```
< raw_input() indica che quello che stà al suo interno è una stringa?
> raw_input() chiede un input all'utente, e qualsiasi cosa venga inserita sarà
  di sicuro una stringa

< int() a cosa serve? Serve a dichiarare una variabile?
> No, in Python le variabili non si dichiarano *mai*. int() converte il valore
  in un numero intero. Esempio:

  a="123" # Stringa "123"
  b=int(a) # Numero 123

  Se proprio bisogna inizializzare una variabile in una certa maniera... Si usa:

  a=[]
  for x in xrange(5):
      a.append(x) # Se non avessimo dato a=[], il comando append() fallirebbe

< try è simile al "vecchio" if?
> Assolutamente no. Il ciclo try/except viene usato quando le operazioni
  all'interno di try potrebbero fallire, causando un crash del programma. Grazie
  ad except, il programma ne esegue altre se il primo blocco di istruzioni
  fallisce. Esistono vari tipo di except, come except ValueError, che viene
  usato se viene richiamata una variabile inesistente o un valore fuori range...
  Esempio:

  a=["a", "b", "c"]
  try:
      a[3] # ValueError, a[3] non esiste (ricordiamoci che si parte da a[0])
  except ValueError:
      print "ValueError"

  Poi ne esistono tantissimi vari tipi, come IOError, oppure riferito a una
  funzione... Tipo:

  try:
      sock.bind(("", port if address == "server" else 0))
  except socket.error: # Errore
      sys.exit("%(address)s:%(port)s already in use" % vars())

< a[] ricorda il "read|readln" di pascal?
> Non c'entra nulla... a[] è privo di senso, infatti è a=[], ovvero, a è uguale
  a una lista vuota.

< Puoi rispiegare il funzionamento di xrange()?
> Certo. xrange(numero_intero) serve a creare una lista composta di numeri da 0
  a numero_intero-1. Usatissimo nei cicli for. Esempio:

  for x in xrange(5) # => [0, 1, 2, 3, 4]
      print x # x assume il primo (poi il secondo, terzo...) valore della lista

< for x in xrange(5): significa che vengono attribuiti ad x i valori da 0 a 4
  in modo progressivo?
> Si. Potremmo anche usare:

  for x in ["a", "b", "x"]:
      print x

  Python è molto dinamico... Si limita ad attribuire ad x in maniera progressiva
  i valori della lista... Se vogliamo invece fargli ripetere una operazione tot
  volte, si usa xrange che ci semplifica la vita. Molto spesso si usa in azione
  combinata con len() (che ricordiamo, calcola la lunghezza di una variabile).
```

```
  Esempio:

  a="Ho fame"
  for x in xrange(len(a)): # Per tot_volte = lunghezza di a
      print a[x]

< la funzione len() tiene conto anche degli spazi vuoti?
> Tiene conto di *tutto*. Calcola la lunghezza di liste, stringhe, ma non
  (ovviamente) di numeri (se non convertendoli prima in stringhe con str()).

< int() e str() sono una l'inversa dell'altra? o cambia qlc altro?
> Non è detto... int() può rendere interi numeri a virgola mobile, e str() fa
  diventare stringhe anche le liste :D Come "[0, 1, 2, 3]" per esempio.

< Esiste anche la funzione raw_output()?
> No, non ce n'è bisogno... Si usa print semplicemente, in quanto nell'output
  non ti devi preoccupare del tipo di cosa che vai a visualizzare.

< print float(a)/b indica che si accettano virgoloni nel risultato della
  divisione?
> Si più o meno... Nel senso che si usa float(a) così a diventa un numero con la
  virgola (anche se ,00), così diviso per b avremo un numero a sua volta con la
  virgola... Altrimenti avremmo una divisione tra interi senza virgole né resto.

< Ma in: print float(a)/b , vale il discorso sul numero di cifre significative
  (nel senso che se a ha 5 cifre significative e b anche, allora anche il
  risultato dell'operazione dovrà averne lo stesso numero oppure no?)
> No, non vale. Fa una divisione matematica.
```