Politecnico
di Milano

Coding in Python

10-14 Giugno 2019

Obiettivo

- Imparare i concetti di base della programmazione
- Imparare a usare Python come linguaggio di programmazione
- Imparare a risolvere piccoli problemi di programmazione

Persone coinvolte

- Lezioni
  - Elisabetta Di Nitto
    - Via Golgi, 42
    - email elisabetta.dinitto@polimi.it
    - tel: 02-2399-3663
    - http://dinitto.faculty.polimi.it
- Tutor
  - Gianmarco Loliva (mattina e pomeriggio)
    - email gianmarco.loliva@mail.polimi.it
  - Marco Arquati (pomeriggio)
    - email marco2.acquati@mail.polimi.it
- Aspetti logistici e amministrativi
  - Barbara Di Santo
    - Via Ponzio 34/5
    - email scuole-deib@polimi.it
    - tel: 02-2399-9607

Organizzazione del laboratorio

- Mattina dalle 9.00 alle 12.30
  - Lezioni
- Pomeriggio dalle 13.30 alle 17.00
  - Studio autonomo e svolgimento di esercizi guidato dai tutor
- Aula G.0.1

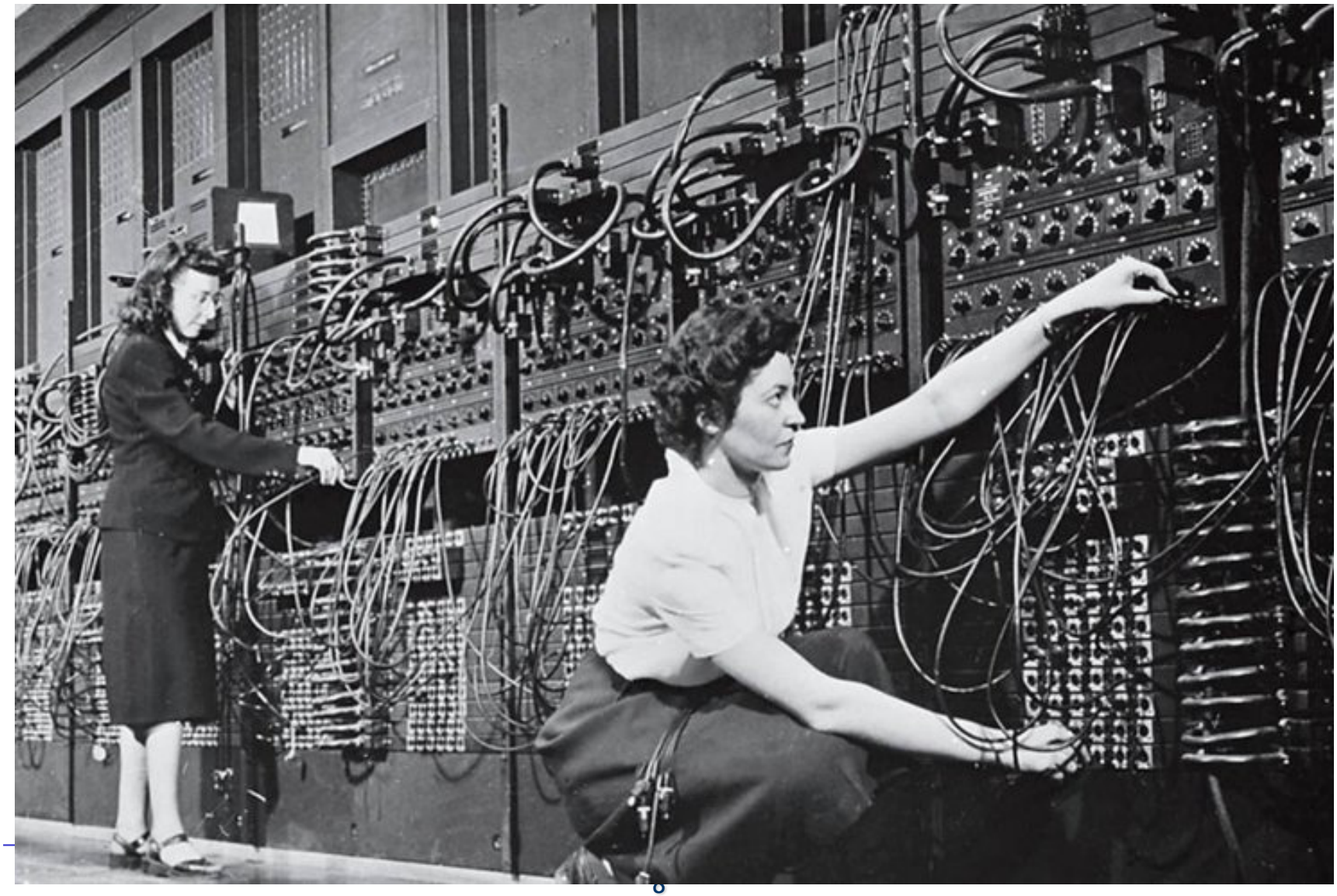
Materiale

- Libri online in Inglese
  - www.py4e.com
  - http://docs.python-guide.org/en/latest/ (argomenti più avanzati)
- Slide del corso

CHE COSA È LA PROGRAMMAZIONE (O CODING)?

Un po’ di storia: da macchine per svolgere un solo compito…

…a macchine programmabili…

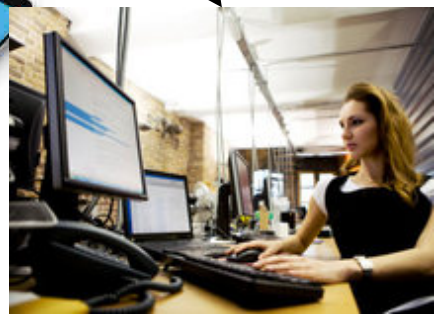


…a macchine programmabili un po' più maneggevoli…

...a???

https://www.youtube.com/watch?v=avP5d16wEp0

Utenti, computer, reti, programmatori

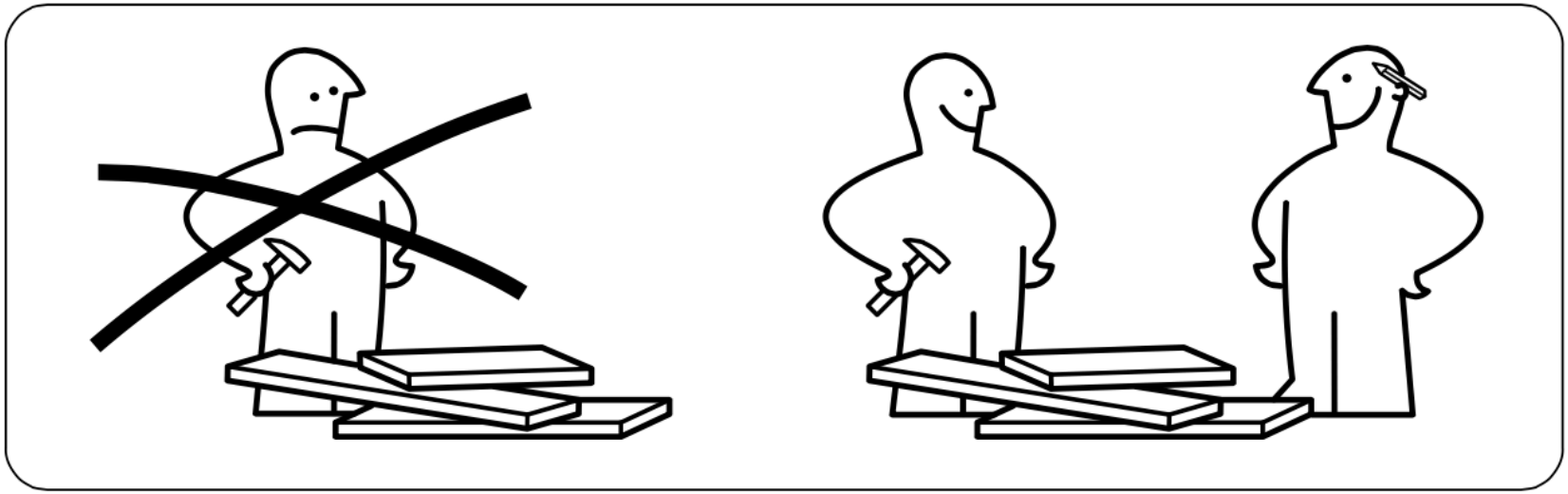


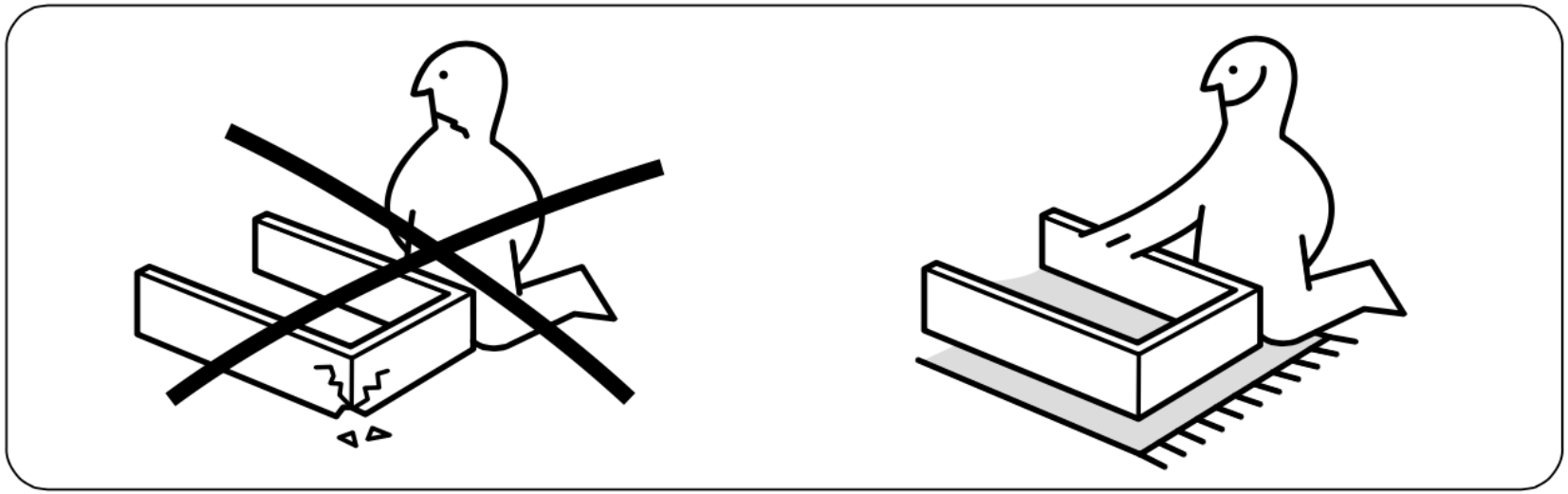
- I programmatori trasformano i dati in informazioni, secondo quanto necessario agli utenti
- Devono tenere conto di tanti vincoli
- I computer e la rete sono i loro alleati

Software, codice e programmi

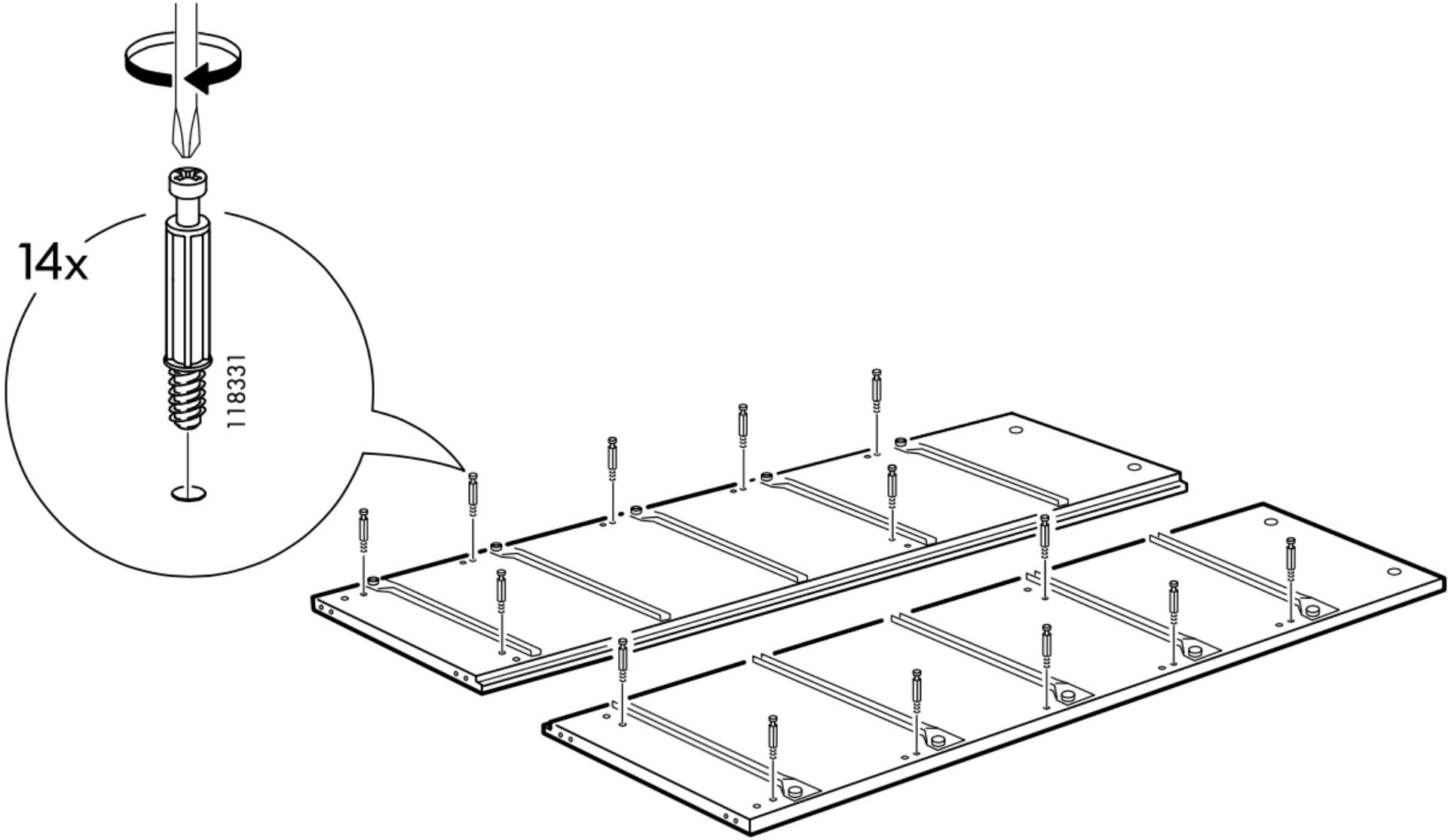
- Una sequenza di istruzioni
  - Può essere usato più volte per svolgere un certo compito
  - Può essere un pezzo d'arte creativa!

Programmi per umani





Programmi per umani



Programmi per umani

Divisione tra polinomi

Vediamo delle regole generali per effettuare la divisione tra due polinomi.

Consideriamo un polinomio A di grado m (dividendo) e un polinomio B di grado n (divisore), con $m \leq m$;

1. Si ordinano i polinomi secondo le potenze decrescenti della lettera x;
2. Si divide il primo termine del dividendo per il primo termine del divisore: il quoziente ottenuto è il primo termine del quoziente dei due polinomi;
3. Si moltiplica il termine in questione per il divisore e si somma il prodotto, cambiato di segno, con il dividendo; il polinomio ottenuto è il promo resto parziale;
4. Si divide il primo termine del resto parziale per il primo termine del divisore e si ottiene il secondo termine del quoziente dei polinomi;
5. Si moltiplica questo termine per il divisore e si somma il prodotto, cambiato di segno, con il precedente resto, ottenendo il secondo resto parziale;
6. Si procede in questo modo finché non si ottiene un resto parziale di grado inferiore al grado del divisore (questo ultimo resto sarà il resto della divisione).

Programmi per il computer …
Un esempio in Python


```
text = input('Write a sentence: ')

Alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
            'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
            's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

notAPangram = False

for letter in Alphabet:
    if letter not in text:
        notAPangram = True

if notAPangram:
    print('the string is not a pangram')
else:
    print('the string is a pangram')
```

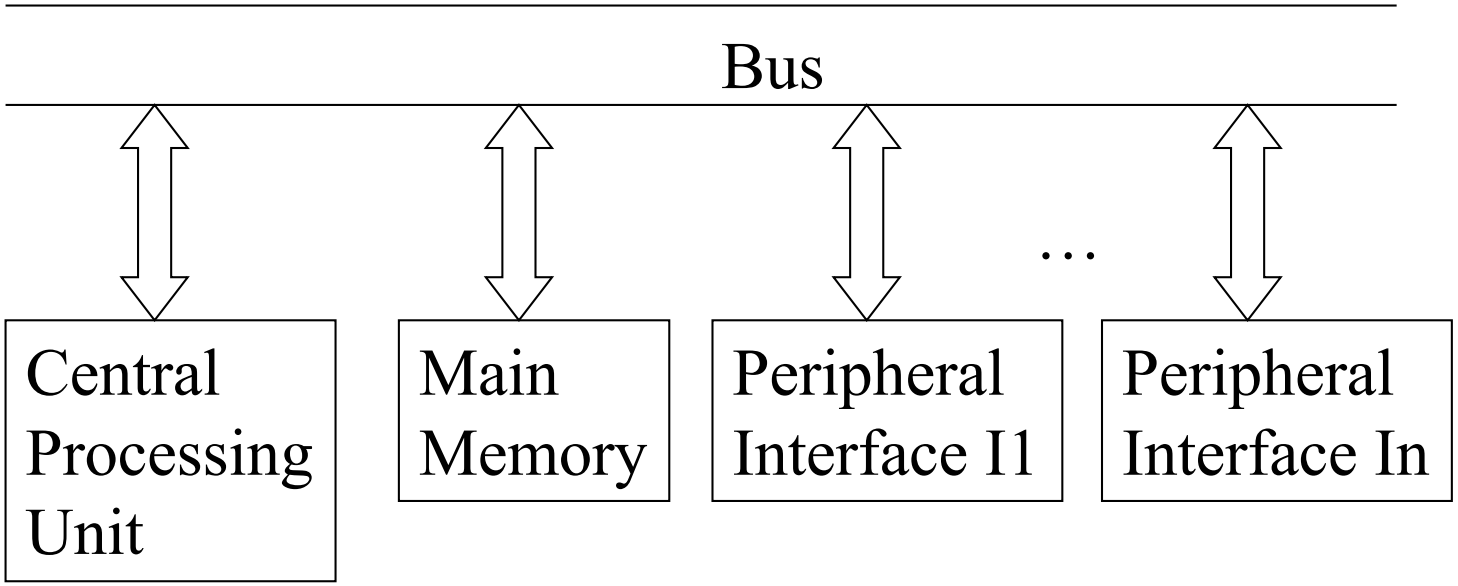
Programmi, esecutori e linguaggi di programmazione

- L'esecutore di un programma deve essere capace di capire il programma e di eseguirne le istruzioni
- Il programma deve essere definito in un linguaggio che l'esecutore capisce e di cui sa eseguire le istruzioni
- Qual è il linguaggio compreso da un computer?

Che cosa è un computer?

http://upload.wikimedia.org/wikipedia/commons/3/3d/RaspberryPi.jpg

Il modello di Von Neumann



Rappresentazione dell'informazione

- Codifica in forma binaria con sequenze finite di 1 e 0
- Unità minima di informazione: bit (binary digit)
- Byte: 8 bit
  - Posso usarlo per rappresentare 2^8 valori diversi (00000000, 00000001, 00000010, …, 11111111)
- Esempi
  - Numeri naturali: intervallo [0, 255]. $255 = 2^8-1$
  - Numeri interi: posso usare un bit per il segno e 7 per rappresentare il numero. Intervallo [-127 ($-(2^{(8-1)}-1)$), +127 ($2^{(8-1)}-1$)]
  - Caratteri: codifica ASCII (American Standard Code for Information Interchange)
    - "a": 01100001
    - "0": 00110000

Esempio di un programma scritto nel linguaggio del calcolatore (versione semplificata)

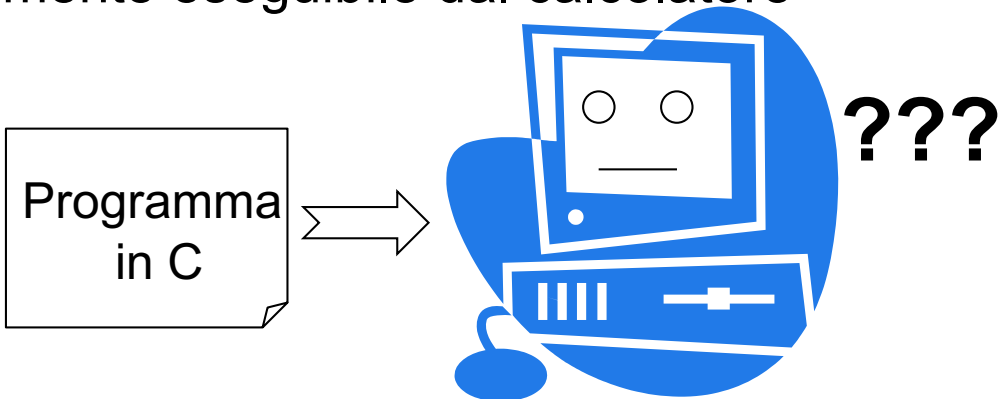
0000	0100000000001000	Acquisisci il primo numero e conservalo in 1000
0001	0100000000001001	Acquisisci il secondo numero e conservalo in 1001
0010	0000000000001000	Carica in A il numero conservato in 1000
0011	0001000000001001	Carica in B il numero conservato in 1001
0100	0110000000000000	Effettua la somma
0101	0010000000001010	Conserva in 1010 il valore che si trova in A
0110	0101000000001010	Stampa il contenuto di 1010
0111	1101000000000000	Termina l'esecuzione
1000		X
1001		Y
1010		Z

Linguaggio macchina vs altri linguaggi di programmazione

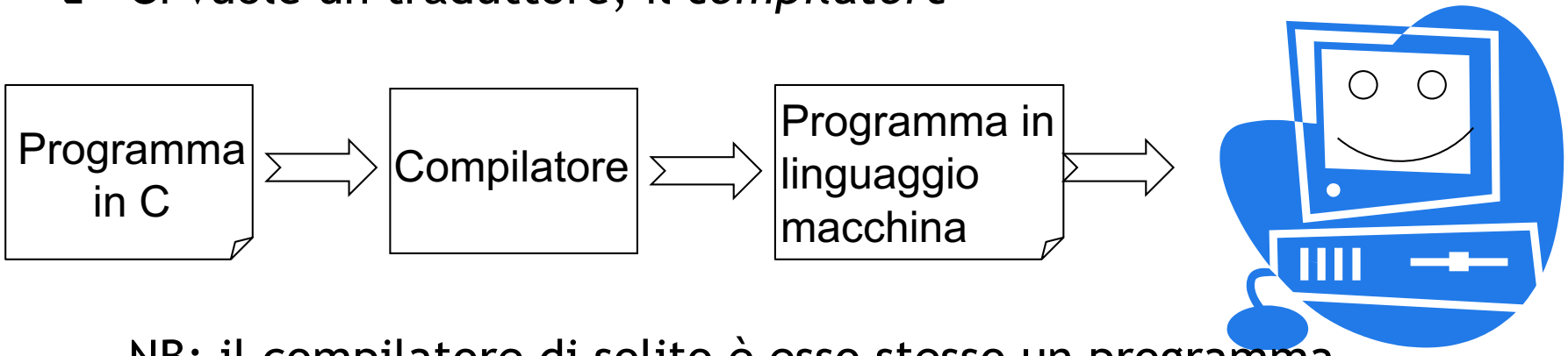
- Il linguaggio macchina è poco espressivo
  - Un compito semplice deve essere decomposto in più istruzioni macchina
- I linguaggi di programmazione di “alto livello” offrono una maggiore espressività e semplicità
  - Python, C, Java, C++, MATLAB…

Come fa la nostra macchina ad eseguire programmi in linguaggio di alto livello?

- Un programma scritto in un linguaggio di alto livello NON è direttamente eseguibile dal calcolatore



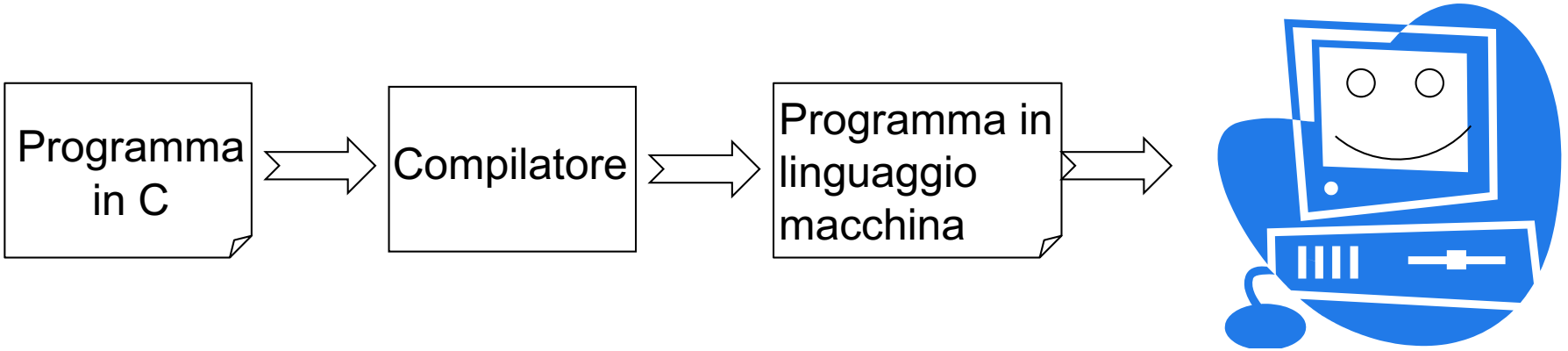
- Ci vuole un traduttore, il *compilatore*



- NB: il compilatore di solito è esso stesso un programma eseguito dal calcolatore

Come fa la nostra macchina ad eseguire programmi in linguaggio di alto livello?

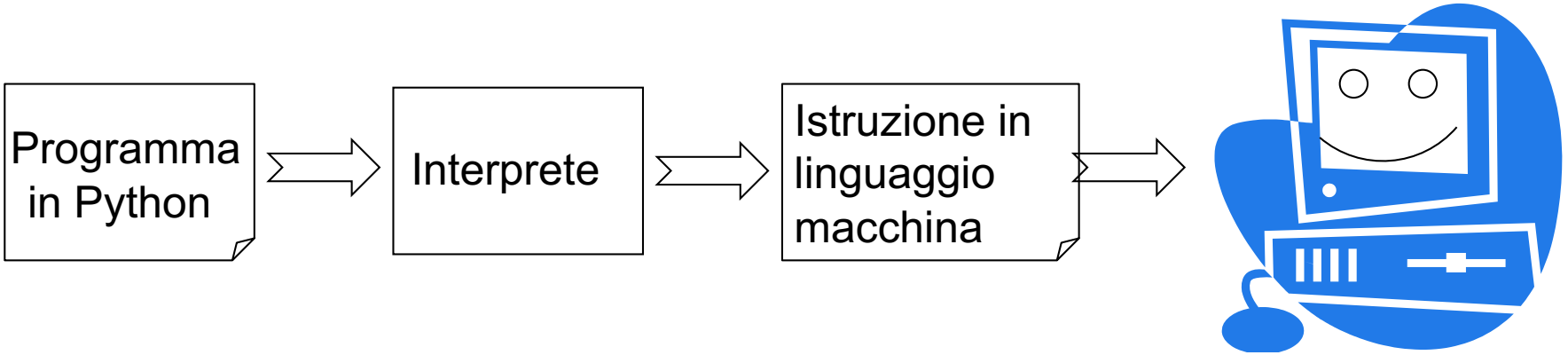
- Soluzione 1 – Compilatore: trasforma un programma in linguaggio macchina



- NB: il compilatore di solito è esso stesso un programma eseguito dal calcolatore

Come fa la nostra macchina ad eseguire programmi in linguaggio di alto livello?

- Soluzione 2 – Interprete: legge e traduce al volo durante l'esecuzione



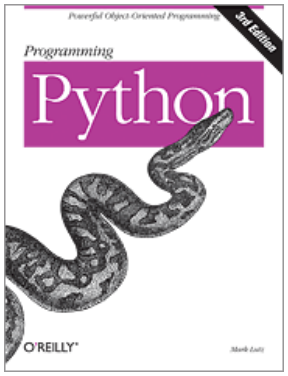
- Il compilatore traduce il programma PRIMA dell'esecuzione, l'interprete lo fa DURANTE l'esecuzione

Ambienti di programmazione

- Forniscono strumenti a supporto della programmazione
- In genere offrono
  - Editor per scrivere i programmi
  - Compilatore: crea un *codice oggetto* per ogni parte del programma
    - Linker: collega le varie parti del programma creando un unico *eseguibile*
    - Debugger: consente il controllo del programma durante la sua esecuzione
  - Oppure interprete

PYTHON COME LINGUAGGIO DI PROGRAMMAZIONE

Python è il linguaggio dell'interprete Python e di quelli che vogliono avere una conversazione con questo interprete.
Una persona che può parlare Python è chiamata Pythonista.
Il primo interprete è stato sviluppato da Guido van Rossum.



Errori sintattici: noi e il computer

- Dobbiamo imparare un nuovo linguaggio
- Faremo tanti errori
- Il computer non sarà comprensivo. Ci dirà “syntax error”. Ci sembrerà crudele.
- Ricordiamo:
  - è più semplice per noi imparare Python che per il computer imparare l’Italiano (o l’Inglese)
  - noi siamo quelli intelligenti e pazienti!

Strumenti per usare Python

- Interprete Python
  - Può essere scaricato e istallato da qui https://www.python.org
- Vari ambienti di programmazione
  - Spyder
  - Jupyter
  - Anaconda
  - https://www.pythonanywhere.com

Elementi di Python

- Vocabolario: parole riservate e variabili
- Struttura delle frasi: istruzioni
- Struttura della storia: costruire un programma che ha uno scopo

Parole riservate

- Parole che hanno un significato specifico

```
False   class   return  is      finally
None    if      for     lambda  continue
True    def     from    while   nonlocal
and     del     global  not     with
as      elif    try     or      yield
assert  else    import  pass
break   except  in      raise
```

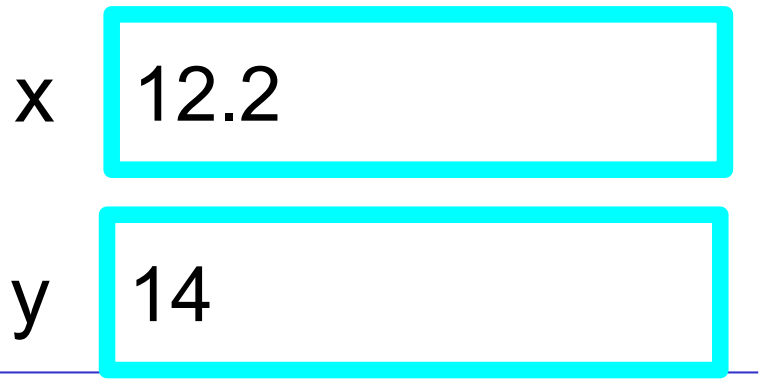
Variabili

- Una variabile corrisponde a una posizione nella memoria del calcolatore
- Può essere usata dai programmatori per immagazinare dati e poi recuperarli
- I programmatori attribuiscono un nome a ciascuna variabile (non possono usare le parole riservate come nomi di variabile)
- Il contenuto di una variabile può cambiare nel tempo

Programma

```
x = 12.2
y = 14
```

Memoria del calcolatore



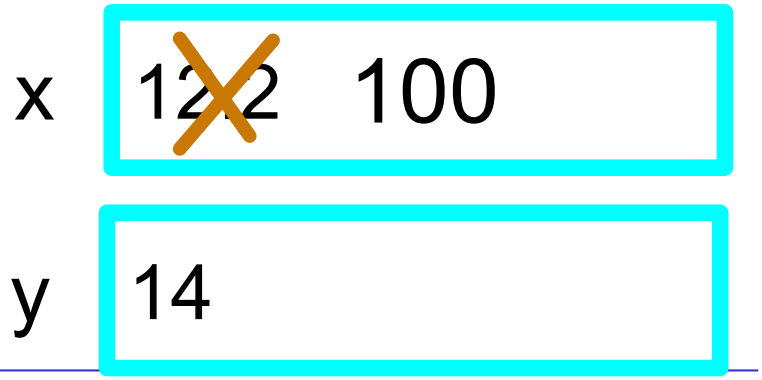
Variabili

- Una variabile corrisponde a una posizione nella memoria del calcolatore
- Può essere usata dai programmatori per immagazinare dati e poi recuperarli
- I programmatori attribuiscono un nome a ciascuna variabile (non possono usare le parole riservate come nomi di variabile)
- Il contenuto di una variabile può cambiare nel tempo

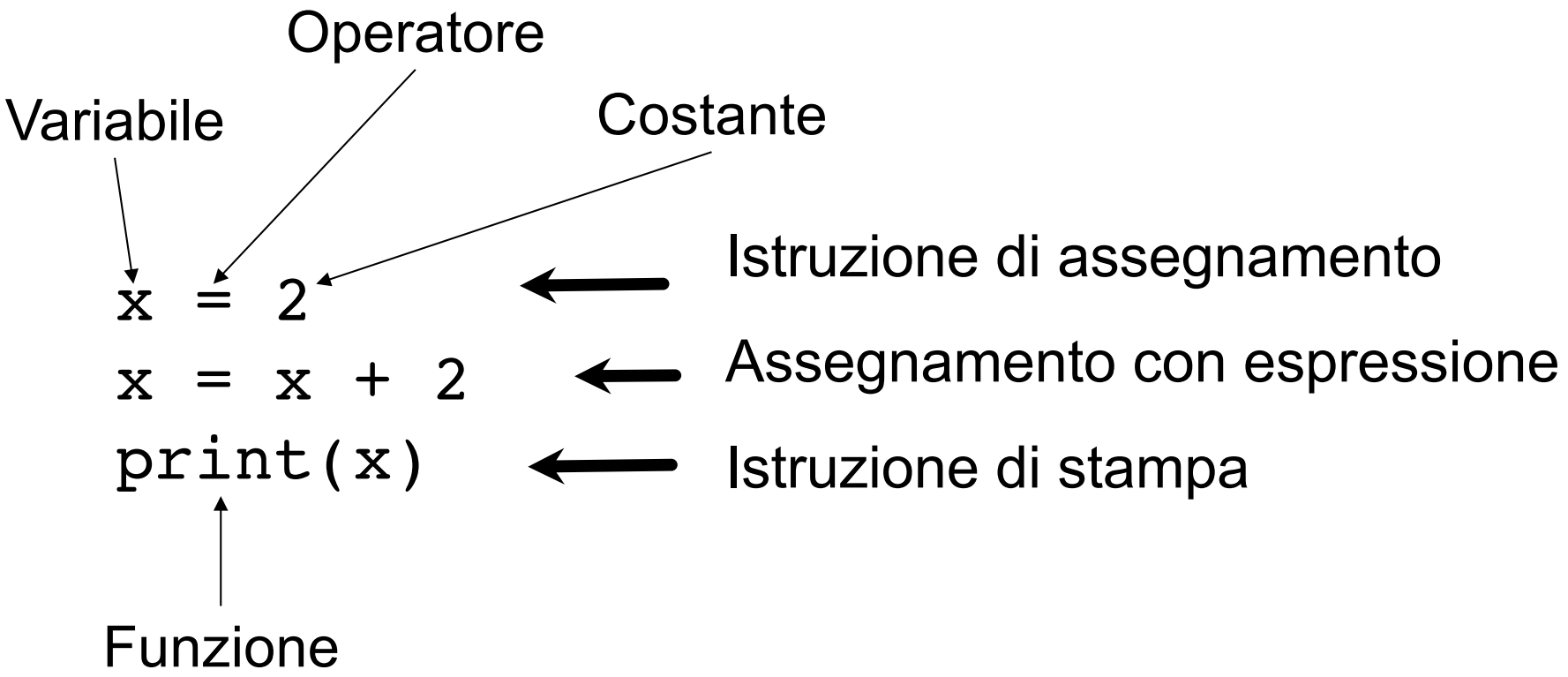
Programma

```
x = 12.2
y = 14
x = 100
```

Memoria del calcolatore



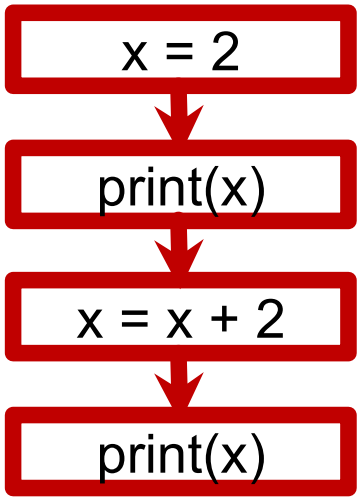
Frasi


Operatore
Variabile
Costante
x = 2
x = x + 2
print(x)
Istruzione di assegnamento
Assegnamento con espressione
Istruzione di stampa
Funzione

Flusso del programma

- Come una ricetta, un programma è una sequenza di istruzioni che devono essere eseguite in un certo ordine.
- Alcune istruzioni sono **condizionali** – possono essere saltate sotto certe condizioni.
- A volte, un'istruzione o un gruppo di istruzioni devono essere **ripetute**.
- A volte, raggruppiamo un insieme di istruzioni che devono essere usate in più punti di un programma o anche in programmi diversi (queste sono le **funzioni**).

Sequenza di istruzioni



Programma:

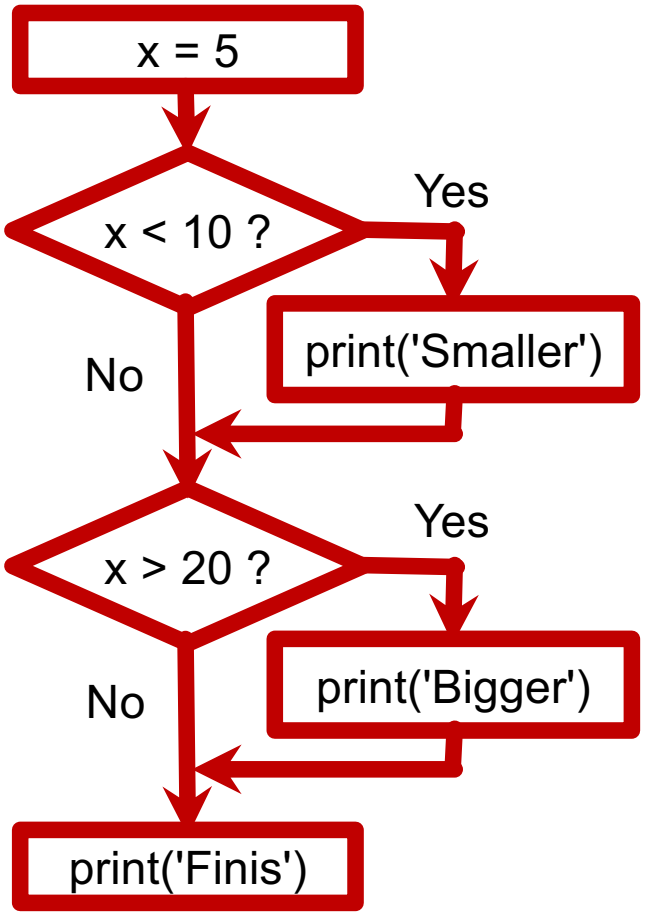
```
x = 2
print(x)
x = x + 2
print(x)
```

Output:

2
4

Quando un programma viene eseguito, le istruzioni di cui è costituito vengono eseguite in ordine. Come programmatori, noi definiamo i “percorsi” che il programma deve seguire.

Istruzioni condizionali



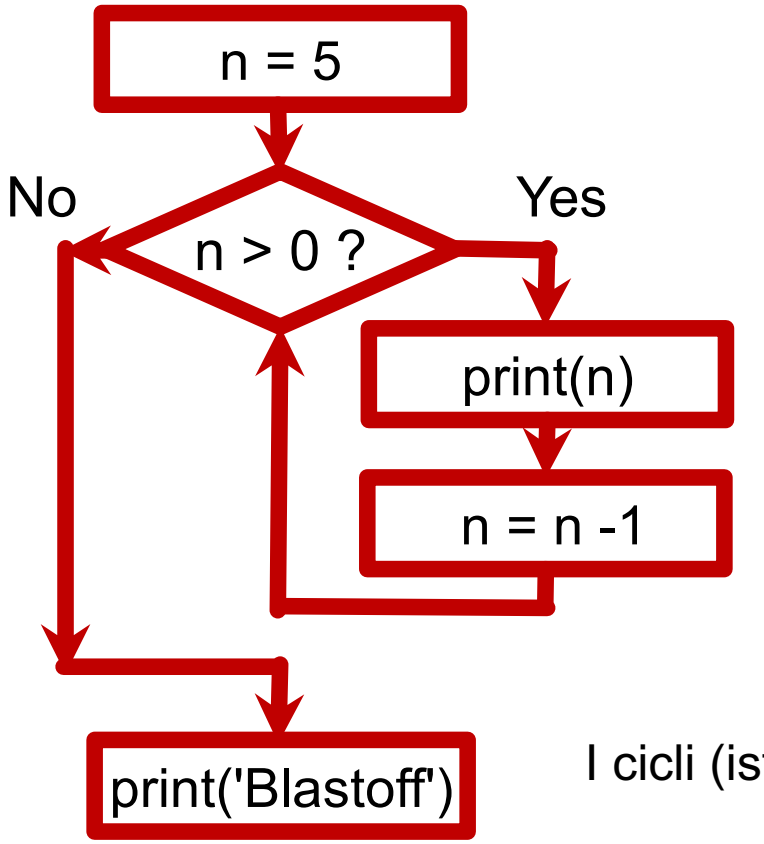
Program:

```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Finis')
```

Output:

```
Smaller
Finis
```

Istruzioni ripetute



Program:

```
n = 5
while n > 0 :
    print(n)
    n = n – 1
print('Blastoff!')
```

Output:

5
4
3
2
1
Blastoff!

I cicli (istruzioni ripetute) utilizzano una o più variabili che controllano l'esecuzione del ciclo.

Classifichiamo le istruzioni del nostro primo esempio…


```
text = input('Write a sentence: ')

Alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
            'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
            's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

notAPangram = False

for letter in Alphabet:
    if letter not in text:
        notAPangram = True

if notAPangram:
    print('the string is not a pangram')
else:
    print('the string is a pangram')
```

Sequenziale

Ciclo

Condizionale

Questa è una breve storia in Python che ci dice come riconoscere un pangramma

Due possibili modi di creare un programma in Python

- Interattivo
  - Scriviamo ogni istruzione direttamente nell'interprete Python e otteniamo immediatamente una risposta
- Script
  - Inseriamo una sequenza di istruzioni in un file utilizzando un editor di testo e chiediamo a Python di eseguire le istruzioni nel file
  - Come convenzione, salviamo il file con l'estensione ".py" per indicare che contiene un programma Python.

VARIABILI E COSTANTI

Costanti e variabili

- Costanti: numeri, lettere e stringhe che non cambiano il loro valore
  - Stringhe costanti sono racchiuse tra apici semplici (')
    o doppi (")
- Variabili: corrispondono ad aree di memoria nel calcolatore. Possono cambiare il loro valore nel tempo

```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Hello world')
Hello world
>>> x = 125
>>> print(x)
125
>>> x = ‘Ciao Mondo!’
>>> print(x)
Ciao Mondo!
```

Espressioni numeriche


```
>>> xx = 2
>>> xx = xx + 2
>>> print(xx)
4
>>> yy = 440 * 12
>>> print(yy)
5280
>>> zz = yy / 1000
>>> print(zz)
5.28
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print(kk)
3
>>> print(4 ** 3)
64
```

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
%	Remainder

Che cosa significa “tipo”?

- In Python variabili e costanti hanno un “tipo”
- Per esempio, numeri e stringhe sono tipi diversi
- Il risultato di un’espressione dipende da tipo degli operandi
- L’operazione “+” significa addizione se applicata a numeri e concatenazione se applicata a stringhe

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

Ogni valore ha un tipo

- Python conosce il tipo di qualsiasi valore
- Alcune operazioni sono proibite per alcuni valori
- È possibile chiedere a Python il tipo di una variabile/valore tramite la funzione type()

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):
File "<stdin>", line 1, in
<module>TypeError: Can't convert
'int' object to str implicitly
>>> type(eee)
<class'str'>
>>> type('hello')
<class'str'>
>>> type(1)
<class'int'>
>>>
```

Diversi tipi numerici

- Esistono due tipi principali di numeri
  - Interi:
    - -14, -2, 0, 1, 100, 401233
  - Numeri con la virgola (Floating Point Numbers) :
    - -2.5 , 0.0, 98.6, 14.0

```
>>> xx = 1
>>> type (xx)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class'float'>
>>>
```

Divisione intera

- La divisione intera produce un risultato float
  - In altri linguaggi di programmazione invece il risultato è un intero

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

Input dell'utente

- Si può chiedere a Python di fermarsi e leggere dati dall'utente usando la funzione input()
- input() restituisce una stringa

Programma

```
nam = input('Who are you? ')
print('Welcome', nam)
```

Esecuzione

Who are you? Chuck
Welcome Chuck

Convertire l'input dell'utente

- Se vogliamo acquisire un numero, dobbiamo convertire il risultato di input da stringa a intero

```
inp = input('Europe floor?')
usf = int(inp) + 1
print('US floor', usf)
```

Europe floor? 0
US floor 1

Commenti in Python

```
# This code transforms floor numbers from the
# European to the US convention
inp = input('Europe floor?')
usf = int(inp) + 1
print('US floor', usf)
```

- # indica l'inizio di un commento
- Perchè usare i commenti?
  - Per descrivere che cosa fa una certa sequenza di istruzioni
  - Per indicare l'autore del codice, la data, la versione, …
  - Per escludere temporaneamente un'istruzione dal programma

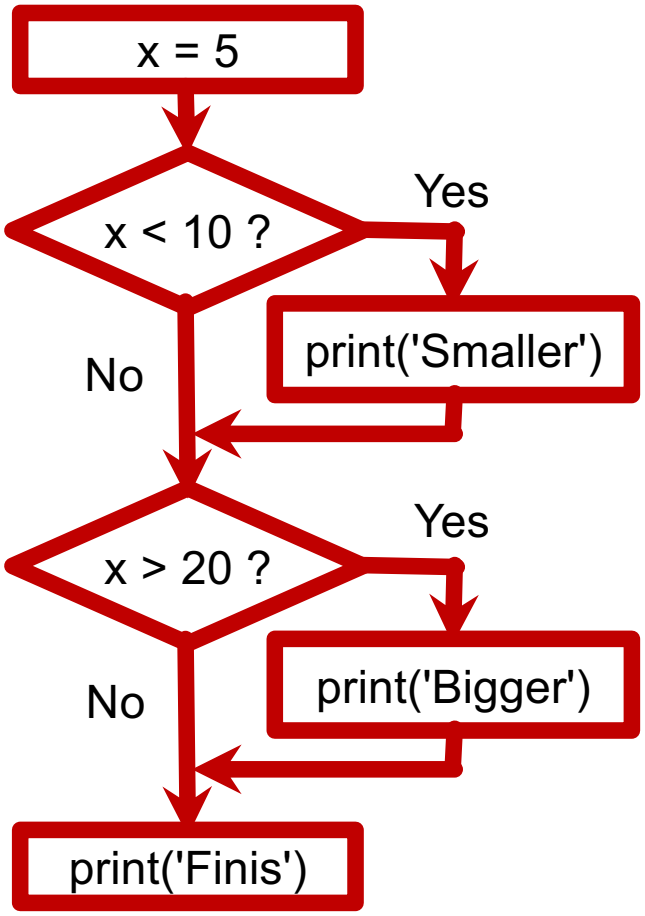
Esercizio

Vogliamo calcolare il salario di un impiegato. Scrivi un programma che chieda all'utente il numero di ore di lavoro e la paga oraria e calcoli il salario totale

```
Inserisci il numero di ore: 35
Inserisci la paga oraria: 2.75

Salario totale: 96.25
```

ANCORA SULLE ISTRUZIONI CONDIZIONALI

Istruzioni condizionali



Programma:

```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Finis')
```

Output:

```
Smaller
Finis
```

Operatori di confronto

- Gli operatori di confronto danno come risultato True / False
- Non modificano il valore delle variabili

Python	Significato
<	Minore di
<=	Minore o uguale a
==	Uguale a
>=	Maggiore o uguale a
>	Maggiore di
!=	Non uguale

RICORDA: “=” è usato per l’assegnamento!

http://en.wikipedia.org/wiki/George_Boole

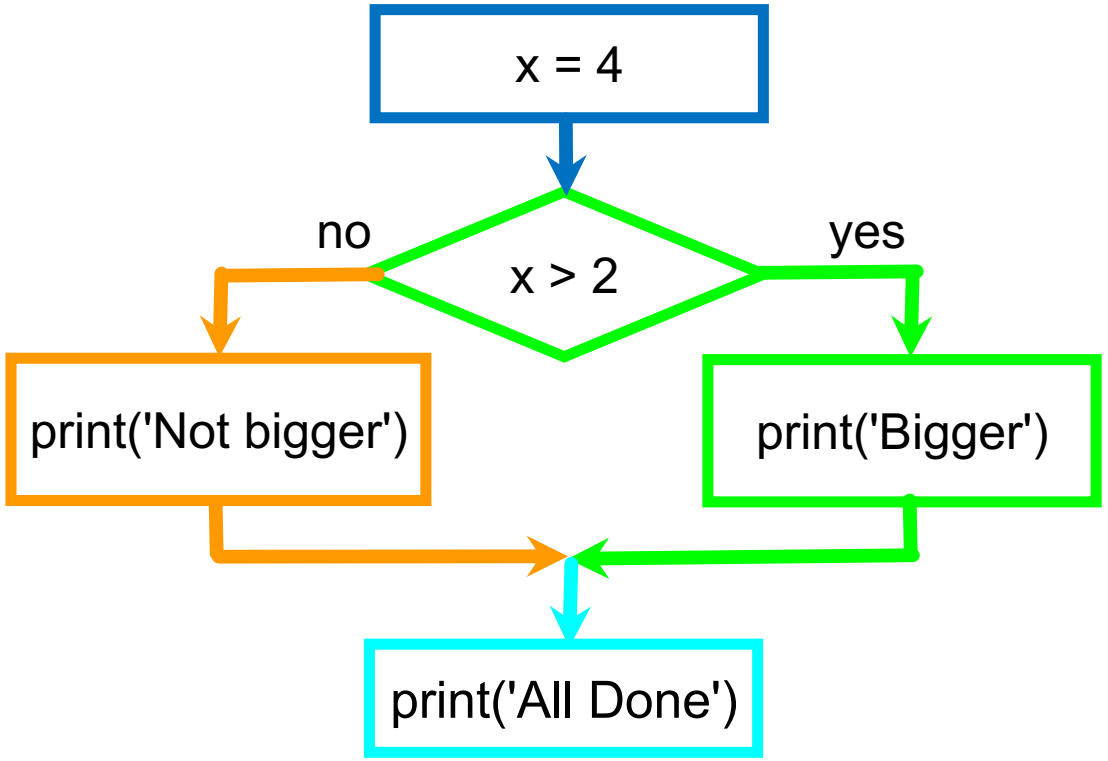
Indentazione

- È il modo per delimitare blocchi di istruzioni

```
x = 5
if x > 2 :
    print('Bigger than 2')
    print('Still bigger')
print('Done with 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Bigger than 2')
    print('Done with i', i)
print('All Done')
```

Else

```
x = 4

if x > 2 :
    print('Bigger')
else :
    print('Smaller')

print('All done')
```



Else if -> elif

```
# No Else
x = 5
if x < 2 :
    print('Small')
elif x < 10 :
    print('Medium')

print('All done')
```

```
if x < 2 :
    print('Small')
elif x < 10 :
    print('Medium')
elif x < 20 :
    print('Big')
elif x < 40 :
    print('Large')
elif x < 100:
    print('Huge')
else :
    print('Ginormous')
```

Esercizio

Riscrivi il programma per il calcolo del salario in modo da attribuire all'impiegato un salario orario pari a 1.5 volte la paga oraria normale per ogni ora di lavoro ulteriore rispetto al minimo di 40 ore.

```
Inserisci il numero di ore: 45
Inserisci la paga oraria: 10

Salario totale: 475
```

475 = 40 * 10 + 5 * 15

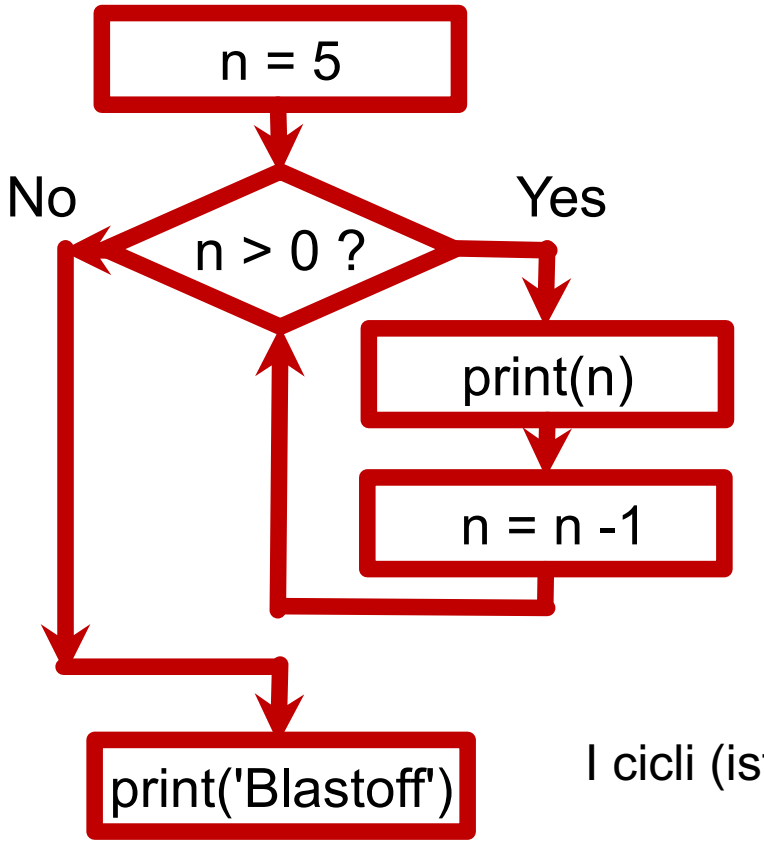
Operatori logici

- **not** A -> vera quando A è falsa e viceversa
- A **and** B -> vera quando sia A che B sono vere, falsa altrimenti
- A **or** B -> vera quando A o B sono vere, falsa quando sia A che B sono false

```
inYear = input('insert a year ')
year = int(inYear)
if (year%400==0) or (year%4==0 and year%100!=0) :
    print('it is bissextile')
else :
    print('it is not bissextile')
```

ANCORA SUI CICLI

Istruzioni ripetute



Program:

```
n = 5
while n > 0 :
    print(n)
    n = n – 1
print('Blastoff!')
```

Output:

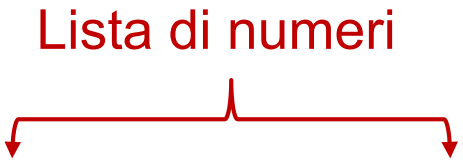
5
4
3
2
1
Blastoff!

I cicli (istruzioni ripetute) utilizzano una o più variabili che controllano l'esecuzione del ciclo.

Definite loop

- Supponiamo di avere un insieme di cose (una stringa, un insieme di numeri, …)
- Possiamo scrivere un ciclo che esegue un blocco di istruzioni per ciascun elemento dell'insieme
- Questo è chiamato "*definite loop*" perchè viene eseguito un numero esatto di volte
- Possiamo dire che i definite loop iterano sui membri di un insieme
- Il costrutto da utilizzare il Python è il for

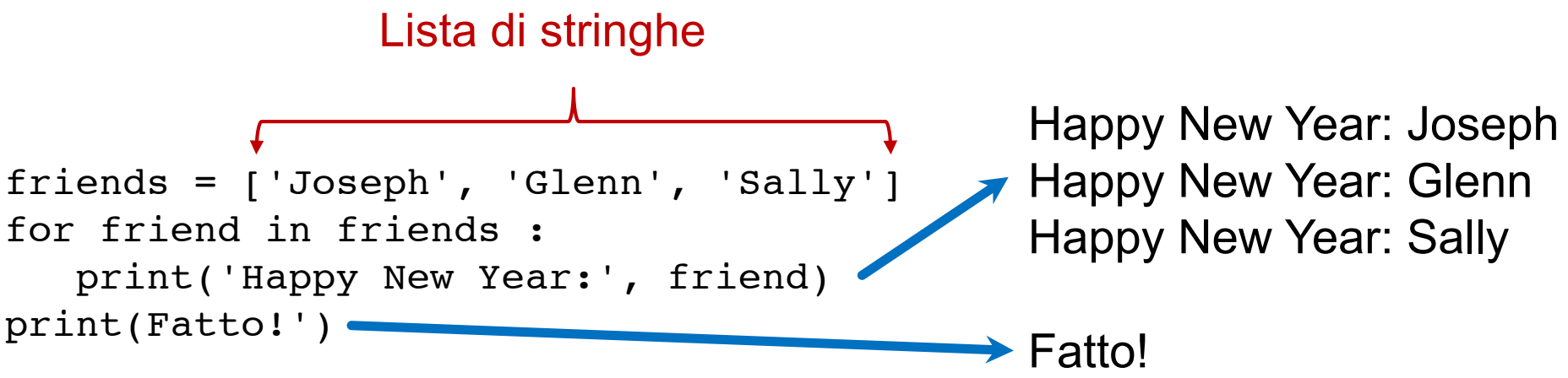
Un semplice definite Loop




```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Fine!')
```

5
4
3
2
1
Fine!

Un definite loop con le stringhe



Esercizio

Riscrivi l'esempio del calcolo del salario in modo che esegua il calcolo per 10 impiegati. Per ciascun impiegato acquisisci, come prima, il numero di ore di lavoro e il salario orario

```
Impiegato 1
Inserisci il numero di ore: 45
Inserisci la paga oraria: 10

Salario totale: 475

Impiegato 2
Inserisci il numero di ore: 32
Inserisci la paga oraria: 12

Salario totale: 384.0
```

Else come parte dei cicli

- Sia il while che il for possono terminare con un else
- L'else è eseguito quando la condizione del ciclo diventa falsa o, nei definite loop, quando non ci sono altri elementi da considerare nell'insieme di valori su cui si itera

Else al lavoro

```
while x > 0 :
    print('I am in the loop', x)
    x = x -1
else :
    print('I am in the else', x)
print('I am out of the loop')
```

Per x == 4
I am in the loop 4
I am in the loop 3
I am in the loop 2
I am in the loop 1
I am in the else 0
I am out of the loop

Per x == -1
I am in the else -1
I am out of the loop

Sommario

- Introduzione alla programmazione
- Caratteristiche generali di Python
- Variabili e costanti
- Tipi
- Conversioni tra tipi
- Operatori
- Input dall'utente
- Commenti (#)
- Istruzioni condizionali
- Istruzioni ripetute

Domande/esercizi utili per il ripasso

- Che cosa è un programma?
- Fornite due esempi di programmi che conoscete
- Perchè è utile scrivere programmi?
- Com'è organizzato un computer?
- Che differenza c'è tra un compilatore e un interprete?
- Qual è la differenza tra istruzioni sequenziali, condizionali e loop?
- Provate a pensare a che cosa è un loop infinito: sapreste costruirne uno in Python?

Domande/esercizi utili per il ripasso

- Scrivere un programma che calcola il valore assoluto di un numero
- Scrivere il programma del pangramma
- Provare a eseguirlo, guardare qui https://it.wikipedia.org/wiki/Pangramma per esempi di pangrammi
- Scoprire che cosa fa questa istruzione
  - a = [chr(ord('a')+i) for i in range(26)]
- Possiamo usarla nel programma del pangramma? Come?
- Cercare informazioni sull'ENIAC e scrivere con parole vostre, in poche righe, di che cosa si tratta
- Cercare informazioni su Cloud Computing e Big data

Acknowledgements / Contributions

Adaptation and extensions: Elisabetta Di Nitto, Politecnico di Milano