Politecnico
di Milano

Coding in Python

11-15 Giugno 2018

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

- Elisabetta Di Nitto (lezione 1, 2 e 5)
  - Via Golgi, 42
  - email elisabetta.dinitto@polimi.it
  - tel: 02-2399-3663
  - http://dinitto.faculty.polimi.it
- Alessandro Campi (lezione 3 e 4)
  - Via Ponzio 34/5
  - email alessando.campi@polimi.it
  - tel: 02-2399-3644
- Tutor
  - Gianmarco Loliva (mattina e pomeriggio)
    - email gianmarco.loliva@mail.polimi.it
  - Michele Lambertucci (pomeriggio)
    - email michele.lambertucci@mail.polimi.it
- Aspetti logistici e amministrativi
  - Laura Brambilla
    - Via Ponzio 34/5
    - email laura.brambilla@polimi.it
    - tel: 02-2399-3427

Organizzazione del laboratorio

- Mattina dalle 9.00 alle 12.00
  - Lezioni
- Pomeriggio dalle 13.00 alle 16.00
  - Studio autonomo e svolgimento di esercizi
- Aula G.0.2

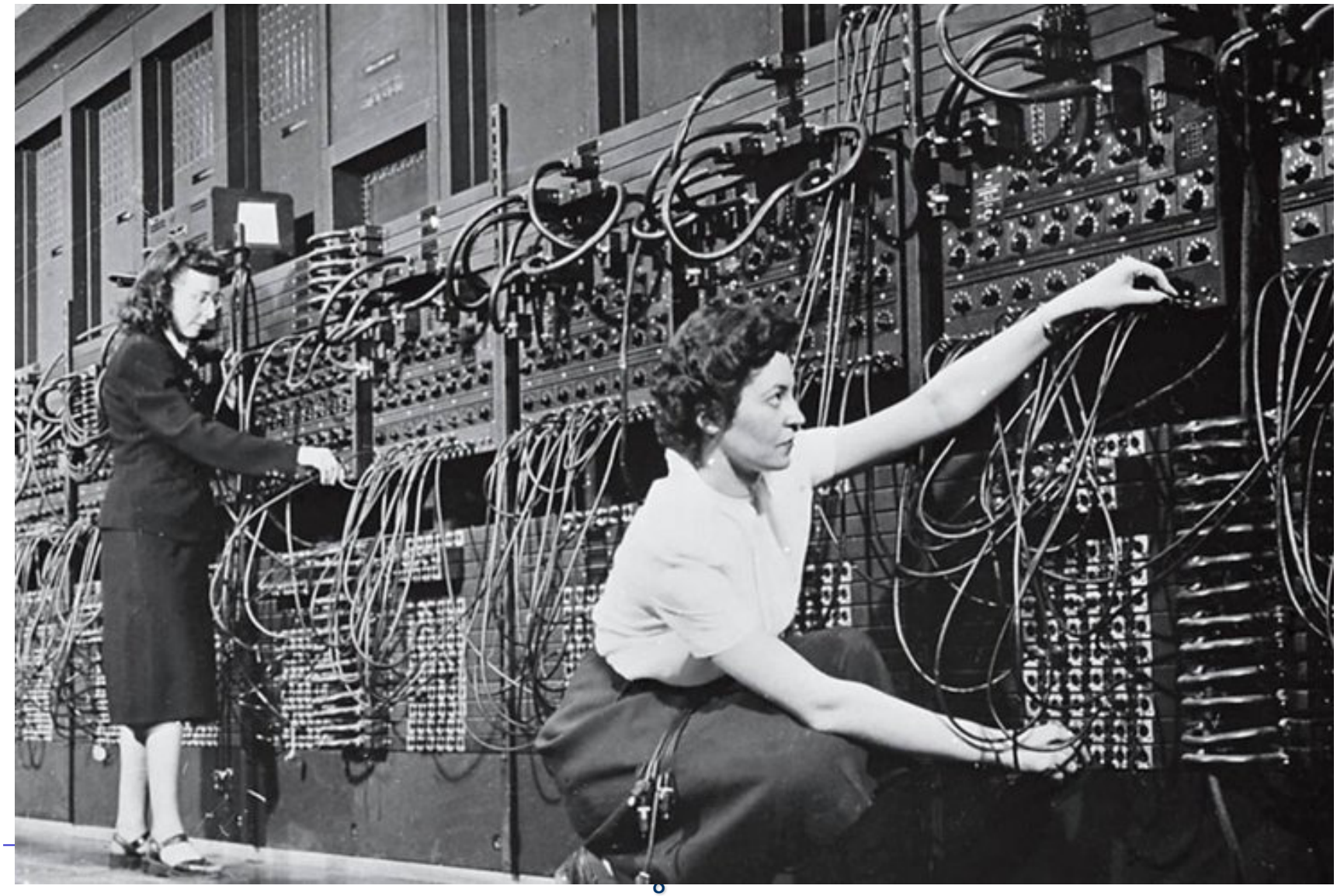
Materiale

- 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…

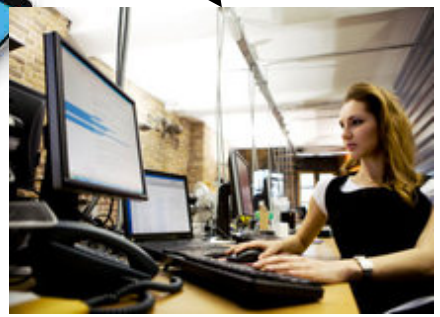


…a macchine programmabili un po' più maneggevoli…

...a???

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

Utenti, computer, reti, programmatori

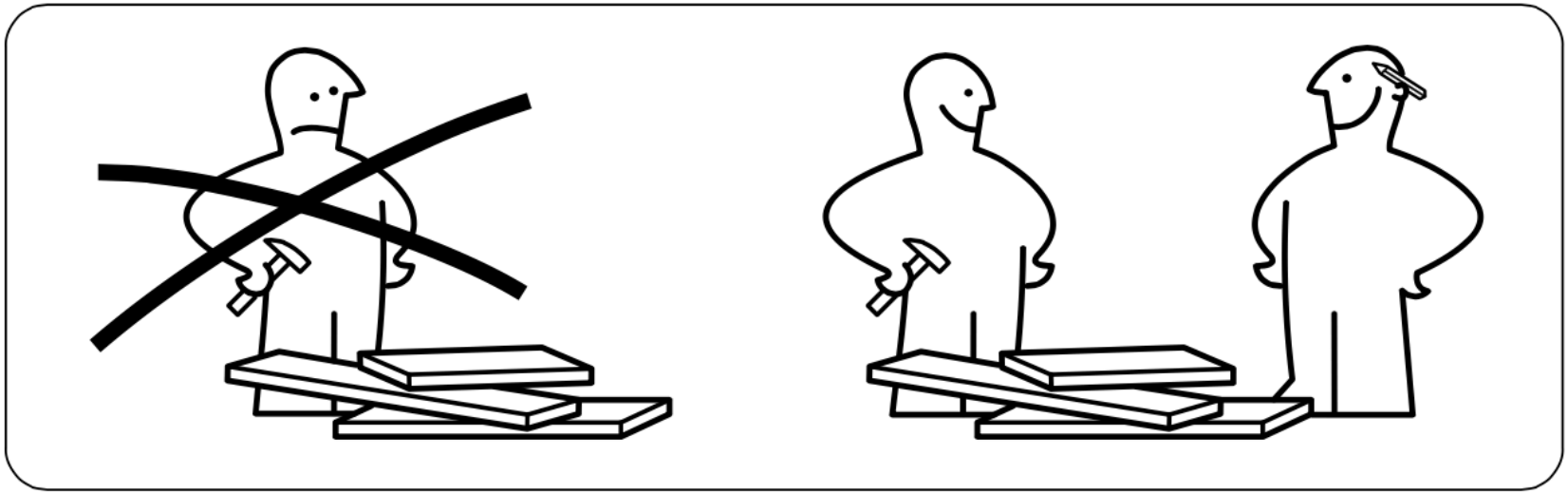


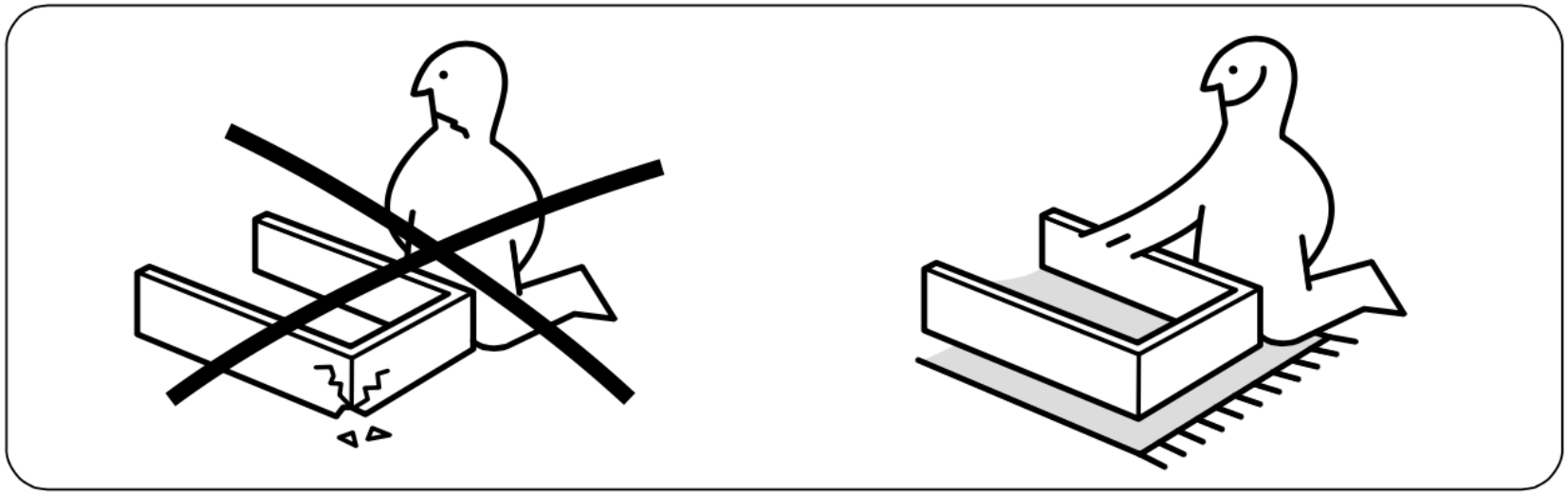
- I programmatori trasformano i dati in informazioni, secondo quanto necessario agli utenti
- Devono tenere conto di tanti vincoli
- I computer e la rete sono i loro alleati

Software, codice e programmi

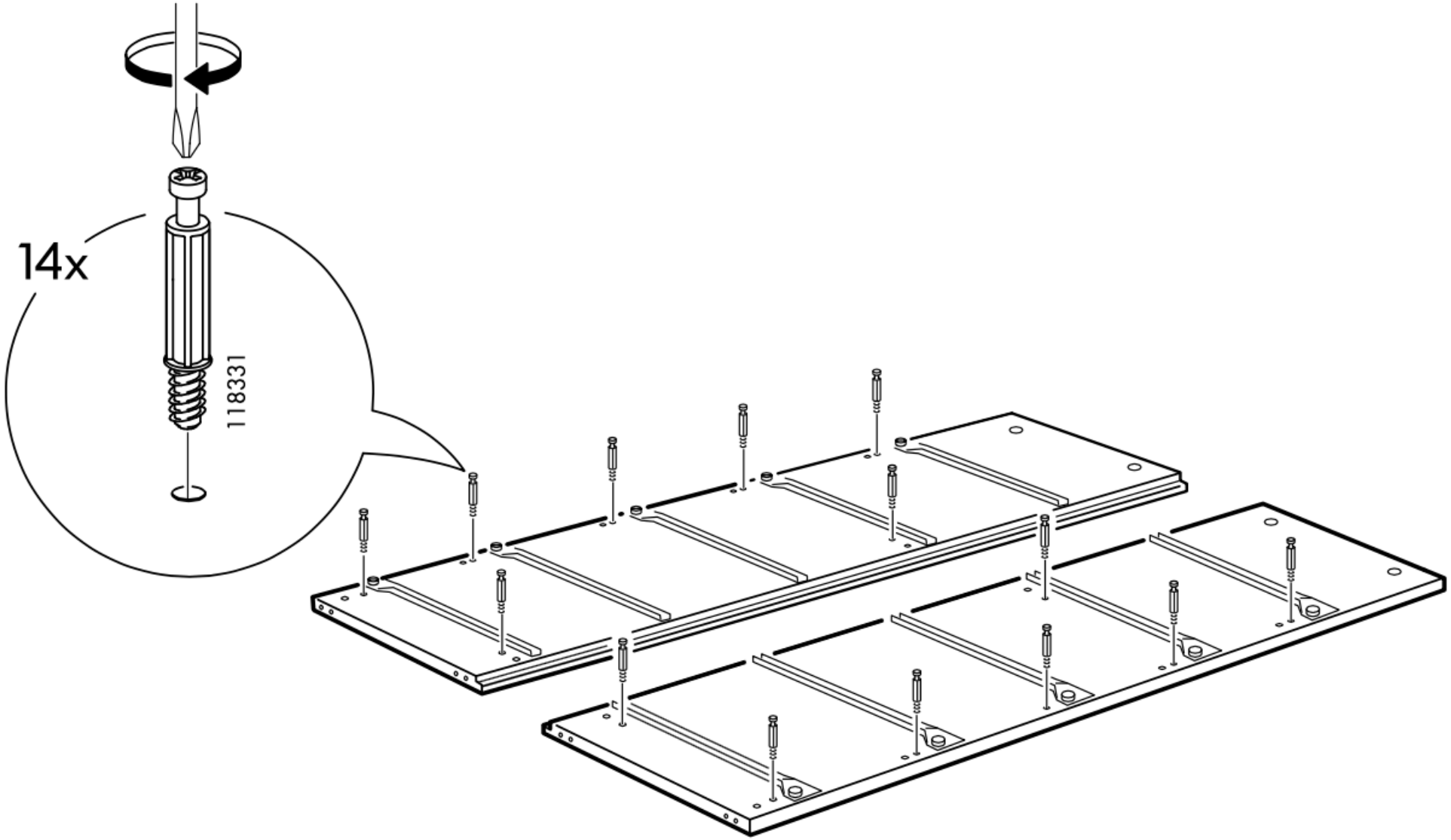
- Una sequenza di istruzioni
  - Può essere usato più volte per svolgere un certo compito
  - Può essere un pezzo d'arte creativa!

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 umani

https://www.youtube.com/watch?v=SWLI5ejQY5c

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')
```

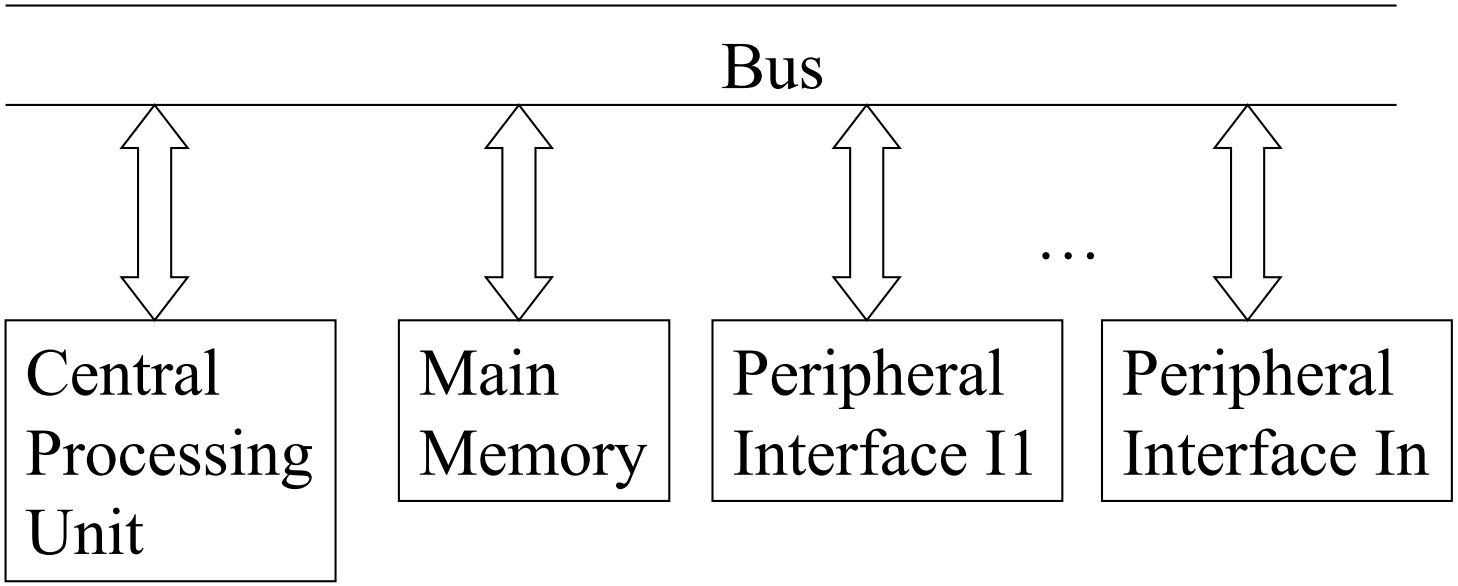
Programmi, esecutori e linguaggi di programmazione

- 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)

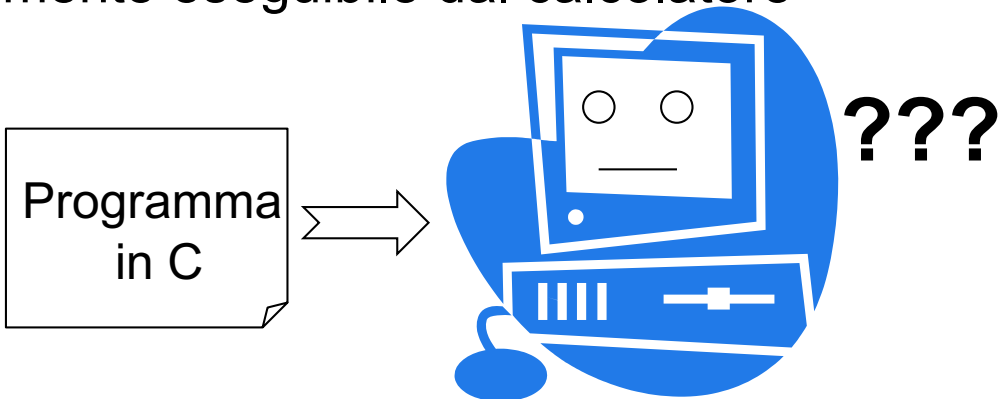
0000	0100000000001000	Acquisisci il primo numero e conservalo in 1000
0001	0100000000001001	Acquisisci il secondo numero e conservalo in 1001
0010	0000000000001000	Carica in A il numero conservato in 1000
0011	0001000000001001	Carica in B il numero conservato in 1001
0100	0110000000000000	Effettua la 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

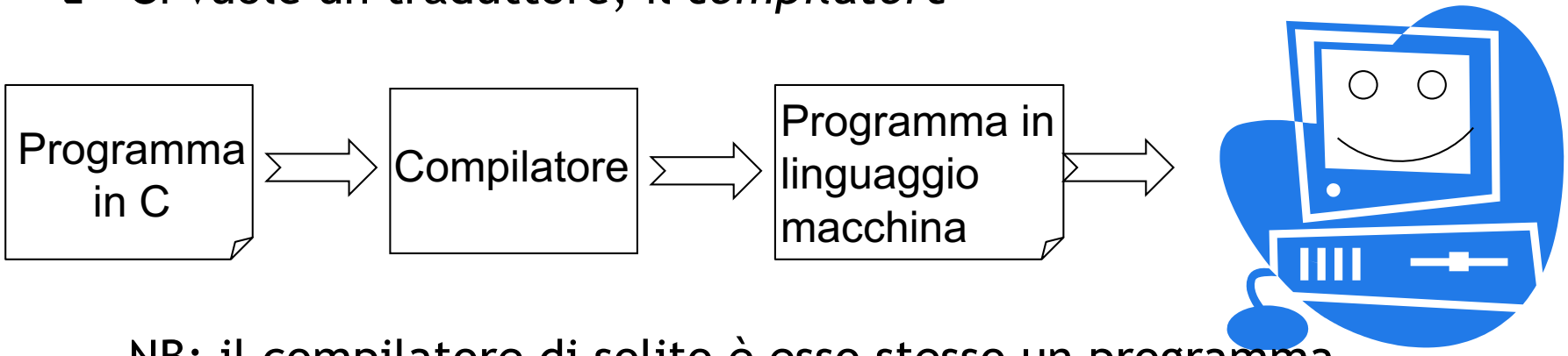
- Il linguaggio macchina è poco espressivo
  - Un compito semplice deve essere decomposto in più istruzioni macchina
- I linguaggi di programmazione di “alto livello” offrono una maggiore espressività e semplicità
  - Python, C, Java, C++, MATLAB…

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



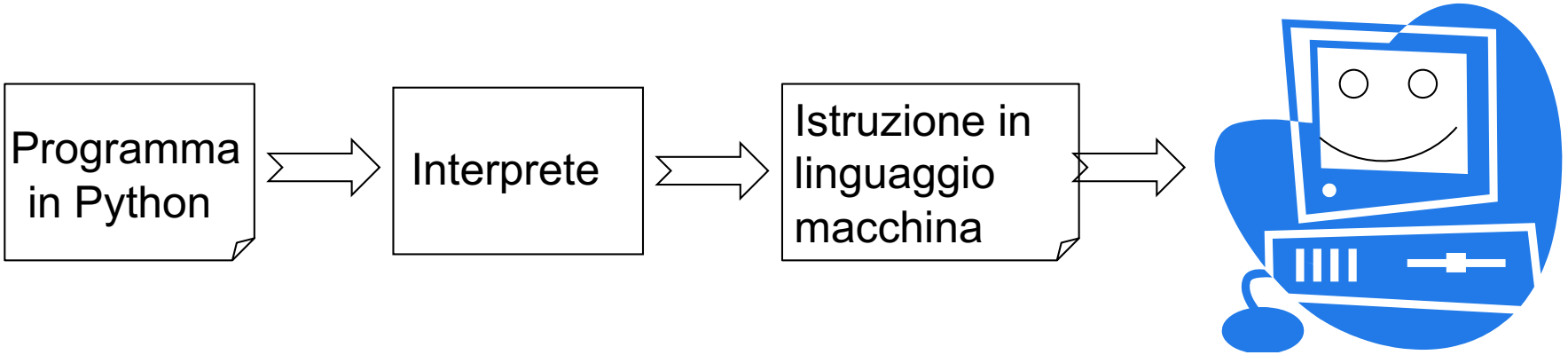
- Ci vuole un traduttore, il *compilatore*



- 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?

- Seconda possibilità: non operiamo una trasformazione a priori prima dell'esecuzione, usiamo i servizi di un interprete



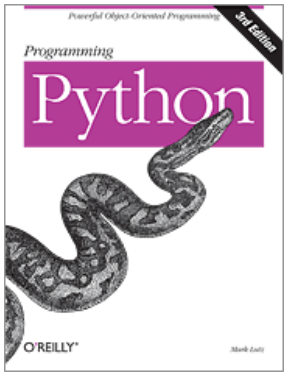
- L'interprete legge e traduce al volo durante l'esecuzione

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 ← Useremo questo

Elementi di Python

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

Parole riservate

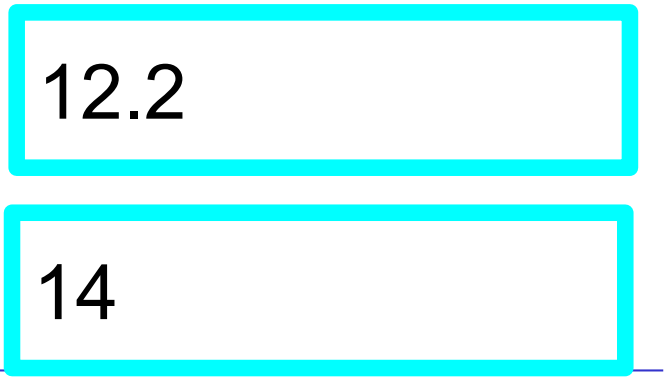
- Parole che hanno un significato specifico

```
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

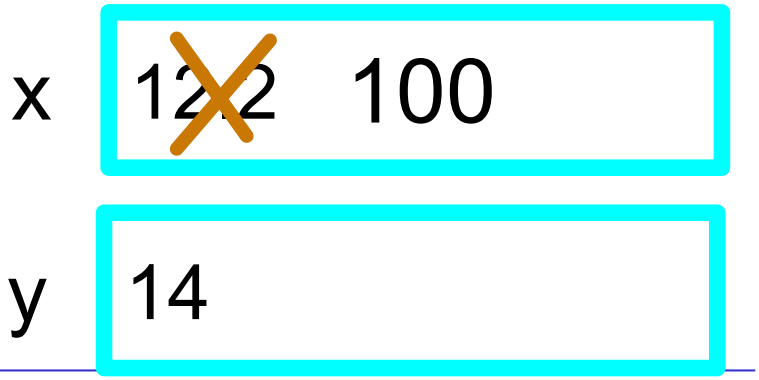
```
x = 12.2
y = 14
```



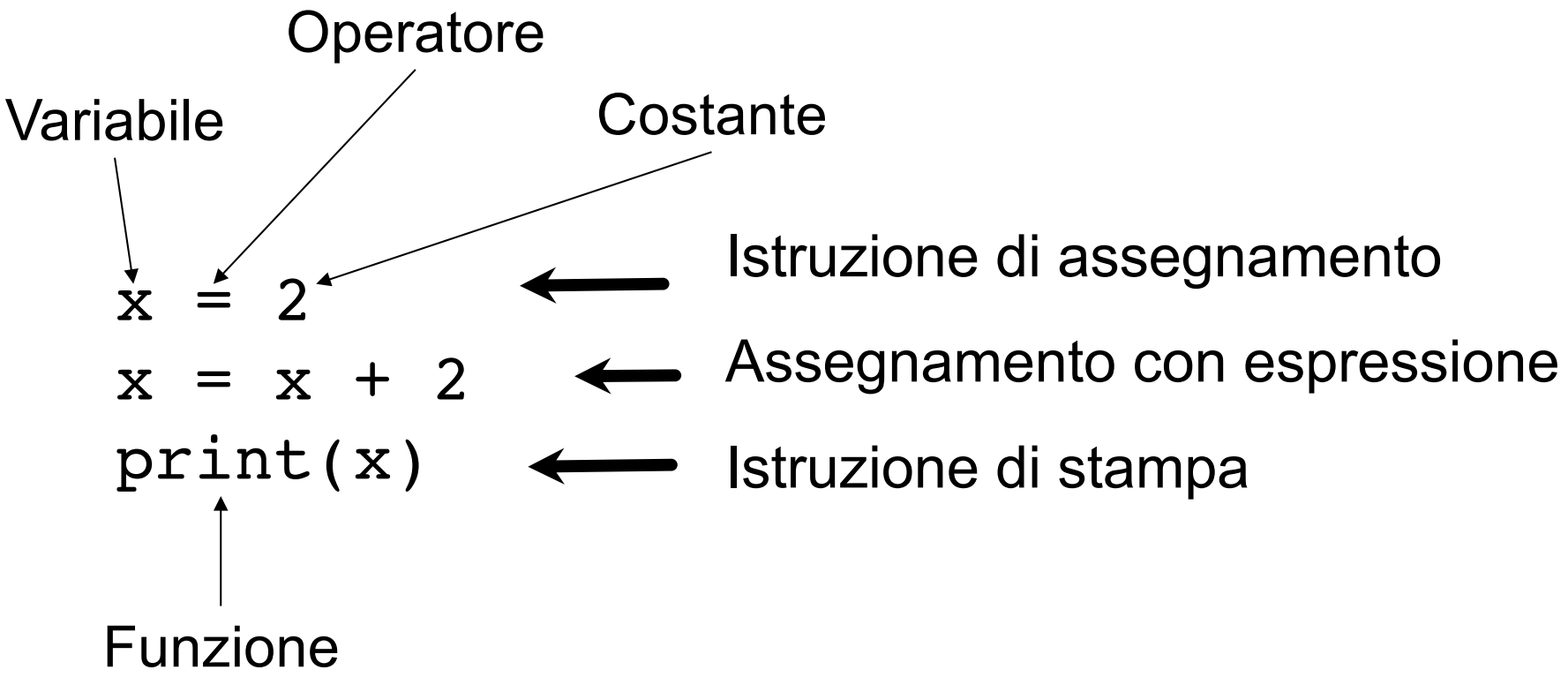
Variabili

- 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

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



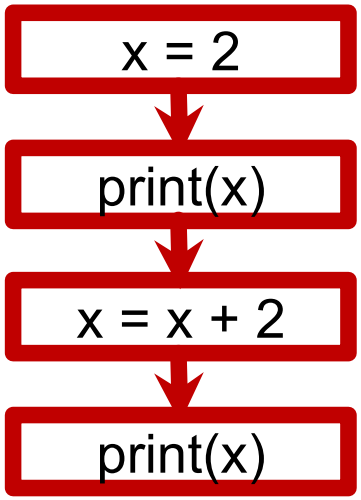
Frasi


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

Program Steps or Program Flow

- Like a recipe or installation instructions, a program is a **sequence** of steps to be done in order.
- Some steps are **conditional** - they may be skipped.
- Sometimes a step or group of steps is to be **repeated**.
- Sometimes we store a set of steps to be used over and over as needed several places throughout the program (these are **functions**).

Sequential Steps



Program:

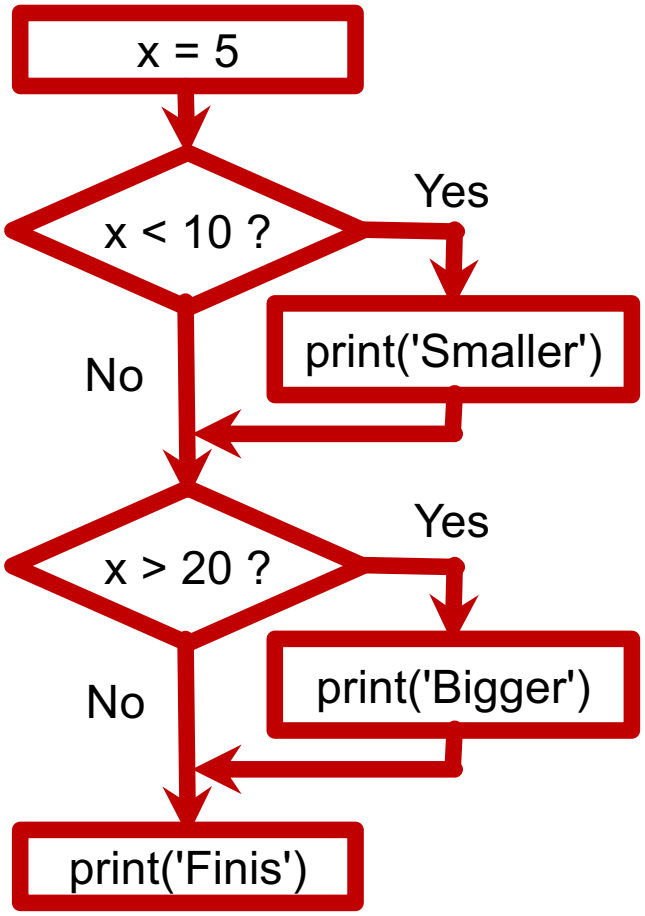
```
x = 2
print(x)
x = x + 2
print(x)
```

Output:

2
4

When a program is running, it flows from one step to the next. As programmers, we set up “paths” for the program to follow.

Conditional Steps



Program:

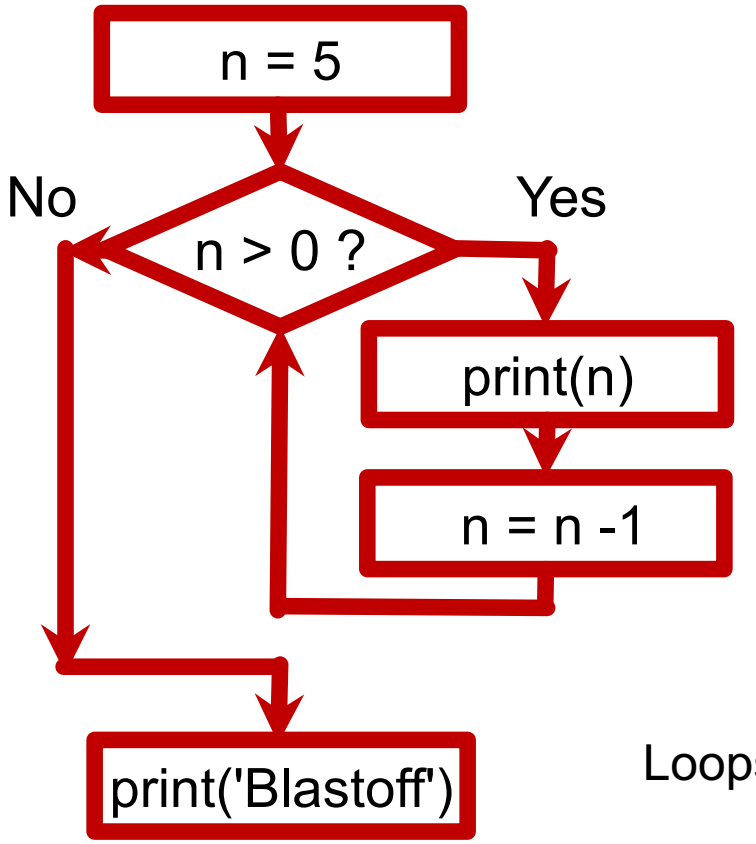
```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Finis')
```

Output:

```
Smaller
Finis
```

Repeated Steps



Program:

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

Output:

5
4
3
2
1
Blastoff!

Loops (repeated steps) have iteration variables that change each time through a loop.

Classifying the instructions of our first example…


```
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')
```

Sequential

Repeated

Conditional

Words, sentences and paragraphs


```
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')
```

A short Python “Story” about how identifying pangrams

A word used to read data from a user

A sentence about considering all alphabet letters

A paragraph about informing the usage of the program outcome

Interactive versus Script

- Interactive
  - We type directly to Python one line at a time and it responds
- Script
  - We enter a sequence of statements (lines) into a file using a text editor and tell Python to execute the statements in the file
  - As a convention, we add ".py" as the suffix on the end of these files to indicate they contain Python.

VARIABLES AND CONSTANTS

Constants

- Fixed values such as numbers, letters, and strings, are called “constants” because their value does not change
- Numeric constants are as you expect
- String constants use single quotes (')
  or double quotes (")

```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Hello world')
Hello world
```

Numeric Expressions


```
>>> 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
```

```
      4 R 3
    ------
5 | 23
    20
    --
     3
```

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

What Does “Type” Mean?

- In Python variables, literals, and constants have a “type”
- Python knows the difference between an integer number and a string
- For example “+” means “addition” if something is a number and “concatenate” if something is a string

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

concatenate = put together

Type Matters

- Python knows what “type” everything is
- Some operations are prohibited
- You cannot “add 1” to a string
- We can ask Python what type something is by using the type() function

```
>>> 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'>
>>>
```

Several Types of Numbers

- Numbers have two main types
  - Integers are whole numbers:
    - -14, -2, 0, 1, 100, 401233
  - Floating Point Numbers have decimal parts:
    - -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'>
>>>
```

Integer Division

- Integer division produces a floating point result

```
>>> 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
```

User Input

- We can instruct Python to pause and read data from the user using the input() function
- The input() function returns a string

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

Who are you? Chuck
Welcome Chuck

Converting User Input

- If we want to read a number from the user, we must convert it from a string to a number using a type conversion function
- Later we will deal with bad input data

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

Europe floor? 0
US floor 1

Comments in Python

- Anything after a # is ignored by Python but very useful for programmers and those willing to understand a piece of code
- Why comment?
  - Describe what is going to happen in a sequence of code
  - Document who wrote the code or other ancillary information
  - Turn off a line of code - perhaps temporarily

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

Exercise

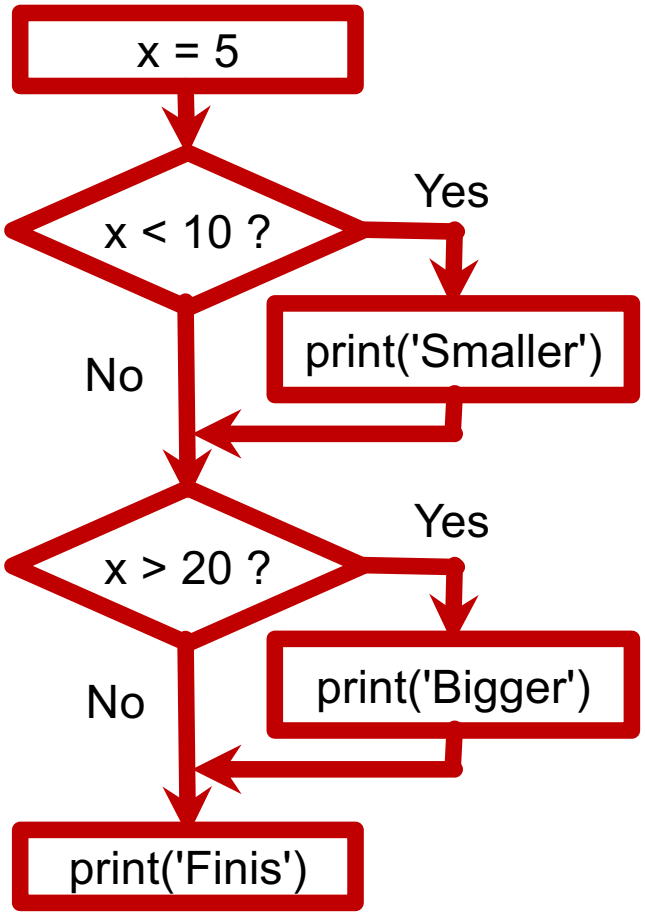
Write a program to prompt the user for hours and rate per hour to compute gross pay.

```
Enter Hours: 35
Enter Rate: 2.75

Pay: 96.25
```

MORE ON CONDITIONAL STEPS

Conditional Steps



Program:

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

print('Finis')
```

Output:

```
Smaller
Finis
```

Comparison Operators

- Boolean expressions using comparison operators evaluate to True / False or Yes / No
- Comparison operators look at variables but do not change the variables

Python	Meaning
<	Less than
<=	Less than or Equal to
==	Equal to
>=	Greater than or Equal to
>	Greater than
!=	Not equal

Remember: "=" is used for assignment.

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

Indentation

- It is the way to delimit a block of instructions

```
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')
```

Indentation

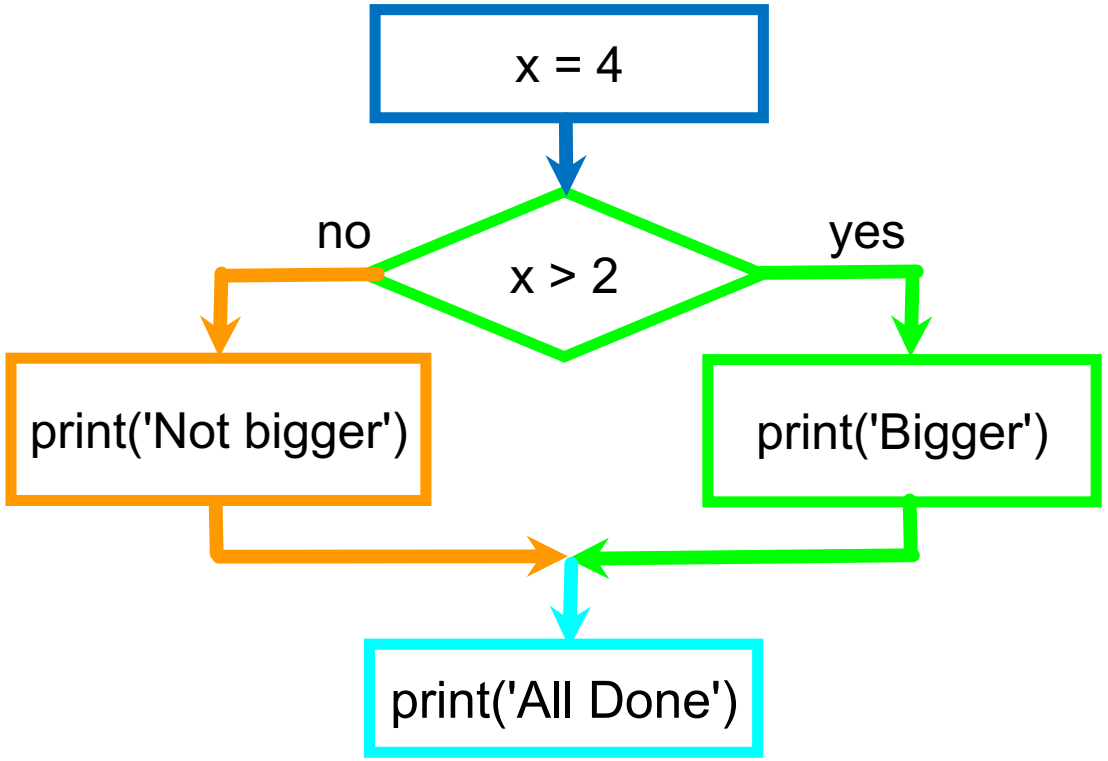
- Increase indent indent after an if statement or for statement (after :)
- Maintain indent to indicate the scope of the block (which lines are affected by the if/for)
- Reduce indent back to the level of the if statement or for statement to indicate the end of the block
- Blank lines are ignored - they do not affect indentation
- Comments on a line by themselves are ignored with regard to indentation

Two-way Decisions with else:

```
x = 4

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

print('All done')
```



Multi-way

```
# 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')
```

Exercise

Rewrite your pay computation to give the employee 1.5 times the hourly rate for hours worked above 40 hours.

```
Enter Hours: 45
Enter Rate: 10

Pay: 475.0
```

475 = 40 * 10 + 5 * 15

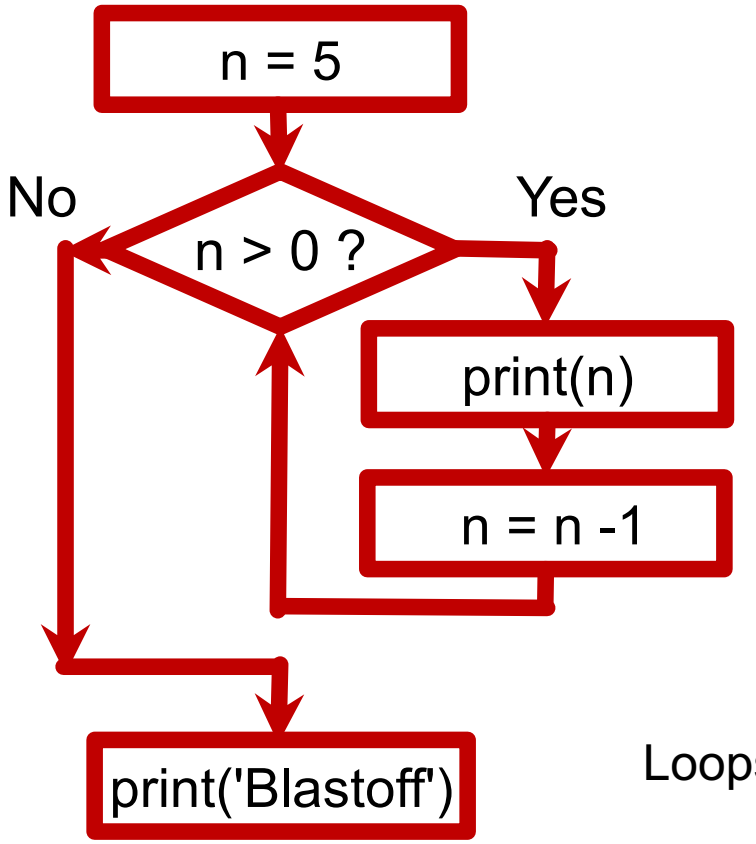
Logical operators

- **not** A -> true when A is false and vice versa
- A **and** B -> true when both A and B are true, false otherwise
- A **or** B -> true when either A or B are true, false when both A and B are 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')
```

MORE ON REPEATED STEPS (LOOPS)

Repeated Steps



Program:

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

Output:

5
4
3
2
1
Blastoff!

Loops (repeated steps) have iteration variables that change each time through a loop.

Definite Loops

- When we have a finite set of things
- We can write a loop that executes a block for each of the items in the set using the Python for construct
- These are called "*definite loops*" because they execute an exact number of times
- We say that "definite loops iterate through the members of a set"

A Simple Definite Loop


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

5
4
3
2
1
Blastoff!

A Definite Loop with Strings

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
   print('Happy New Year:', friend)
print('Done!')
```

Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally

Done!

Exercise

Rewrite your pay computation to compute the salary of 10 employees. For each employee, acquire the hours and the rate as before

```
Employee 1
Enter Hours: 45
Enter Rate: 10

Pay: 475.0

Employee 2
Enter Hours: 32
Enter Rate: 12

Pay: 384.0
```

Else as part of loops

- Both while and for can terminate with an else statement
- The else is executed when the condition in the loop becomes false

Else at work

```
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')
```

If 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

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

Summary

- Introduction to programming
- Python overview
- Variables and constants
- Types
- Operators
- Conversion between types
- User input
- Comments (#)
- Conditional steps
- Repeated steps

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