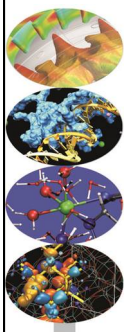
CINECA SCAI
SuperComputing Applications and Innovation



Python: introduzione al linguaggio

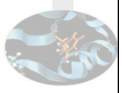


secondo giorno

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

CINECA

CINECA SCAI
SuperComputing Applications and Innovation



Prima parte: Agenda

Martedì 16/9/2013 Ore 10-13	**Introduzione a Python**
Martedì 16/9/2013 Ore 14-17	***Strutture dati e flussi di controllo***
Mercoledì 17/9/2013 Ore 10-13	***I/O, funzioni e moduli***

CINECA SCAI
SuperComputing Applications and Innovation

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**

3

CINECA SCAI
SuperComputing Applications and Innovation

Seconda parte: Agenda

Mercoledì 18/9/2013 Ore 14-17	**Introduzione a *NumPy***
Giovedì 19/9/2013 Ore 10-13	***Matplotlib*: il modulo *pylab*** ***Introduzione a SciPy***
Giovedì 19/9/2013 Ore 14-17	***mixed language programming***

4

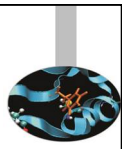
CINECA SCAI
SuperComputing Applications and Innovation

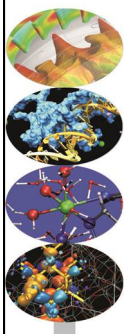
Seconda parte: Sommario

1. **Introduzione (con *overview* di IPython)**
2. **Introduzione a *NumPy***
   1. **L'oggetto *NDarray***
   2. **Operazioni su array**
3. ***Matplotlib*: il modulo *pylab***
4. **Introduzione a *SciPy***
5. **Performance in Python: *mixed language programming***
   1. **Introduzione**
   2. ***F2PY***
   3. ***Cython***
   4. ***Wave equation*: un caso reale**

5

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

Utilizzo di I/O

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

Esplorare un file: concetti e *parsing*

Ipotizziamo di avere un file composto da linee del tipo:

```
1,Joe,Doe,1976
```

Python legge le righe del file, che sono stringhe.

Bisogna applicare un lavoro in più per ottenere e gestire i dati di ogni linea:

parsing

CINECA SCAI
SuperComputing Applications and Innovation

Leggere un file

1. Aprire il file
   a. Utilizzo di un *filehandler*
   b. Il filehandler viene creato dalla funzione **open**
   c. Attenzione al path relativo / assoluto
2. Lettura del contenuto
   a. Di solito basandoci su un ciclo
3. Chiusura del file
   a. Liberare le risorse, chiudendo il *filehandler*

CINECA SCAI
SuperComputing Applications and Innovation

Esempio di file

0.0 0.0 0.0
1.0 0.0 0.0
1.0 1.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
1.0 0.0 1.0
1.0 1.0 1.0
0.0 1.0 1.0

CINECA SCAI
SuperComputing Applications and Innovation

Leggere un file: il codice

1. Aprire
2. Leggere
3. Chiudere

```
>>> fh=open('points.txt')
>>> fh
<open file 'points.txt', mode 'r' at 0xb748df98>

>>> fh.read()
'0.0 0.0 0.0\n1.0 0.0 0.0\n1.0 1.0 0.0\n0.0 1.0 0.0\n0.0 0.0
1.0\n1.0 0.0 1.0\n1.0 1.0 1.0\n0.0 1.0 1.0\n'

>>> fh.close()
>>> fh
<closed file 'points.txt', mode 'r' at 0xb748df98>
```

CINECA SCAI
SuperComputing Applications and Innovation

Leggere un file: riga per riga

1. Aprire
2. Leggere
3. Chiudere?

```
>>> fh=open('points.txt')
>>> while True:
...     line = fh.readline()
...     if line=="":
...         break
...     print line.replace('\n','')
...

0.0 0.0 0.0
1.0 0.0 0.0
1.0 1.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
1.0 0.0 1.0
1.0 1.0 1.0
0.0 1.0 1.0
```

CINECA SCAI
SuperComputing Applications and Innovation

Leggere un file: parsing

1. Leggere
2. Chiudere

```
>>> fh.seek(0,0)
>>> lista = list()
>>> lista
[]

>>> while True:
...     line = fh.readline()
...     if line=="":
...         break
...     lista.append(line.replace('\n','').split(' '))
...

>>> lista
[['1.0', '0.0', '0.0'], ['0.0', '1.0', '0.0'], ['1.0', '0.0', '1.0'], ['0.0', '1.0',
'1.0']]
>>> lista[1]
['0.0', '1.0', '0.0']
>>> fh.close()
```

CINECA SCAI
SuperComputing Applications and Innovation

Il formato CSV

- *CSV* sta per "**C**omma **S**eparated **V**alues"
- Si tratta di files con dati separati da virgole
  - si possono usare anche altri separatori
    - es punto e virgola, tab, ecc.
- Ogni riga rappresenta un *record diverso*
  - La prima riga indica solitamente i nomi dei campi
- Tutti i *fogli di calcolo* possono essere rappresentati con questo formato

CINECA SCAI
SuperComputing Applications and Innovation

Il modulo CSV di Python

1. Import
2. Handler
3. Iterazioni

```
>>> import csv

>>> lista = list()

>>> lines = csv.reader(open('points.csv'))
>>> lines.next()
['X', 'Y', 'Z']

>>> for line in lines:
...     lista.append(line)
...
>>> lista
[['0.0', '0.0', '0.0'], ['1.0', '0.0', '0.0'], ['1.0', '1.0', '0.0'], ['0.0', '1.0',
'0.0'], ['0.0', '0.0', '1.0'], ['1.0', '0.0', '1.0'], ['1.0', '1.0', '1.0'], ['0.0',
'1.0', '1.0']]
```

CINECA SCAI
SuperComputing Applications and Innovation

Il formato XLS

#Un diverso delimitatore?

```
>>> rows = csv.reader(open("passwd"), delimiter=':')
```

#Il dialetto XLS

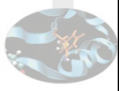
```
>>> rows = csv.reader(open("data.csv"), dialect='excel')
```

#Indovinare il dialetto

```
dialect = csv.Sniffer().sniff(open('data.csv').read())
rows = csv.reader(open("data.csv"), dialect=dialect)
```

CINECA SCAI
SuperComputing Applications and Innovation

Scrivere un file



1. Aprire il file **in modalità scrittura**
   a. Utilizzo di un *filehandler*
   b. Il filehandler viene creato dalla funzione **open**
   c. Attenzione al path relativo / assoluto
2. Scrittura del contenuto
   a. Di solito a partire da una struttura dati
3. Chiusura del file
   a. Liberare le risorse, chiudendo il *filehandler*

CINECA SCAI
SuperComputing Applications and Innovation

Scrivere un file

#Come si apre un file in scrittura?

```
>>> help('open')
    Help on built-in function open in module __builtin__:

    open(...)
      open(name[, mode[, buffering]]) -> file object

      Open a file using the file() type, returns a file object.  This is the
      preferred way to open a file.  See file.__doc__ for further information.
```

#Quali sono i *mode* a disposizione?

```
>>> print(file.__doc__)
    file(name[, mode[, buffering]]) -> file object

    Open a file.  The mode can be 'r', 'w' or 'a' for reading (default),
    writing or appending.  The file will be created if it doesn't exist
    when opened for writing or appending; it will be truncated when
    opened for writing.
```

CINECA SCAI
SuperComputing Applications and Innovation

Scrivere un file

1. Aprire
2. Scrivere
3. Chiudere

#Il file viene creato

```
>>> fo = open("/tmp/foo.txt", "w")
```

#Buffer di scrittura

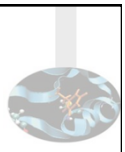
```
>>> fo.write( "Python is a great language.\nYeah its great!!\n");
```

#Il contenuto viene riversato nel file aperto

```
>>>  fo.close()
```

CINECA SCAI
SuperComputing Applications and Innovation

Esercizio: lettura e scrittura file



Aprire il file 'points.txt', ottenere la seconda colonna e moltiplicarne ogni valore per 7,3 .

Scrivere un file di output 'ypoints.csv' che contiene una sola riga con tutti i nuovi valori in formato csv.

(Consigliato l'uso del modulo python apposito.)

Soluzione

```
lista=list()
fh = open("points.txt")
for line in fh:
    lista.append(line.split(' ')[1] * 7.3)
fh.close()

import csv
with open("ypoints.csv","w") as f:
    ch = csv.writer(f)
    ch.writerow(lista)
    f.close()
```

CINECA SCAI
SuperComputing Applications and Innovation

Salvataggio e recupero dei dati

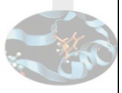
Pickle è il modulo per gestire i dati su file. **cPickle** è la sua versione in C.

Permettono di scrivere e recuperare su file delle strutture dati di Python.

```
>>> dizionario = { 'A' : 1, 'B' : 2 }
#Dump di un dizionario
>>> import cPickle
>>> fh = open("dizionario.data", 'w')
>>> cPickle.dump(dizionario, fh)
>>> fh.close()


#Recupero del dato salvato
>>> fh = open("dizionario.data")
>>> recupero = cPickle.load(fh)
>>> fh.close()
>>> recupero
    {'A': 1, 'B': 2}
```

CINECA SCAI
SuperComputing Applications and Innovation

Accesso file system via OS



os è il modulo che consente attraverso le primitive del sistema operativo l'accesso a files e directory con strutture Python.

```
>>> import os
>>> os.getcwd()
'/home/paulie'
>>> os.chdir('..')
>>> os.getcwd()
'/home'


>>> os.listdir("/var")
['metrics', 'opt', 'run', 'mail', 'lock', 'local', 'spool',
'backups', 'lib', 'log', 'cache', 'tmp', 'crash']
```

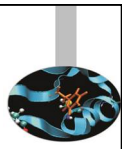
SCAI
SuperComputing Applications and Innovation

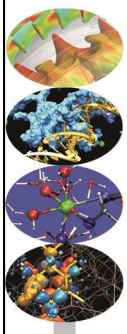
Accesso file system via OS

os è il modulo che consente attraverso le primitive del sistema operativo l'accesso a files e directory con strutture Python.

```
>>> import os

>>> os.remove("/tmp/foo.txt“)
>>> os.remove("/tmp/foo.txt")
    Traceback (most recent call last):
      File "<stdin>", line 1, in <module>
    OSError: [Errno 2] No such file or directory: '/tmp/
    foo.txt'

>>> os.rename("/tmp/foo2.txt", "/tmp/foo.txt")
>>> os.mkdir("/tmp/newdir")
>>> os.listdir("/tmp/newdir")
    []
>>> os.path.split("/tmp/foo2.txt")
    ('/tmp', 'foo2.txt')
```

SCAI
SuperComputing Applications and Innovation





Corso python

Rendere modulare il codice

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

Le funzioni

Le funzioni sono il metodo più diffuso per modularizzare il codice.

- Una funzione prende uno o più parametri come input
- Esegue delle operazioni basate sui parametri ricevuti
- Ritorna il valore calcolato

CINECA SCAI
SuperComputing Applications and Innovation

Le funzioni

Le funzioni sono il metodo più diffuso per modularizzare il codice.

```
def FunctionName(argument1, argument2, ...):
    """ Optional description """
    ... FUNCTION_CODE ...
return DATA
```

CINECA SCAI
SuperComputing Applications and Innovation

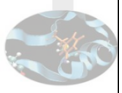
Un esempio semplice di funzione

Vogliamo avere sempre a disposizione nel nostro codice una funzione in grado di **raddoppiare** un numero.

```
>>> def raddoppia(numero):
...     """ Moltiplica il numero in input per due """
...     return numero*2
...
>>> raddoppia(4)
8
>>> a=3
>>> raddoppia(a)
6
>>> b = raddoppia(a*2)
>>> b
12
>>> raddoppia
<function raddoppia at 0xb74cacdc>
>>> help(raddoppia)
raddoppia(numero)
    Moltiplica il numero in input per due
```

CINECA SCAI
SuperComputing Applications and Innovation

Funzioni: uno o più parametri



Una funzione può avere uno o più parametri. Senza parametri non ha molto senso.

Una funzione puo' naturlamente chiamare al suo interno altre funzioni.

```
>>> def ucfirst(stringa):
...     return stringa[0].upper() + stringa[1:]
...

>>> def info(nome, cognome):
...     print "Cognome:\t" + ucfirst(cognome)
...     print "Nome:\t\t" + ucfirst(nome)
...

>>> info("paolo","d'Onorio De Meo")
Cognome:        D'Onorio De Meo
Nome:                   Paolo


>>> def inutile():
...     return "inutile"
...
>>> inutile()
'inutile'
```

CINECA SCAI
SuperComputing Applications and Innovation

Funzioni: valori di default

Un parametro puo' essere opzionale, se ne indichiamo il valore di default durante la definizione della funzione.

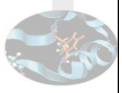
```
>>> def info(nome, cognome, eta = 18):
...     print "Cognome:\t" + ucfirst(cognome)
...     print "Nome:\t\t" + ucfirst(nome)
...     print "Eta:\t\t" + str(eta)
>>> info("Mario",“rossi")
Cognome:        Rossi
Nome:                   Mario
Eta:            18
>>> info(“luca","Bianchi",30)
Cognome:        Bianchi
Nome:                   Luca
Eta:            30
>>> info("Luca")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: info() takes at least 2 arguments (1 given)
```

#Dove lo avevamo gia' incontrato?

```
>>> fh=open("filename.txt")
>>> fh=open("filename.txt",'w')
```

CINECA SCAI
SuperComputing Applications and Innovation

Funzioni: valori di ritorno



Possiamo ritornare anche un dato complesso.

```
>>> def ucfirst(stringa):
...     return stringa[0].upper() + stringa[1:].lower()
...

>>> def info(nome, cognome, eta = 18):
...     return [ucfirst(nome), ucfirst(cognome), eta]
...

>>> info("Mario","Rossi")
['Mario', 'Rossi', 18]
>>> info("Luca","Bianchi",30)
['Luca', 'Bianchi', 30]

>>> def info(nome, cognome, eta = 18):
...     return {'Nome':ucfirst(nome), 'Cognome':ucfirst(cognome),
"Eta'":eta }
...
>>> info("Mario","Rossi")
{'Cognome': 'Rossi', "Eta'": 18, 'Nome': 'Mario'}
```

CINECA SCAI
SuperComputing Applications and Innovation

Funzioni: lo *scope* di una variabile

Attenzione al valore di una variabile in base al contesto in cui viene usata

```
>>> def test(numero):
...     x=3
...     print("Valore di x = " + str(x))
...     return numero*2
...

>>> x=55
>>> test(4)
Valore di x = 3
8
>>> x
55
>>> numero
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'numero' is not defined
```

CINECA SCAI
SuperComputing Applications and Innovation

Funzioni: numero di valori variabile

Il numero di parametri passato a una funzione può essere definito variabile!

```
def average(*numbers):
    if len(numbers)==0:
            return None
    else:
            total = sum(numbers)
    return float(total)/len(numbers)

>>> average(2,4)
3
>>> average(6,4,8)
9
```

CINECA SCAI
SuperComputing Applications and Innovation

Esercizio sulle funzioni

1. Scrivere e *testare* una funzione che prenda una lista di parole e ritorni la dimensione della più lunga

2. Scrivere e *testare* una funzione che calcoli il numero di caratteri contenuti in un file

Soluzione 1

```
def stringapiugrande(lista):
    maxlen = 0
    for i in lista:
        if len(i) > maxlen:
            maxlen = len(i)
    return maxlen

lista = [ "test", "prova", "a" ]
print stringapiugrande(lista)
#Stampa: 5
```

Soluzione 2

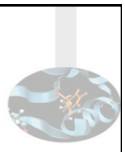
```
def lunghezza_file(nome_file):
    counter = 0
    for line in open(nome_file):
        counter += len(line)
    return counter

f="points.txt"
print "Il file "+f+" risulta lungo " + str(lunghezza_file(f)) + " caratteri"
#Il file points.txt risulta lungo 96 caratteri
```

CINECA SCAI
SuperComputing Applications and Innovation

MODULI: il concetto

- Cos'è un modulo?
  - Un file contentente definizioni di funzioni e tipi di strutture Python
- Namespace
  - Uno spazio indipendente di nomi
  - math.log è diverso da eventuale funzione log definita da noi
- Il nome del modulo
  - il file "mio_modulo.py" contiene il modulo "mio_modulo"

CINECA SCAI
SuperComputing Applications and Innovation



MODULI: import

```
>>> import math
>>> pi
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pi' is not defined
>>> math.pi
3.141592653589793
```

#Don't use "from module import *" unless YOU KNOWN WHAT YOU ARE DOING

```
>>> from math import *
>>> pi
3.141592653589793
>>> dir
<built-in function dir>
>>> dir()
['__builtins__', '__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', 'math', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

CINECA SCAI
SuperComputing Applications and Innovation

MODULI: dove Python va a cercare

1. Nella stessa directory in cui ci troviamo al momento dell'esecuzione dell'interprete
2. Nella stessa directory dove si trovano gli eseguibili del Python
3. In una directory specificata da noi (variabile PYTHONPATH)

```
>>> import sys
>>> sys.path
['', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-i386-linux-gnu', '/usr/lib/
python2.7/lib-tk', … ]
```

CINECA SCAI
SuperComputing Applications and Innovation

MODULI: installazione

- PYTHONPATH
  - Variabile di ambiente del sistema operativo
- Pacchettizzazione di sistema
  - Ubuntu: apt
  - RedHat: yum
  - Suse: yast
- Easy install
  - $ easy_install
    Il programma "easy_install" non è attualmente installato. È possibile installarlo digitando:
    sudo apt-get install python-setuptools
  - sudo easy_install NOMEMODULO
- Standard
  - $ python setup.py install

CINECA SCAI
SuperComputing Applications and Innovation

MODULI: creare il proprio

HOME/test.py

```
#un valore di default
zero=0
#lasciare il valore intatto
def lascia(numero=zero):
    return numero
#valore per 2
def raddoppia(numero=zero):
    return numero*2
```

bash:HOME$ python

```
>>> import test
>>> dir(test)
['__builtins__', '__doc__', '__file__', '__name__',
'__package__', 'lascia', 'raddoppia', 'zero']
>>> print __doc__
None
>>> test.lascia()
0
>>> test.lascia(2)
2
>>> test.raddoppia(2)
4
>>> test.raddoppia()
0
>>> test.zero
0
>>> test.raddoppia(test.zero)
0
```

CINECA SCAI
SuperComputing Applications and Innovation

Esercizio

Crea il tuo primo modulo:

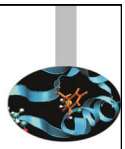
L'agenda

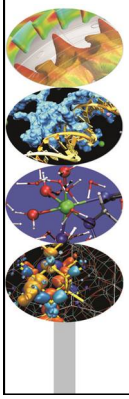
Gestire un'agenda telefonica attraverso almeno due funzioni:

a) Inserimento dati (con richiesta interattiva di Nome e Numero)

b) Ricerca numero (a partire dal Nome)

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

Gestione degli errori

CINECA

SCAI
SuperComputing Applications and Innovation

Cos'è l'errore

```
>>> fh = open("nonesisto.csv")
    Traceback (most recent call last):
      File "<stdin>", line 1, in <module>
    IOError: [Errno 2] No such file or directory: 'nonesisto.csv'
```

NAESER'S LAW:

You can make it foolproof, but you can't make it damn-foolproof.

SCAI
SuperComputing Applications and Innovation

Evitare errori non previsti

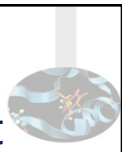
Possiamo evitare di cadere in alcuni errori con dei controlli.

E.g.

```
>>> if os.access("nonesisto.csv",os.W_OK):
...     fh = open("nonesisto.csv")
...
```

CINECA SCAI
SuperComputing Applications and Innovation

Gestione errori: blocco try/except



```
try:
    code block 1
    # ...some error prone code...
except:
    code block 2
    # ...do something with the error...
[else:
    code block 3
    # ...to do when there is no error...
finally:
    code block 4
     #...some clean up code..].
```

CINECA SCAI
SuperComputing Applications and Innovation

Houston abbiamo un problema

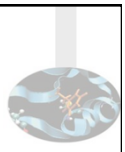
```
try:
   print 0/0
except:
   print("Houston, we have a problem...")
```

Il risultato è

Houston, we have a problem...

CINECA SCAI
SuperComputing Applications and Innovation

Caso reale di gestione



```
#Gestione di un vocabolario
d = {"A":"Adenine","C":"Cisteine","T":"Timine","G":"Guanine"}
try:
    print d[raw_input("Enter letter: ")]
except:
    print "No such nucleotide”
#Gestione del tipo di errore!
d={"A":"Adenine", "C":"Cisteine", "T":"Timine", "G":"Guanine"}
try:
    print(d[raw_input("Enter letter: ")])
except EOFError:
    print("Good bye!")
except KeyError:
    print("No such nucleotide")
```

CINECA SCAI
SuperComputing Applications and Innovation

Tipi di eccezioni

```
import sys

try:
   f = open('myfile.txt')
   s = f.readline()
   i = int(s.strip())
except IOError, (errno, strerror):
   print "Errore I/O (%s): %s" % (errno, strerror)
except ValueError:
   print "Non si può convertire il dato in un intero."
except:
   print "Errore inatteso:", sys.exc_info()[0]
   raise
```

CINECA SCAI
SuperComputing Applications and Innovation

Tipi di eccezioni

- exception **IOError**
- *exception* **ArithmeticError**
- exception **MemoryError**
- exception **NameError**
- exception **OSError**
- exception **SyntaxError**
  - **IndentationError**
- exception **SystemExit**
- Un utente può anche definire una proprio tipo di eccezione
  - Richiede conoscenza di OOP

CINECA SCAI
SuperComputing Applications and Innovation

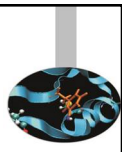
Provocare una *eccezione*

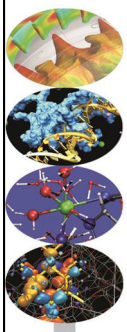
L'istruzione raise permette al programmatore di forzare una specifica eccezione.

Per esempio:

```
>>> raise NameError, 'HiThere'

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: HiThere
```

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

Come continuare a lavorare sulle basi?

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

DOCS

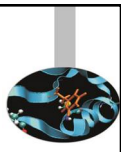
Help inline

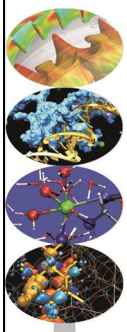
- introspezione
- documentazione comando help()

Help online

- ufficiale del linguaggio
  - http://www.python.org/doc/current
- Tradotti in italiano
  - http://docs.python.it

CINECA SCAI
SuperComputing Applications and Innovation





Fine della prima parte

Bravi!

CINECA