

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# Questa è la prima linea di un programma in python. Viene usata anche in altri
# linguaggi. Dice semplicemente con cosa eseguire il file. Ora gli ho detto di
# usare l'interprete Python. Non è una parte del programma, non viene eseguita
# (quindi la potremo definire "commento")... Serve semplicemente al sistema
# per sapere come comportarsi.
# La seconda linea indica la codifica del file (UTF-8, ISO-8859 o cosa vuoi).
# Noi usiamo UTF-8 sotto Linux, MacOS X e Windows da NT in su.
# L'interprete python legge questa riga per riconoscere correttamente tutti i
# caratteri usati in questo file.
# Queste due linee sono un must. Si inseriscono sempre in ogni file (o quasi).
# Ora iniziamo a programmare! :D

##### Input/output

cognome = raw_input("Ciao Leo, qual è il tuo cognome? ") # Prendo il cognome
print "Il tuo cognome è", cognome + "." # Visualizzo il cognome

# raw_input() prende quello che viene digitato sempre come stringa, quindi sai
# sempre che il risultato deve essere una stringa
# Differenza tra l'operatore "," e "+":
# * "+" si usa per i numeri, oppure se dobbiamo concatenare due stringhe senza
# creare lo spazio " " tra le due.
# * "," concatena una stringa e un numero, o due stringhe creando uno spazio
# " " tra di esse.

##### Try/Except

anni = raw_input("Quanti anni hai? ") # E non mentire! ;)
try:
    anni = int(anni)
except:
    print "Diamine, non hai inserito un numero lurido bastardo!"
    print "Allora faccio finta che tu ne abbia 12! :P"
    anni = 12
print "Hai", anni, "anni."

# Try/except (figata del python, scordati la gestione degli errori figa in c):
# try:
#     # Blocco di istruzioni
#     # Prova a fare queste istruzioni. Se fallisce, passa ad except
# except:
#     # Blocco di istruzioni
#     # Esegue queste istruzioni se try fallisce
# In questo esempio si passa ad except ogni volta che non inserisci un numero,
# in quanto non può trasformare in numero una stringa :)

##### If/elif/else

if anni < 8:
    print "Ma ciao picinin!"
elif anni < 16:
    print "Hai ancora da crescere..."
else:
    print "Wela boy!"

# If/elif/else:
# if condizione:
#     # Blocco di istruzioni da eseguire se la condizione è soddisfatta
```

```
# elif condizione:
#     # Blocco di istruzioni da eseguire se la condizione precedente non è
#     # soddisfatta ma lo è quella attuale
# else:
#     # Blocco di istruzioni da eseguire se nessuna condizione è soddisfatta
# N.b.: Questo sostituisce in tutto e per tutto la funzione switch degli altri
#     linguaggi di programmazione. E' molto utile usare elif perché così il
#     programma non deve guardare a tutte le condizioni anche quando si sa
#     che solo una è soddisfabile. Fa risparmiare tempo :)

##### For

for x in xrange(3):
    print "Ripeto questa istruzione tre volte."

# For:
# for x in lista/stringa:
#     # Blocco di istruzioni finché non termina la lista/stringa
# xrange(numero) è utile perché crea una lista partendo da 0 di numero elementi.
# x in questo caso assume il valore del punto della lista corrente.

for x in xrange(len(cognome)):
    print "Lettera:", cognome[x]

# Spiegazione:
# len(cognome) => lunghezza del cognome (ad esempio 5)
# xrange(len(cognome)) => crea una lista di numeri da uno a... (vedi sopra)
# cognome[x] => richiama la x lettera del cognome

##### While

x = 1
while x: # Si intende: finché x, ovvero finché x == 1
    altezza = raw_input("Quanto sei alto (in cm)? ")
    try:
        altezza = int(altezza) # Converte la stringa in un numero intero
        x = 0 # Cambia il valore di x, in modo che dopo esca da while
    except:
        print "Numero non valido, reinserire."

##### Break
# Semplifichiamo l'esempio fatto qui sopra con l'istruzione break
# L'istruzione break può essere usata sia in while sia in for

while: # Scrivendolo così crea un ciclo infinito
    altezza = raw_input("Quanto sei alto (in cm)? ")
    try:
        altezza = int(altezza)
        break # Esce in maniera forzata dal ciclo
    except:
        print "Numero non valido, reinserire."

# Per esempio, potremmo fare un ciclo per far sì che un utente sia obbligato
# ad inserire un input corretto...

while:
    pwd = raw_input("Password? ")
    if pwd == "1084":
        print "Accesso effettuato correttamente..."
        break
    else:
```

```
        print "Accesso negato!"

##### Operazioni di base

a = 2
b = 3

print a+b # Addizione
print a*b # Moltiplicazione
print a/b # Divisione. Attenzione, qui non calcola il resto... Bisogna fare...
print float(a)/b
print float(a) # Ti da l'idea di come trasforma il numero float...
print a**b # Elevazione a potenza
print float(13)/73 # Tante cifre quante servono
a += 1 # a = a + 1
a -= 1 # a = a - 1
print a

##### Liste e stringhe

lista = ["banane", "pere", "torrent", "gnocche", "chips"]
print lista[0] # Banane
print lista[2] # Torrent
print lista[0:2] # Da 0 a 2 (2 escluso) => ["banane", "pere"]
print lista[-1] # Ultimo elemento
print lista[-2] # Penultimo
print lista[0:-1] # Da 0 all'ultimo (ultimo escluso)
print lista[:-1] # E' la stessa cosa di sopra... quando c'è lo 0 si omette

# Queste operazioni si possono fare anche sulle stringhe!
nome = "leonardo"
print nome[:3]

# Quindi volendo, si può fare questo gioco anche con un nome che è dentro una
# lista... Basta aggiungere altre parentesi quadre!
print lista[0] # Banane
print lista[0][:3] # Ban

# Volendo possiamo creare liste nelle liste...
lista_figa = [["a", "b", "c"], "linux"]
print lista_figa[0] # ["a", "b", "c"]

# Ci basta richiamare il valore così:
print lista_figa[0][0] # "a"

# E un ciclo for che visualizzi il contenuto della lista_figa[0]
for x in xrange(len(lista_figa[0])):
    print lista_figa[0][x]

# Possiamo fare anche altre operazioni sulle liste e sulle stringhe
var1 = "mammamia"
var2 = list(var1) # Converte var1 in una lista ['m', 'a', 'm', 'm', 'a', ...]
print var2
var2.reverse() # Inverte l'ordine dei vari elementi
print var2

# carattere.join(lista) ci permette di ottenere una stringa delimitando i vari
# valori con il carattere scelto
print "".join(var2) # "aimammam"
print "-".join(var2) # "a-i-m-a-m-m-a-m"
```

```
# Per suddividere una stringa (e trasformarla così in una lista), usiamo questo:
var3 = "io sono molto figo"
print var3.split(" ") # ["io", "sono", "molto", "figo"]

##### Assegnamento multiplo
# Consiste nel dare valori a più variabili contemporaneamente

# blocco1 = lista_figa[0]
# blocco2 = lista_figa[1]
blocco1, blocco2 = lista_figa[0], lista_figa[1]

# Lo possiamo anche usare per invertire due variabili...
a, b = 2, 3
print a, b # 2 3
a, b = b, a # Inverte (senza l'uso di una terza variabile)
print a, b # 3 2

##### Conversioni
# Come convertire le varie variabili

var4 = "184" # Stringa
var4 = int(var4) # Intero
var4 = float(var4) # Numero a virgola mobile con "il giusto numero" di decimali
var4 = str(var4) # Torna una stringa
var4 = list(var4) # Diventa una lista
var4 = "".join(var4) # Torna una stringa

##### Cicli for "accorciati" (programmazione funzionale)
# Rendere più brevi e stilosi dei cicli for

var5 = "mi piace la pizza"
var5 = [x for x in var5]
# (istruzione) for (x) in (lista/stringa)
# al posto di...
# for (x) in (lista/stringa):
#   (istruzione)

# Quindi...
print [x for x in xrange(3)] # ["0", "1", "2"]

# Volendo possiamo renderlo più articolato...
print [int(x) for x in xrange(3)] # Li rende interi

##### Funzioni
# Le utilissime funzioni: eliminano il codice rindondante

# Definisce una funzione con due parametri
def funzione(var1, var2):
    return var1*var2
# Passa due parametri e visualizza il risultato
print funzione(2, 1)

# N.b.: specificare sempre tanti argomenti quanti ne sono richiesti
# Per evitare questo problema si può scrivere codice di questo tipo:
def funzione(var1, var2=2):
    return var1*var2
print funzione(a, b)
print funzione(a)

# In parole povere, se la variabile non viene passata, si assume un valore fisso
```