

Programmazione in Python

Dario Pescini - Mirko Cesarini

Università degli Studi di Milano-Bicocca

Dipartimento di Statistica e Metodi Quantitativi

`nome.cognome@unimib.it`

- far eseguire all'elaboratore compiti tediosi
- velocizzare operazioni ripetitive
- ridurre errore umano
-

- Obiettivo: imparare la programmazione
- Perché?
 - Come statistici avrete a che fare con grosse moli di dati
 - Alternative: fare i conti a mano / far lavorare i computer
- Che cosa richiede?
 - Acquisire le competenze necessarie per:
 - progettare e implementare (piccole) procedure di elaborazione automatica di dati e informazioni
 - essere in grado di leggere, capire ed eventualmente modificare codice scritto da altre persone
 - Imparare ad analizzare i problemi con un approccio analitico
 - Distinguere la patologia dai sintomi
 - Distinguere i problemi principali dai problemi secondari

- La programmazione richiede di imparare a pensare in modo diverso da quello a cui siete abituati
- ciò non vale solo per l'informatica (ma anche per tante altre discipline)
- Relazione tra teoria e pratica
 - conosco un argomento se sono in grado di metterlo in pratica
 - L'applicazione a casi reali aiuta a capire meglio gli aspetti teorici
 - Miglioramento continuo: mettete in discussione le vostre idee, verificate se stanno in piedi, miglioratele nel tempo
- Le esercitazioni in laboratorio sono parte integrante delle lezioni
 - Per imparare la programmazione dovete esercitarvi (tanto)
 - Le parti teoriche non sono opzionali: ripassate gli argomenti delle lezioni prima dei laboratori (altrimenti buttate via il tempo)

Programmieren....

Continua manipolazione di **oggetti** tramite **azioni** predefinite.

Azioni

Istruzioni

Espressioni

Funzioni

Variabili



nome
contenuto

Aziendi



ordine delle istruzioni

strutture di controllo

Algoritmo

procedimento che a partire da uno stato iniziale, consente di ottenere in un tempo finito un risultato atteso eseguendo un insieme di operazioni descritte in maniera completa e non ambigua

Può essere eseguito da un elaboratore poiché

- è una **descrizione completa** e **non ambigua** di un procedimento
- produce un risultato in un **tempo finito**

Algoritmo

Algoritmo: sequenza di passi che portano alla soluzione del problema.

Elementi di cui serve una chiara definizione:

- problema da risolvere
- input
- output
- sequenza di passi (logica)

Elementi

- **problema** Cuocere un uovo al tegamino
- **input** uovo, sale
- **output** uovo cotto
- **logica**
 - prendi uovo
 - mettilo in padella
 - cuocilo
 - salalo
 - mettilo nel piatto

É sufficiente? riuscireste ad ottenere l'uovo cotto come vorreste?

Algoritmo - Uovo al tegamino

logica

- 1 prendi padella
- 2 metti la padella sul fuoco
- 3 accendi il fuoco sotto la padella
- 4 attendi temperatura corretta
- 5 rompi uovo
- 6 metti uovo in padella
- 7 aggiungi sale
- 8 attendi 5'
- 9 prendi piatto
- 10 toglì la padella dal fuoco
- 11 versa il contenuto della padella nel piatto

Algoritmo - Uovo al tegamino

logica

- 1 prendi padella
- 2 metti la padella sul fuoco
- 3 accendi il fuoco sotto la padella
- 4 attendi temperatura corretta
- 5 rompi uovo *e se l'uovo è scaduto?*
- 6 metti uovo in padella *e se l'uovo si rompe?*
- 7 aggiungi sale
- 8 attendi 5' *se è crudo o bruciato?*
- 9 prendi piatto
- 10 toglì la padella dal fuoco
- 11 versa il contenuto della padella nel piatto

Algoritmo

Algoritmo: sequenza di passi che portano alla soluzione del problema.

Correttezza

- capacità di risolvere il problema senza difettare di alcun passaggio fondamentale
- capacità di risolvere il problema per ogni caso possibile

Efficienza

capacità di risolvere il problema utilizzando il minimo necessario di risorse (spazio, tempo)

- Immaginate di dover ordinare in ordine crescente i numeri seguenti:

10	5	8	1	4	3
----	---	---	---	---	---

Ruolo degli algoritmi

- Immaginate di dover ordinare in ordine crescente i numeri seguenti:

10	5	8	1	4	3
----	---	---	---	---	---

1	3	4	5	8	10
---	---	---	---	---	----

Esperimento

- Immaginate di avere un'aula didattica, con 100 studenti disposti a forma di scacchiera rettangolare.
- Immaginate che ogni studente riceva (casualmente) un numero intero
- Supponete che i numeri debbano essere ordinati (in ordine crescente), da sinistra a destra e dal basso all'alto, come nell'esempio seguente

91	92	93	...	99	100
...
11	12	13	...	19	20
1	2	3	...	9	10

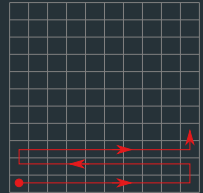


- Proviamo ad eseguire l'esperimento in aula (il docente vi distribuirà dei fogli con dei numeri sopra)

Provate a sfruttare il seguente algoritmo:

Bubble Sort Umano Distribuito

- 1 Individuate una linea che si snoda su ogni riga della scacchiera (vedi a lato)
- 2 Confrontate il vostro numero con quello del collega che vi segue (secondo lo schema)
- 3 Se i vostri 2 numeri non sono nell'ordine giusto, scambiatevi i fogli
- 4 Se uno dei fogli del vostro vicino cambia, ripetete le operazioni dal punto 2
- 5 Ripetete le operazioni dal punto 2 fino a che non ci sarà più alcuno scambio

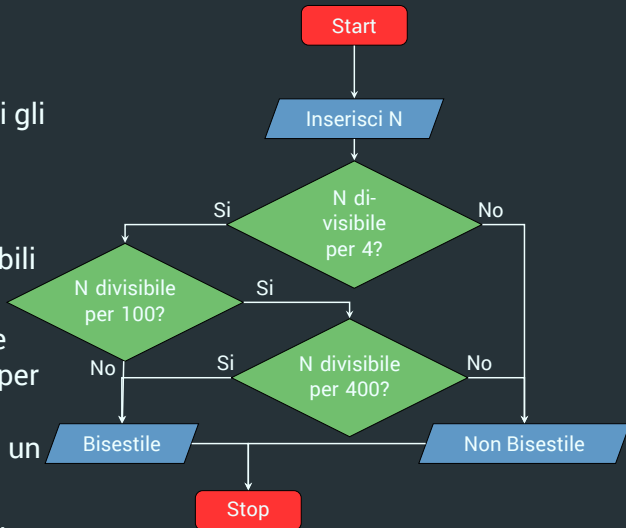


Algoritmi, considerazioni

- Vedete, quello che appariva semplice (con piccoli numeri e con una sola persona coinvolta) ...
- ...si è rivelato molto più complicato in uno scenario più complesso
- La soluzione è stata l'applicazione di un algoritmo
- Nella slide precedente abbiamo descritto in maniera **completa** e **non ambigua** un procedimento, che ha permesso a più persone di cooperare.

Esempio di algoritmo: test di anno bisestile

- La regola (l'algoritmo) dice che sono bisestili gli anni divisibili per quattro, tranne quelli di inizio secolo non divisibili per 400.
- Il 1944 è bisestile perché divisibile per quattro, ma 1500 non lo è perché è un inizio secolo non divisibile per 400.



Algoritmi e linguaggi di programmazione

Come si arriva a far eseguire un algoritmo ad un calcolatore?

Algoritmi e linguaggi di programmazione

Come si arriva a far eseguire un algoritmo ad un calcolatore?

Scomponiamo il problema in 2 sotto-problemi:

- 1 rappresentare/ *descrivere* un algoritmo
 - Un *linguaggio di programmazione* è un linguaggio artificiale che usato per descrivere algoritmi
 - Un *programma* è una sequenza di istruzioni (scritte in uno specifico linguaggio di programmazione) che codifica un algoritmo
- 2 Come far *eseguire* la “descrizione di un algoritmo” (un *programma*) ad un calcolatore

Linguaggi di programmazione

Sono linguaggi che **permettono la codifica di algoritmi** per renderli eseguibili dal calcolatore.

Tassonomia per “somiglianza” con il linguaggio naturale:

- **linguaggi di alto livello:**
 - i più simili al linguaggio naturale
 - istruzioni astratte
 - indipendenti dai dettagli dell'architettura della macchina
 - necessitano di una “traduzione” per essere eseguiti
- **linguaggi assembler:**
 - istruzioni codificate in sequenze di caratteri alfa-numerici
 - dipendenti dai dettagli dell'architettura della macchina
 - necessitano di una “traduzione” per essere eseguiti
- **linguaggi macchina:**
 - i più lontani dal linguaggio naturale
 - istruzioni codificate in sequenze di bit
 - dipendenti dai dettagli dell'architettura della macchina
 - non necessitano di una “traduzione” per essere eseguiti

Linguaggi di programmazione

Alto Livello

C

pascal

basic

C++

Java

Python

`b = a + 1`

Assembler

```
MOV AX, [16]
ADD AX, 1
MOV [15], AX
```

Basso Livello

Linguaggio
macchina

```
010000 100000 000001
011000 000001 000001
010000 000001 011111
```

Dall'algoritmo al linguaggio macchina

Esempio di linguaggio macchina

- Un computer esegue soltanto programmi scritti in "linguaggio macchina"

	Comando	Param.	Traduz. Comando	Traduz. Param.
0	0100	00000000	(READ)	
1	0010	00100000	(STORE)	32
2	0001	00000000	(LOAD=)	0
3	0010	00100001	(STORE)	33
4	0000	00100000	(LOAD)	32
5	1100	00001101	(BEQ)	13
6	0100	00000000	(READ)	
7	0110	00100001	(ADD)	33
8	0010	00100001	(STORE)	33
9	0000	00100000	(LOAD)	32
10	0011	00000001	(SUB=)	1
11	0010	00100000	(STORE)	32
12	1000	00000100	(BR)	4
13	0000	00100001	(LOAD)	33
14	0101	00000000	(WRITE)	
15	1111	00000000	(END)	

Dall'algoritmo al linguaggio macchina

Esempio di linguaggio macchina

- Un computer esegue soltanto programmi scritti in "linguaggio macchina"
- Come si passa da un algoritmo descritto con un linguaggio di programmazione al linguaggio macchina?

	Comando	Param.	Traduz. Comando	Traduz. Param.
0	0100	00000000	(READ)	
1	0010	00100000	(STORE)	32
2	0001	00000000	(LOAD=)	0
3	0010	00100001	(STORE)	33
4	0000	00100000	(LOAD)	32
5	1100	00001101	(BEQ)	13
6	0100	00000000	(READ)	
7	0110	00100001	(ADD)	33
8	0010	00100001	(STORE)	33
9	0000	00100000	(LOAD)	32
10	0011	00000001	(SUB=)	1
11	0010	00100000	(STORE)	32
12	1000	00000100	(BR)	4
13	0000	00100001	(LOAD)	33
14	0101	00000000	(WRITE)	
15	1111	00000000	(END)	

Linguaggi di alto livello

I linguaggi ad alto livello necessitano di un “traduttore” che renda la codifica eseguibile dal calcolatore.

Questo processo può avvenire in due maniere distinte a seconda del tramite: **interprete** o **compilatore**.

- **Interprete:**

- 1 verifica sintattica istruzione corrente
- 2 traduzione istruzione corrente
- 3 esecuzione istruzione corrente
- 4 passaggio all'istruzione successiva

- **Compilatore:**

- 1 verifica sintattica dell'intero codice
- 2 compilazione del codice (traduzione in linguaggio macchina)
- 3 collegamento delle librerie esterne
- 4 esecuzione del programma

Risolvere un problema: algoritmo, programma e processo

Problema:

Data una sequenza di città, calcolare la lunghezza del percorso che le unisce.

Risolvere un problema: algoritmo, programma e processo

Problema:

Data una sequenza di città, calcolare la lunghezza del percorso che le unisce.

Algoritmo

Input Coordinate delle città

Output lunghezza percorso

- leggere da terminale le coordinate
- calcolare le distanze tra le città
- sommare le distanze
- stampa a video la lunghezza del percorso

Statico

Risolvere un problema: algoritmo, programma e processo

Problema:

Data una sequenza di città, calcolare la lunghezza del percorso che le unisce.

Algoritmo

Input Coordinate delle città

Output lunghezza percorso

- leggere da terminale le coordinate
- calcolare le distanze tra le città
- sommare le distanze
- stampa a video la lunghezza del percorso

Statico

Programma

```
import math

haltFlag = 1
lunghezza = 0
print "inserire le coordinate delle città':"
print "un valore negativo termina l'inserimento"
xOld = input("ascissa ")
yOld = input("ordinata ")
while (haltFlag > 0):
    x = input("ascissa ")
    y = input("ordinata ")
    if (x < 0 or y < 0):
        haltFlag = 0
    else:
        deltaX = x - xOld
        deltaY = y - yOld
        lunghezza += (deltaX**2 + deltaY**2)
lunghezza = math.sqrt(lunghezza)
print "la lunghezza del percorso e' ", lunghezza
```

Risolvere un problema: algoritmo, programma e processo

Problema:

Data una sequenza di città, calcolare la lunghezza del percorso che le unisce.

Algoritmo

Input Coordinate delle città

Output lunghezza percorso

- leggere da terminale le coordinate
- calcolare le distanze tra le città
- sommare le distanze
- stampa a video la lunghezza del percorso

Statico

Programma

```
import math

haltFlag = 1
lunghezza = 0
print "inserire le coordinate del:
print "un valore negativo termina
xOld = input("ascissa ")
yOld = input("ordinata ")
while (haltFlag > 0):
    x = input("ascissa ")
    y = input("ordinata ")
    if (x < 0 or y < 0):
        haltFlag = 0
    else:
        deltaX = x - xOld
        deltaY = y - yOld
        lunghezza += (deltaX**2 + deltaY**2)
lunghezza = math.sqrt(lunghezza)
print "la lunghezza del percorso e' ",lunghezza
```

Processo

```
dario@vulcano: python percorso.py
inserire le coordinate delle città':
un valore negativo termina l'inserimento
ascissa 5
ordinata 5
ascissa 3
ordinata 3
ascissa -1
ordinata 0
la lunghezza del percorso e' 2.82842712475
dario@vulcano: python percorso.py
inserire le coordinate delle città':
un valore negativo termina l'inserimento
ascissa 2
ordinata 7
ascissa 3
ordinata 5
ascissa -4
ordinata 0
la lunghezza del percorso e' 2.2360679775
dario@vulcano:
```

Dinamico

Dal problema all'algoritmo

Data una sequenza di città,
calcolare la lunghezza del
percorso che le unisce.

Input Coordinate delle città

Output lunghezza percorso

- leggere da terminale le coordinate
- calcolare le distanze tra le città
- sommare le distanze
- stampa a video la lunghezza del percorso

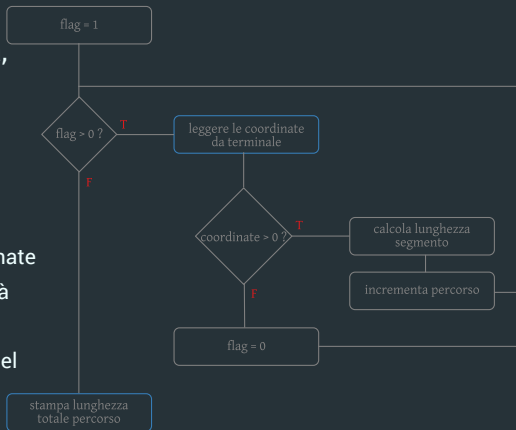
Dal problema all'algoritmo

Data una sequenza di città,
calcolare la lunghezza del
percorso che le unisce.

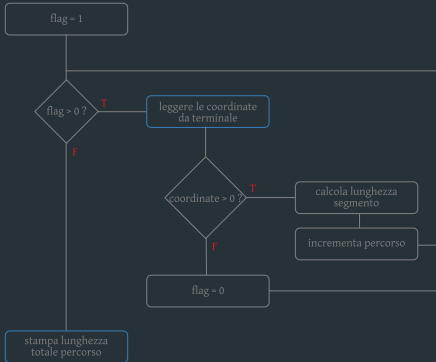
Input Coordinate delle città

Output lunghezza percorso

- leggere da terminale le coordinate
- calcolare le distanze tra le città
- sommare le distanze
- stampa a video la lunghezza del percorso



Dall'algoritmo al programma



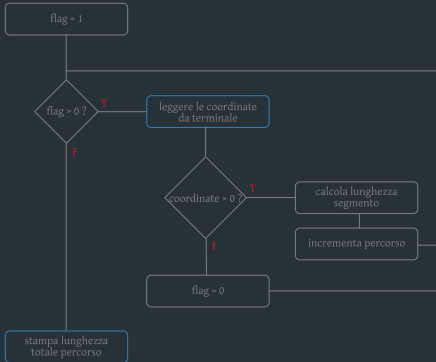
```
import math
```

```

haltFlag = 1
lunghezza = 0
print "inserire le coordinate delle città':"
print "un valore negativo termina l'inserimento"
xOld = input("ascissa ")
yOld = input("ordinata ")
while (haltFlag > 0):
    x = input("ascissa ")
    y = input("ordinata ")
    if (x < 0 or y < 0):
        haltFlag = 0
    else :
        deltaX = x - xOld
        deltaY = y - yOld
        lunghezza += (deltaX**2 + deltaY**2)
lunghezza = math.sqrt(lunghezza)
print "la lunghezza del percorso e' ",lunghezza

```

Dall'algoritmo al programma



```
import math
```

```

haltFlag = 1
lunghezza = 0
print "inserire le coordinate delle citta':"
print "un valore negativo termina l'inserimento"
xOld = input("ascissa ")
yOld = input("ordinata ")
while (haltFlag > 0):
    x = input("ascissa ")
    y = input("ordinata ")
    if (x < 0 or y < 0):
        haltFlag = 0
    else :
        deltaX = x - xOld
        deltaY = y - yOld
        lunghezza += (deltaX**2 + deltaY**2)
lunghezza = math.sqrt(lunghezza)
print "la lunghezza del percorso e' ", lunghezza

```

```

dario@vulcano: dir
total 4.0K
-rw-r--r-- 1 dario staff 495 Aug 10 11:16 percorso.py

```

Dal programma al processo

```
darío@vulcano: ipython percorso.py  
inserire le coordinate delle città':  
un valore negativo termina l'inserimento  
ascissa _
```

Dal programma al processo

```
darío@vulcano: ipython percorso.py  
inserire le coordinate delle città':  
un valore negativo termina l'inserimento  
ascissa _
```

Esecuzione di una istanza = processo

Dal programma al processo

```
dario@vulcano: python percorso.py
inserire le coordinate delle citta':
un valore negativo termina l'inserimento
ascissa 5
ordinata 5
ascissa 3
ordinata 3
ascissa -1
ordinata 0
la lunghezza del percorso e' 2.82842712475
dario@vulcano: python percorso.py
inserire le coordinate delle citta':
un valore negativo termina l'inserimento
ascissa 2
ordinata 7
ascissa 3
ordinata 5
ascissa -4
ordinata 0
la lunghezza del percorso e' 2.2360679775
dario@vulcano:
```

Dal programma al processo

```
dario@vulcano: python percorso.py
inserire le coordinate delle citta':
un valore negativo termina l'inserimento
ascissa 5
ordinata 5
ascissa 3
ordinata 3
ascissa -1
ordinata 0
la lunghezza del percorso e' 2.82842712475
dario@vulcano: python percorso.py
inserire le coordinate delle citta':
un valore negativo termina l'inserimento
ascissa 2
ordinata 7
ascissa 3
ordinata 5
ascissa -4
ordinata 0
la lunghezza del percorso e' 2.2360679775
dario@vulcano:
```

output da due
istanze distinte
del programma

Sintassi e semantica dei linguaggi

- **Sintassi.** L'insieme di regole formali che dettano le **modalità** per costruire frasi corrette nel linguaggio stesso.
- **Semantica.** l'insieme dei **significati** da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio.
- Una frase può essere sintatticamente corretta e tuttavia non avere significato.
 - semanticamente corretta \rightarrow sintatticamente corretta
 - sintatticamente corretta \nrightarrow semanticamente corretta

Sintassi vs Semantica

L'utenza potenziale porta avanti la verifica critica degli obbiettivi istituzionali e l'individuazione di fini qualificanti in una visione organica e ricondotta a unità recuperando ovvero rivalutando nella misura in cui ciò sia fattibile un indispensabile salto di qualità.

Sintassi vs Semantica

L'utenza potenziale porta avanti la verifica critica degli obbiettivi istituzionali e l'individuazione di fini qualificanti in una visione organica e ricondotta a unità recuperando ovvero rivalutando nella misura in cui ciò sia fattibile un indispensabile salto di qualità.

- Che cosa vuol dire?

Sintassi vs Semantica

L'utenza potenziale porta avanti la verifica critica degli obbiettivi istituzionali e l'individuazione di fini qualificanti in una visione organica e ricondotta a unità recuperando ovvero rivalutando nella misura in cui ciò sia fattibile un indispensabile salto di qualità.

- Che cosa vuol dire?
- Dal punto di vista sintattico la frase è corretta

Sintassi vs Semantica

L'utenza potenziale porta avanti la verifica critica degli obbiettivi istituzionali e l'individuazione di fini qualificanti in una visione organica e ricondotta a unità recuperando ovvero rivalutando nella misura in cui ciò sia fattibile un indispensabile salto di qualità.

- Che cosa vuol dire?
- Dal punto di vista sintattico la frase è corretta
- Dal punto di vista semantico, la frase non vuol dire niente

Dal programma al programma corretto: debugging

Esistono **tre tipi di errori** che bisogna eliminare perchè un programma sia corretto:

1 **sintattici**

I linguaggi di programmazione sono molto più rigidi dei linguaggi naturali, l'interprete deve essere guidato tramite una struttura sintattica molto rigida. Ogni deviazione da quanto atteso dall'interprete genera un errore.

2 **esecutivi**

Il calcolatore è un insieme di risorse condivise da diversi utenti e processi, ogni tentativo di accesso ad "aree" non pertinenti o la generazione di risultati non "coerenti" induce il blocco dell'esecuzione del programma.

3 **semantici**

L'interprete non assegna un significato all'insieme di istruzioni che gli abbiamo impartito, e questo può differire dalle nostre intenzioni.

Dal programma al programma corretto: debugging

Errori sintattici:

```
In [4]: dòlkjsklldjlkas ljdhasj kjsnxJoe 8793217 jdioajs
```

Dal programma al programma corretto: debugging

Errori sintattici:

```
In [4]: d0lkjsklldjlkas ljdhasj kjsnxJoe 8793217 jdioajs
File "<ipython-input-4-8cd1115f2298>", line 1
    d0lkjsklldjlkas ljdhasj kjsnxJoe 8793217 jdioajs
      ^
SyntaxError: invalid syntax
```


Dal programma al programma corretto: debugging

Errori sintattici:

```
In [4]: d0lkjskldjlkas ljdhasj kjsnxJoe 8793217 jdioajs
File "<ipython-input-4-8cd1115f2298>", line 1
    d0lkjskldjlkas ljdhasj kjsnxJoe 8793217 jdioajs
      ^
SyntaxError: invalid syntax
```

```
In [5]: if (lato > 2)
```

Dal programma al programma corretto: debugging

Errori sintattici:

```
In [4]: d0lkjskljdkas ljdhasj kjsnxJoe 8793217 jdioajs
File "<ipython-input-4-8cd1115f2298>", line 1
    d0lkjskljdkas ljdhasj kjsnxJoe 8793217 jdioajs
      ^
SyntaxError: invalid syntax
```

```
In [5]: if (lato > 2)
File "<ipython-input-5-bd95c7fe8be2>", line 1
    if (lato > 2)
      ^
SyntaxError: invalid syntax
```

Dal programma al programma corretto: debugging

Errori esecutivi:

```
In [7]: print lato
```

Dal programma al programma corretto: debugging

Errori esecutivi:

```
In [7]: print lato
```

```
-----  
NameError
```

```
Traceback (most recent call last)
```

```
<ipython-input-7-9716ab3e5b6b> in <module>()
```

```
----> 1 print lato
```

```
NameError: name 'lato' is not defined
```

Dal programma al programma corretto: debugging

Errori esecutivi:

```
In [7]: print lato
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-7-9716ab3e5b6b> in <module>()  
----> 1 print lato
```

```
NameError: name 'lato' is not defined
```

```
In [18]: lati = [3, 7]
```

```
In [19]: print lati [3]
```

Dal programma al programma corretto: debugging

Errori esecutivi:

```
In [7]: print lato
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-7-9716ab3e5b6b> in <module>()  
----> 1 print lato  
  
NameError: name 'lato' is not defined
```

```
In [18]: lati = [3, 7]
```

```
In [19]: print lati [3]
```

```
-----  
IndexError                                Traceback (most recent call last)  
<ipython-input-19-7a81e9b29d47> in <module>()  
----> 1 print lati [3]  
  
IndexError: list index out of range
```

Dal programma al programma corretto: debugging

Errori semantici:

```
In [20]: base = 5
```

```
In [21]: altezza = 'nove'
```

```
In [22]: area = base * altezza
```

```
In [23]: print area
```

Dal programma al programma corretto: debugging

Errori semantici:

```
In [20]: base = 5
```

```
In [21]: altezza = 'nove'
```

```
In [22]: area = base * altezza
```

```
In [23]: print area
```

```
novenovenovenovenove
```

```
In [24]: _
```


Come si programma in Python

Esistono due forme di interazione con l'interprete:

- da linea di comando

- 1 avvio dell'interprete dei comandi `python`
- 2 introduzione dell'istruzione
- 3 output
- 4 introduzione dell'istruzione
- 5 output
- 6 ...

- da codice sorgente

- 1 scrittura del file sorgente

Creazione, con un editor di testo, del file `sorgente.py` che contiene le istruzioni nel linguaggio di programmazione Python.

- 2 esecuzione del programma

Il calcolatore interpreta ed esegue la sequenza di istruzioni codificate all'interno del programma.

`python sorgente.py`

da linea di comando: python

Prompt >>> per inserire il comando

```
dario@vulcano: python
Python 2.7.10 (default, May 25 2015, 13:06:17)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.56)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> latoA=3
>>> latoB=5
>>> area=latoA*latoB
>>> print area
15
>>> _
```

per uscire dal prompt digitare: `exit()` oppure `ctrl + d`

da linea di comando: ipython

In[] per inserire il comando

```
dario@vulcano: ipython
Python 2.7.10 (default, Oct 5 2015, 09:41:46)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.0 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.

In [1]: latoA = 3

In [2]: latoB = 5

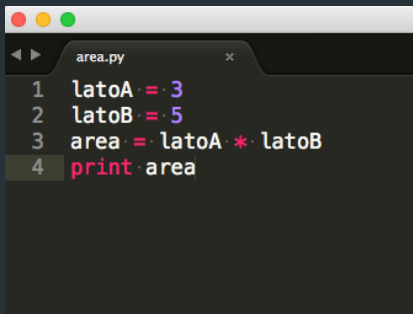
In [3]: area = latoA * latoB

In [4]: print area
15

In [5]: _
```

per uscire dal prompt digitare: `exit()` oppure `ctrl + d`

file sorgente



```
1 latoA = 3
2 latoB = 5
3 area = latoA * latoB
4 print area
```

```
dario@vulcano: python area.py
```

```
15
```

```
dario@vulcano: _
```

notebook

localhost:8888/notebooks/Untitled.ipynb?kernel_name=python2

jupyter Untitled

File Edit View Insert Cell Kernel Help Python 2

Code Cell Toolbar: None

```
In [1]: latoA = 5
```

```
In [2]: latoB = 7
```

```
In [3]: area = latoA * latoB
```

```
In [4]: print area
```

35

```
In [ ]:
```