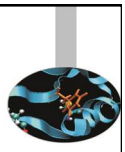
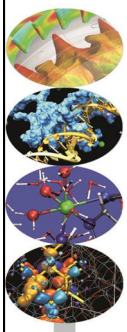
CINECA SCAI
SuperComputing Applications and Innovation



Corso PYTHON: Benvenuti



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***

2

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

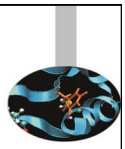
CINECA SCAI
SuperComputing Applications and Innovation

Seconda parte: Sommario

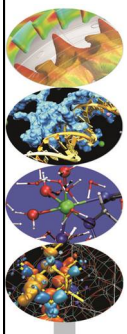
1. **Introduzione (con *overview* di IPython)**
2. **Introduzione a *NumPy***
   1. **L'oggetto *NDarray***
   2. **Operazioni su array**
3. ***Matplotlib*: il modulo *pylab***
4. **Introduzione a *SciPy***
5. **Performance in Python: *mixed language programming***
   1. **Introduzione**
   2. ***F2PY***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

5

CINECA SCAI
SuperComputing Applications and Innovation



Python: introduzione al linguaggio



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

CINECA

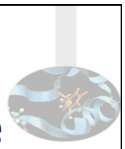
CINECA SCAI
SuperComputing Applications and Innovation

Python

Un linguaggio di programmazione
moderno
che sta imponendo
il suo potenziale
all'attenzione
degli sviluppatori
e della comunità scientifica

7

CINECA SCAI
SuperComputing Applications and Innovation

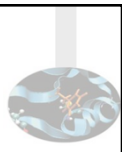


PYTHON: principali caratteristiche

- 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

8

CINECA SCAI
SuperComputing Applications and Innovation



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.

9

CINECA SCAI
SuperComputing Applications and Innovation

Comparazione: leggibilità

PYTHON

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

JAVA

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

10

CINECA SCAI
SuperComputing Applications and Innovation

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"))
```

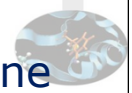
C

```
#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);
}
```

11

CINECA SCAI
SuperComputing Applications and Innovation

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

12

CINECA SCAI
SuperComputing Applications and Innovation

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

13

CINECA SCAI
SuperComputing Applications and Innovation

PYTHON: flavors

Python is actually a language definition

Implementazione standard: **cPython**

14

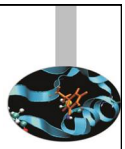
CINECA SCAI
SuperComputing Applications and Innovation

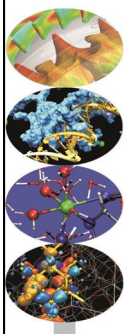
PYTHON: flavors

- 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

15

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

IL PROBLEMA DELLE VERSIONI

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

Python e le versioni recenti

- Python 2.7.5
  - was released on May 15, 2013
- Python 3.0 final
  - was released on December 3rd, 2008.
- Python 3.3.2
  - was released on May 15th, 2013

17

CINECA SCAI
SuperComputing Applications and Innovation

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

18

CINECA SCAI
SuperComputing Applications and Innovation

Python 2 e Python 3

- *Guido van Rossum* ha deciso di ripulire Python 2.x in maniera appropriata...
  - ….senza preoccuparsi INTENZIONALMENTE della retrocompatibilità!
- 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!

19

CINECA SCAI
SuperComputing Applications and Innovation

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

20

Python 2	Python 3
"pragmatici"	*"idealisti"*

CINECA SCAI
SuperComputing Applications and Innovation

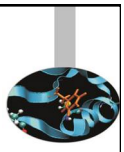
Python 2 o Python 3?

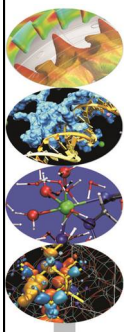
Python 2 sarà il riferimento di questo corso

- Standard attuale de facto
- Riferimento per la comunità scientifica
- Compatibilità al 100% con tutte le librerie matematiche e di plotting

22

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

INSTALLAZIONE

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

Installazione

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

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

CINECA SCAI
SuperComputing Applications and Innovation

Installazione

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

```
paulie@revolution:~$ which python
   /usr/bin/python
paulie@revolution:~$ python -V
   Python 2.7.4
```

25

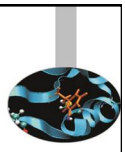
CINECA SCAI
SuperComputing Applications and Innovation

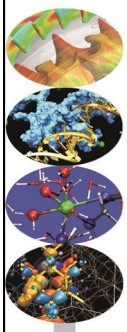
RH5: script di installazione da bash

```
#DOWNLOAD
echo "Downloading installer";
url="https://www.enthought.com/downloads/canopy/rh5-64/free/"
script="${proj}_install.sh"
wget --quiet "$url" -O $script >/dev/null 2>&1
#EXECUTE in batch mode (no interactive requests)
echo "Launching installer";
bash $script -b >/dev/null 2>&1 << EOL
yes
EOL
rm -f $script
#CONFIGURE
echo "Launching display configuration";
installer="$HOME/$uc/$proj"
$installer
```

26

CINECA SCAI
SuperComputing Applications and Innovation





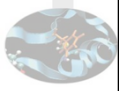
Corso python

PRIMI PASSI: **INTERPRETE**

Il miglior modo per imparare il python è usarlo!

CINECA

CINECA SCAI
SuperComputing Applications and Innovation



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
>>>
```

28

CINECA SCAI
SuperComputing Applications and Innovation

Ciao Mondo!

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

29

CINECA SCAI
SuperComputing Applications and Innovation

Ciao Mondo!

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

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

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

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

30

CINECA SCAI
SuperComputing Applications and Innovation

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
```

31

CINECA SCAI
SuperComputing Applications and Innovation

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!
```

32

SCAI
SuperComputing Applications and Innovation

Richiedere un input

Funzione **raw_input**

Prende una stringa in interattivo

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

>>> name
'Seba'
```

33

SCAI
SuperComputing Applications and Innovation

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
```

34

CINECA SCAI
SuperComputing Applications and Innovation

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'
```

35

CINECA SCAI
SuperComputing Applications and Innovation

Richiedere un input

Funzione **input**

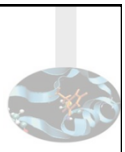
Richiede una stringa, ma cerca di valutarla come se fosse un oggetto Python

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

36

CINECA SCAI
SuperComputing Applications and Innovation

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
```

37

CINECA SCAI
SuperComputing Applications and Innovation

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
```

38

SCAI
SuperComputing Applications and Innovation

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'
```

39

SCAI
SuperComputing Applications and Innovation

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'
```

40

CINECA SCAI
SuperComputing Applications and Innovation

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
```

41

CINECA SCAI
SuperComputing Applications and Innovation

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
```

42

CINECA SCAI
SuperComputing Applications and Innovation

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.)

43

CINECA SCAI
SuperComputing Applications and Innovation

Soluzione

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

```
>>> numero = raw_input("Inserisci un numero: ")
Inserisci un numero: 3
>>> numero
'3'
#Attenzione: una stringa
>>> numero * 4
'3333'
#Attenzione alla precedenza
>>> 4 * int(numero) + 1
13
#Corretto
>>> 4 * ( int(numero) + 1 )
16
```

44

SCAI
SuperComputing Applications and Innovation

Exit

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

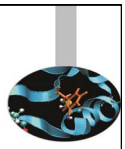
Non è necessaria dentro i file .py

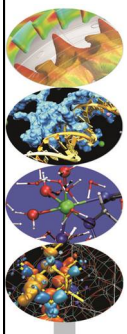
Exit termina la nostra "sessione"

```
paulie@revolution:~$ python
Python 2.7.4
[GCC 4.7.3] on linux2

>>> exit()

paulie@revolution:~$
```

45

SCAI
SuperComputing Applications and Innovation





Corso python

PRIMI PASSI: **BATCH MODE**

Il miglior modo per imparare il python è usarlo!

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

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**.

47

CINECA SCAI
SuperComputing Applications and Innovation

Il nostro primo file Python

Contenuto del file

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

Interprete

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

48

CINECA SCAI
SuperComputing Applications and Innovation

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!’)’
```

49

CINECA SCAI
SuperComputing Applications and Innovation

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!
```

50

SCAI
SuperComputing Applications and Innovation

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!
```

51

SCAI
SuperComputing Applications and Innovation

I commenti

Contenuto del file

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

Interprete

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

52

CINECA SCAI
SuperComputing Applications and Innovation

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");}
```

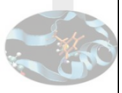
Con indentazione

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

53

CINECA SCAI
SuperComputing Applications and Innovation

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")
```

54

CINECA SCAI
SuperComputing Applications and Innovation

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

55

CINECA SCAI
SuperComputing Applications and Innovation

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)

56

Soluzione

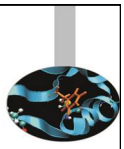
File

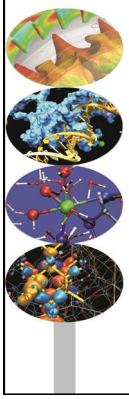
```
numero = raw_input("Inserisci un numero: ")
risultato = int(numero) / 5.0 * 4
print "Il risultato e' " + str(risultato)
```

Interprete

```
paulie@revolution:~$ python file.py
Inserisci un numero: 7
Il risultato e' 5.6
```

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

Primi accenni sui moduli

CINECA

SCAI
SuperComputing Applications and Innovation

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/

59

SCAI
SuperComputing Applications and Innovation

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
```

60

CINECA SCAI
SuperComputing Applications and Innovation

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(...)
```

61

CINECA SCAI
SuperComputing Applications and Innovation

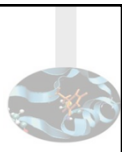
Moduli: help e documentazione

```
>>> 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
```

62

CINECA SCAI
SuperComputing Applications and Innovation

Moduli: help e documentazione

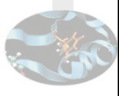


```
>>> help()

    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
>>>
```

63

CINECA SCAI
SuperComputing Applications and Innovation

Moduli e introspezione

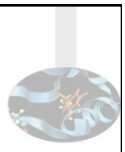


```
# it isn't as easy to remember
# what each module contains.
>>> import math
>>> dir(math)
    ['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh',
    'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e',
    'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial',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

64

CINECA SCAI
SuperComputing Applications and Innovation

Moduli e introspezione

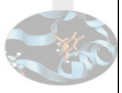


```
# What did i load so far?

>>> import math

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

65

CINECA SCAI
SuperComputing Applications and Innovation

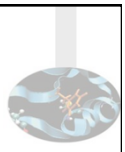
Moduli e introspezione



```
# 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
```

66

CINECA SCAI
SuperComputing Applications and Innovation



Esercizio su introspezione

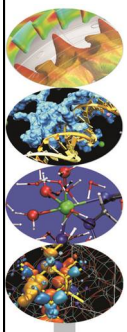
Applicare il seno del logaritmo di 4

CINECA SCAI
SuperComputing Applications and Innovation

Soluzione

Applicare il seno del logaritmo di 4

```
>>> dir(math)

>>> math.sin(math.log(4))
    0.9830277404112437
```



Corso python

I TIPI DI DATO: **stringhe**

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'>
```

70

SCAI
SuperComputing Applications and Innovation

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
```

71

SCAI
SuperComputing Applications and Innovation

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'
```

72

CINECA SCAI
SuperComputing Applications and Innovation

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'
```

73

CINECA SCAI
SuperComputing Applications and Innovation

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'
```

74

CINECA SCAI
SuperComputing Applications and Innovation

Manipolazione di stringhe

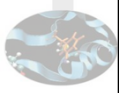
Cercate di capire cosa sta accadendo a questa stringa

```
>>> 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'
```

75

CINECA SCAI
SuperComputing Applications and Innovation

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

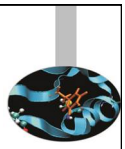
76

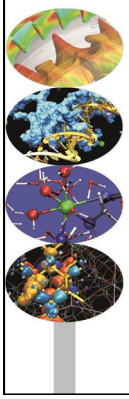
Soluzione

- 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

```
>>> stringa="Contengo tre parole"
>>> pezzi = stringa.split(‘ ‘)
>>> pezzi[0][-2:] + pezzi[1][-2:] + pezzi[2][-2:]
    'gorele'
```

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

I TIPI DI DATO: **liste**

“List Is the Workhorse Datatype in Python”

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

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
    []
```

79

CINECA SCAI
SuperComputing Applications and Innovation

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]
```

80

CINECA SCAI
SuperComputing Applications and Innovation

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
```

81

CINECA SCAI
SuperComputing Applications and Innovation

Modificare una lista

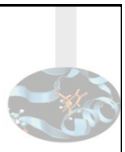
Inserire e rimuovere oggetti nella lista

```
>>> 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]
```

82

CINECA SCAI
SuperComputing Applications and Innovation

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]
```

83

CINECA SCAI
SuperComputing Applications and Innovation

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

84

Soluzione

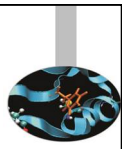
1. Accedere all'elemento di valore '5'

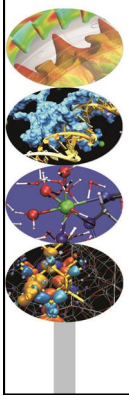
```
>>> lista = [ 1, [7, 6], [8, [9, 4, 5] ] ,6, [ 0, 2] ]
>>> lista[2][1][2]
    5
>>> lista[2][1][-1]
    5
```

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

```
>>> [10*100] * 10
   [1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000]
>>> lista = [10*100] * 10
>>> lista.insert(5,5)
>>> lista
   [1000, 1000, 1000, 1000, 1000, 5, 1000, 1000, 1000, 1000, 1000]
```

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

I TIPI DI DATO: **tuple**

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

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'>

>>>
```

87

CINECA SCAI
SuperComputing Applications and Innovation

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?
```

88

SCAI
SuperComputing Applications and Innovation

A cosa serve allora una tupla?

Esempio tipico:

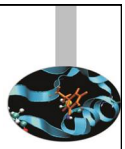
un sistema di coordinate.

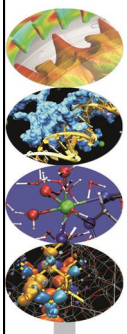
Anche nei valori di ritorno di funzioni risultano molto comode.

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

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

89

SCAI
SuperComputing Applications and Innovation





Corso python

I TIPI DI DATO: **sequenze**

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

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'
```

91

CINECA SCAI
SuperComputing Applications and Innovation

Sequenze e slicing

Pezzetti della sequenza "Python"

P	y	t	h	o	n
0	1	2	3	4	5	6

```
>>> 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'
```

92

CINECA SCAI
SuperComputing Applications and Innovation

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
```

93

CINECA SCAI
SuperComputing Applications and Innovation

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
```

94

CINECA SCAI
SuperComputing Applications and Innovation

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
```

95

CINECA SCAI
SuperComputing Applications and Innovation

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

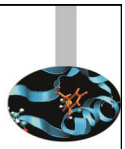
96

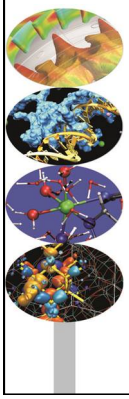
Soluzione

- 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

```
>>> stringa = "prova“
>>> punto = (1, 3, -1)
>>> risultato = min(stringa) + stringa[::-1] + str(max(punto))
>>> risultato
   'aavorp3'
```

CINECA SCAI
SuperComputing Applications and Innovation





Corso python

I TIPI DI DATO: **dizionari**

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

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'>
```

99

CINECA SCAI
SuperComputing Applications and Innovation

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!

100

CINECA SCAI
SuperComputing Applications and Innovation

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
```

101

CINECA SCAI
SuperComputing Applications and Innovation

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')
```

102

CINECA SCAI
SuperComputing Applications and Innovation

Gestione di un elemento

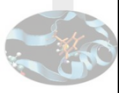
La funzione **get** permette di ottenere un valore a partire da una chiave.

```
#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'
```

103

CINECA SCAI
SuperComputing Applications and Innovation

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'}
```

104

CINECA SCAI
SuperComputing Applications and Innovation

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

105

CINECA SCAI
SuperComputing Applications and Innovation

Dizionari in python 3: *dict views*

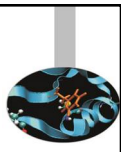
Non abbiamo più la lista come output di default nella gestione dei dizionari.

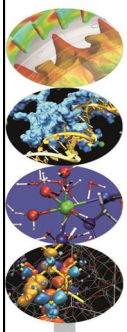
Le dict_views sono *sincronizzate* con l'oggetto originale

```
>>> 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']
```

106

CINECA SCAI
SuperComputing Applications and Innovation



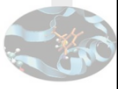


Corso python

I TIPI DI DATO: **gli insiemi**

CINECA

CINECA SCAI
SuperComputing Applications and Innovation



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'>
```

108

SCAI
SuperComputing Applications and Innovation

Gli insiemi

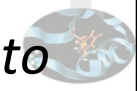
Operazioni base

```
>>> 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'
```

109

SCAI
SuperComputing Applications and Innovation

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
```

110

CINECA SCAI
SuperComputing Applications and Innovation

Insiemi: *le operazioni fondamentali*

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

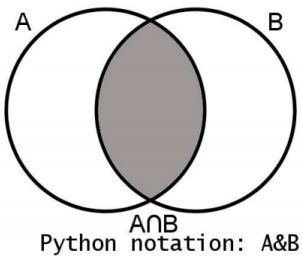
Questo tipo di operazioni non sono facili su strutture dati più comuni.

```
>>> 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'])
```

111

CINECA SCAI
SuperComputing Applications and Innovation

Insiemi: *intersezione*



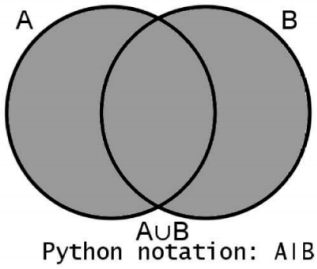
Python notation: A&B

```
>>> 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'])
```

112

CINECA SCAI
SuperComputing Applications and Innovation

Insiemi: *unione*

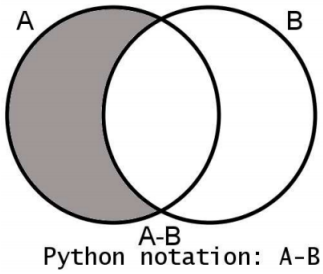



```
>>> a = set('abracadabra')
>>> a
    set(['a', 'r', 'b', 'c', 'd'])
>>> b = set('alacazam')
>>> b
    set(['a', 'c', 'z', 'm', 'l'])

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

113

CINECA SCAI
SuperComputing Applications and Innovation

Insiemi: *differenza*

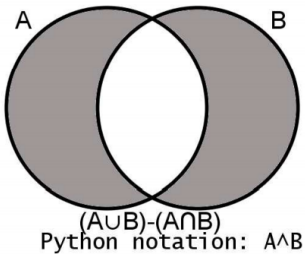



```
>>> a = set('abracadabra')
>>> a
    set(['a', 'r', 'b', 'c', 'd'])
>>> b = set('alacazam')
>>> b
    set(['a', 'c', 'z', 'm', 'l'])

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

114

SCAI
SuperComputing Applications and Innovation

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'])
```

115

SCAI
SuperComputing Applications and Innovation

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'}
```

116

CINECA SCAI
SuperComputing Applications and Innovation

Esercizio sugli insiemi

Trovare le lettere che si trovano sia nella stringa “fruttato” che in “affondare”

e che contemporaneamente

non siano presenti nell’insieme delle lettere diverse tra le stringhe “maturando” e “famelico”.

117

Soluzione

Trovare le lettere che si trovano sia nella stringa “fruttato” che in “affondare”

e che contemporaneamente

non siano presenti nell’insieme delle lettere diverse tra le stringhe “maturando” e “famelico”.

```
>>> p1 = set(list("fruttato"))
>>> p2 = set(list("affondare"))
>>> p3 = set(list("maturando"))
>>> p4 = set(list("famelico"))
>>> (p1 & p2) - (p3 ^ p4)
    set(['a', 'o'])
```

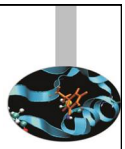
SCAI
SuperComputing Applications and Innovation

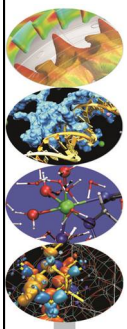
Python 2: sommario tipi di dato

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])`

119

SCAI
SuperComputing Applications and Innovation





Corso python

Assegnazione variabili
VS
Binding di un nome ad un oggetto

CINECA

SCAI
SuperComputing Applications and Innovation

Passaggio per valore

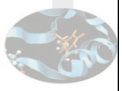
Salvataggio
del valore di **a**
nella lista **b**

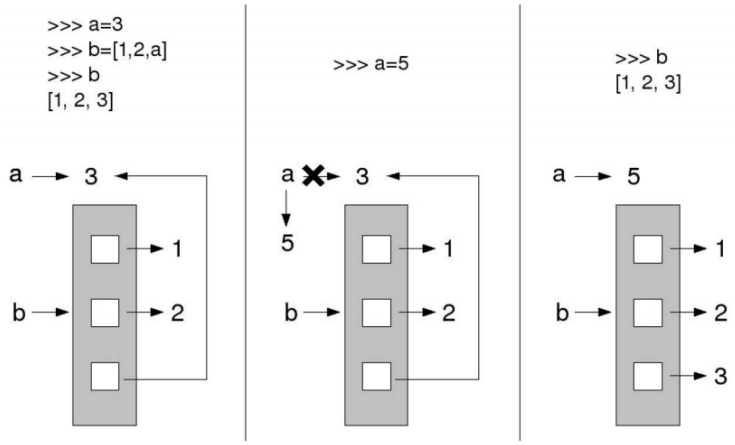
```
>>> a = 3
>>> b = [ 1, 2, a ]
>>> b
[1, 2, 3]

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

121

SCAI
SuperComputing Applications and Innovation

Passaggio per valore





122

SCAI
SuperComputing Applications and Innovation

Binding di un nome a un oggetto

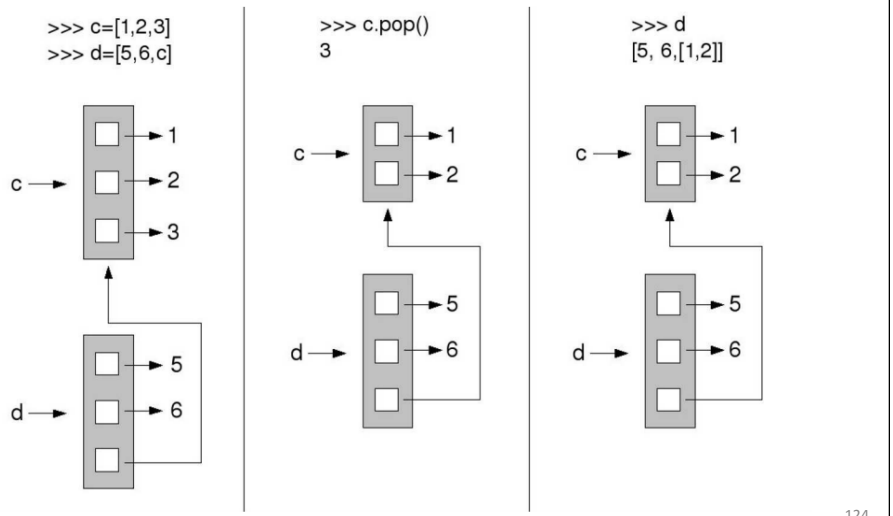
Salvataggio della lista **c** nella lista **d**

```
>>> 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]]
```

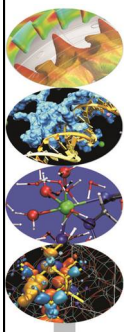
123

SCAI
SuperComputing Applications and Innovation

Binding di un nome a un oggetto



124



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
```

126

SCAI
SuperComputing Applications and Innovation

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'
```

127

SCAI
SuperComputing Applications and Innovation

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'
```

128

CINECA SCAI
SuperComputing Applications and Innovation

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'
```

129

CINECA SCAI
SuperComputing Applications and Innovation

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'
```

130

SCAI
SuperComputing Applications and Innovation

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
```

131

SCAI
SuperComputing Applications and Innovation

Controllo del flusso 2: *for*

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

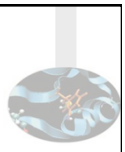
for VAR **in** ITERABLE:
    BLOCK

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

132

SCAI SuperComputing Applications and Innovation

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
```

133

SCAI SuperComputing Applications and Innovation

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
```

134

CINECA SCAI
SuperComputing Applications and Innovation

Controllo del flusso 2: *for*

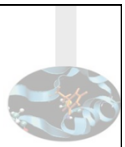
Una stringa è comunque una lista di caratteri. Perciò:

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

135

CINECA SCAI
SuperComputing Applications and Innovation

Esercizio sui flussi



Stampare gli interi compresi tra 10 e 100
che sono divisibili contemporaneamente per 7 e per 2.

136

Soluzione

Stampare gli interi compresi tra 10 e 100

che sono divisibili contemporaneamente per 7 e per 2.

```
>>> for i in range(10,100):
...     if (i % 7 == 0) and (i % 2 == 0) :
...         print(str(i) +" divisibile per 7 e 2“)
...
    14 divisibile per 7 e 2
    28 divisibile per 7 e 2
    42 divisibile per 7 e 2
    56 divisibile per 7 e 2
    70 divisibile per 7 e 2
    84 divisibile per 7 e 2
    98 divisibile per 7 e 2
```

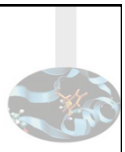
Soluzione più carina

Stampare gli interi compresi tra 10 e 100

che sono divisibili contemporaneamente per 7 e per 2.

```
>>> for i in range(10,100,2):
...     if i % 7 == 0:
...         print(str(i) +" divisibile per 7 e 2“)
...
    14 divisibile per 7 e 2
    28 divisibile per 7 e 2
    42 divisibile per 7 e 2
    56 divisibile per 7 e 2
    70 divisibile per 7 e 2
    84 divisibile per 7 e 2
    98 divisibile per 7 e 2
```

CINECA SCAI
SuperComputing Applications and Innovation



Controllo del flusso 3: *while*

Ripetizione di un blocco finchè una condizione è vera.

while EXPRESSION**:**
    BLOCK

L'espressione indica la condizione da verificare.

CINECA SCAI
SuperComputing Applications and Innovation

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
```

CINECA SCAI
SuperComputing Applications and Innovation

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
```

141

CINECA SCAI
SuperComputing Applications and Innovation

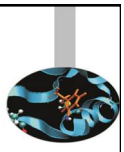
Controllo del flusso

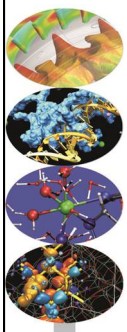
Riassunto:

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**

142

CINECA SCAI
SuperComputing Applications and Innovation





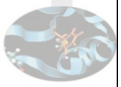
Primo giorno: Esercitazione finale

La massima del giorno:

più introspezione

CINECA

CINECA SCAI
SuperComputing Applications and Innovation



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**

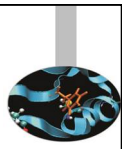
CINECA SCAI
SuperComputing Applications and Innovation

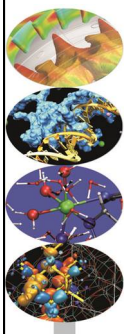
Soluzione

```
>>> import random
>>> random.random()
0.5313499052937661
>>> random.randrange(1,500)
114

>>> while True:
        r = random.randrange(1,500)
        if r % 13 == 0 and str(r)[1] == "5":
            print "Trovato " + str(r)
            break
```

145

CINECA SCAI
SuperComputing Applications and Innovation





Fine del primo giorno!

Bravi

CINECA