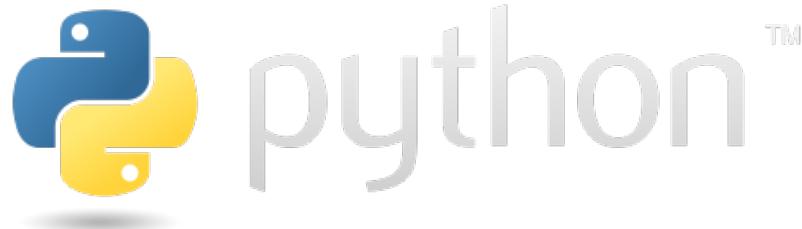




# LEZIONE 1

28 Maggio 2018

## Python



Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

As of March 2018, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 130,000 packages with a wide range of functionality, including:

- Graphical user interfaces
- Web frameworks
- Multimedia
- Databases
- Networking
- Test frameworks
- Automation
- Web scraping
- Documentation
- System administration
- Scientific computing
- Text processing
- Image processing

## Anaconda



Anaconda is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS.

<https://www.anaconda.com/download/> (<https://www.anaconda.com/download/>)

## Jupyter



The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

<http://jupyter.org/> (<http://jupyter.org/>)

## Il linguaggio Python

No. Il nome deriva da “Monty Python's Flying Circus” (gruppo di comici inglese) Show BBC ma anche film: ricordiamo tra gli altri "Brian di Nazareth", "Il Senso della Vita"...

Guido van Rossum (padre di Python) è un fan..

Sito ufficiale del linguaggio: interprete linux / win / MacOS (ultima versione 2.5), IDE per Win, tutorial, reference: <http://www.python.org> (<http://www.python.org>)

“Dive into Python” (free book molto completo)

<http://diveintopython.org/index.html> (<http://diveintopython.org/index.html>)

“How to Think Like a Computer Scientist with Python” (più introduttivo)

<http://greenteapress.com/thinkpython> (<http://greenteapress.com/thinkpython>)

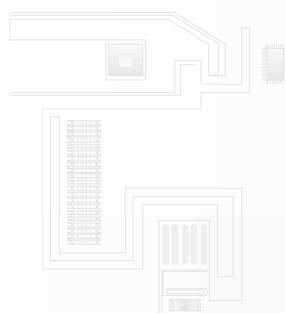
```
In [20]: from IPython.display import IFrame  
IFrame("Python.pdf", width=900, height=800)
```

Out[20]:



# Soluzione

```
def add(a, b):  
    return a+b  
  
def squareBySum(n):  
    return accumulate(add, map(ith0d
```



## Python

Python, a differenza di C/C++, è interpretato (anche se poi molte implementazioni lo compilano per motivi di efficienza): si può interagire con una macchina virtuale Python in maniera interattiva oppure come nel nostro caso attraverso un notebook Jupyter.

```
In [21]: #sono un commento  
2+2
```

```
Out[21]: 4
```

```
In [22]: 10/3 #divisione intera
```

```
Out[22]: 3
```

```
In [3]: pippo = 7  
pippo = 'ciao'  
pippo
```

```
Out[3]: 'ciao'
```

```
In [4]: a = b = c = 1.1
```

```
In [5]: b
```

```
Out[5]: 1.1
```

## Stringhe

```
In [23]: "sono una stringa"
```

```
Out[23]: 'sono una stringa'
```

```
In [24]: 'sono una stringa'
```

```
Out[24]: 'sono una stringa'
```

```
In [27]: stringa = 'ciao'  
stringa + ' mondo'
```

```
Out[27]: 'ciao mondo'
```

```
In [28]: stringa*4
```

```
Out[28]: 'ciaociaociaociao'
```

## Stringhe

```
In [11]: "Questa lezione mi sta annoiando parecchio"[2:-2] #uno slice!
```

```
Out[11]: 'esta lezione mi sta annoiando parecch'
```

```
In [29]: casa = "voglio andare a casa" #le stringhe non sono modificabili!
casa[3]='A'
```

```
-----
-----
TypeError                                     Traceback (most rec
ent call last)
<ipython-input-29-8f71e5412286> in <module>()
      1 casa = "voglio andare a casa" #le stringhe non sono mo
dificabili!
----> 2 casa[3]='A'
```

```
TypeError: 'str' object does not support item assignment
```

```
In [30]: casetta = casa[0:2] + 'A' + casa[4:]
casetta
```

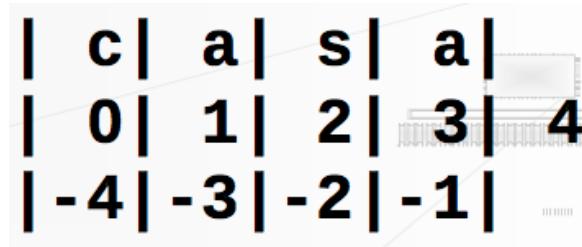
```
Out[30]: 'voAio andare a casa'
```

```
In [31]: casa, casett = casetta, casa
```

```
In [32]: casa
```

```
Out[32]: 'voAio andare a casa'
```

## Stringhe



```
In [33]: parola = 'casa'
parola[:2]
```

```
Out[33]: 'ca'
```

```
In [35]: parola[2:]
```

```
Out[35]: 'sa'
```

```
In [36]: parola[:] #è una copia!
```

```
Out[36]: 'casa'
```

```
In [38]: len(parola)
```

```
Out[38]: 4
```

## Liste

- Fondamentali e molto usate in Python
- Sono sequenze, come le stringhe (ergo: indici, slice, len)
- Però sono modificabili
- Sono la generalizzazione degli array in altri linguaggi

```
In [40]: lista = ['ciao', 3, 4.3234, "pippo"]  
lista
```

```
Out[40]: ['ciao', 3, 4.3234, 'pippo']
```

```
In [41]: len(lista)
```

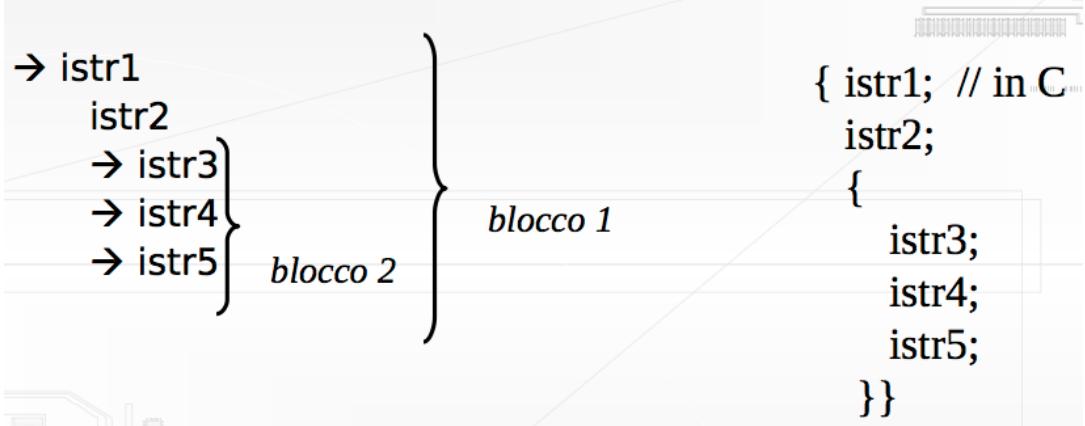
```
Out[41]: 4
```

```
In [42]: lista.append(8)  
lista
```

```
Out[42]: ['ciao', 3, 4.3234, 'pippo', 8]
```

## Ciclo while (e come si delimitano i blocchi di istruzioni in Python)

In Python, a differenza della stragrande maggioranza degli altri linguaggi, il corpo del while (un blocco in generale) è delimitato per mezzo della indentazione! (posso usare spazi o tab, basta che siano lo stesso numero)



```

In [43]: a, b = 0, 1      # assegnamento con tupla (detto multiplo) !
while b < 10:
    print b, # "," serve ad evitare \n finale
    a, b = b, a+b

```

1 1 2 3 5 8

## IF

La forma più generale:

```

In [ ]: if cond1 :
            # cond1 vera
        elif cond2 :
            # cond2 vera
        elif cond3 :
            # cond3 vera
        else:
            # nemmeno una vera!

```

## FOR

Il for, a differenza del C, itera su sequenze (es. stringhe o liste)

In pratica:

for i in seq:

fai qualcosa con i

```
In [49]: for i in ['Ma questa', 'è', 1, 'lista?'] :
    print i,
```

Ma questa è 1 lista?

```
In [51]: for i in range(0,6):
    print i,
```

0 1 2 3 4 5

## Costruttori di liste

Assomigliano molto alla notazione insiemistica, per es. un analogo di  $\{(x,y) \mid x \in A, y \in B, x \neq y\}$  si può scrivere come:

```
In [53]: A = [1,2,3]
B = ['a', 'b', 'c']
[(x,y) for x in A for y in B if x != y]
```

```
Out[53]: [(1, 'a'),
           (1, 'b'),
           (1, 'c'),
           (2, 'a'),
           (2, 'b'),
           (2, 'c'),
           (3, 'a'),
           (3, 'b'),
           (3, 'c')]
```

A proposito, (x,y) è un esempio di tupla (come in matematica) – in Python è una sequenza immutabile e si può usare come alternativa più efficiente alle liste, se non si devono modificare dati Attenzione! E' comunque una lista, non un insieme (è un insieme totalmente ordinato che ammette più occorrenze dello stesso elemento)...

## Le funzioni

```
In [54]: def f(n) :  
    """Be' se proprio voglio qui ci metto la stringa di  
    documentazione"""  
    if n == 0:  
        return 1  
    else:  
        return n*f(n-1)
```

```
In [58]: f(4)
```

```
Out[58]: 24
```

```
In [63]: def incipitizza(seq, incipit = 'banale'): #argomenti di default  
    seq[0] = incipit  
    return seq  
  
a = [1,2,3]  
incipitizza(a)
```

```
Out[63]: ['banale', 2, 3]
```

```
In [64]: incipitizza(a,1)
```

```
Out[64]: [1, 2, 3]
```

## Argomenti con parole chiave

Sono molto usati!

```
In [66]: def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
    print "-- This parrot wouldn't", action,
    print "if you put", voltage, "Volts through it."
    print "-- Lovely plumage, the", type
    print "-- It's", state, "!"
```

```
In [67]: parrot(1000)
```

```
-- This parrot wouldn't voom if you put 1000 Volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's a stiff !
```

```
In [68]: parrot(action = 'VOOOOOM', voltage = 1000000)
```

```
-- This parrot wouldn't VOOOOOM if you put 1000000 Volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's a stiff !
```

```
In [69]: parrot('a thousand', state = 'pushing up the daisies')
```

```
-- This parrot wouldn't voom if you put a thousand Volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's pushing up the daisies !
```

## Dizionari

- Sono anche chiamati memorie associative o array associativi
- A differenza delle sequenze, gli indici non sono interi bensì chiavi (es. stringhe)
- Sintassi: {chiave1 : val1, chiave2 : val2, ...}
- il metodo keys() restituisce la lista delle chiavi di un dizionario

```
In [71]: tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
tel
```

```
Out[71]: {'guido': 4127, 'jack': 4098, 'sape': 4139}
```

```
In [72]: tel['jack']
```

```
Out[72]: 4098
```

```
In [73]: del tel['sape']
tel['irv'] = 4127
tel
```

```
Out[73]: {'guido': 4127, 'irv': 4127, 'jack': 4098}
```

```
In [74]: tel.keys()
```

```
Out[74]: ['jack', 'irv', 'guido']
```

```
In [75]: tel.has_key('guido')
```

```
Out[75]: True
```

## dir()

dir() applicato a qualcosa mi dice quali nomi sono definiti in questo qualcosa (un po' vago...)

```
In [77]: a = [1,2,3]
for i in dir(a):
    print i,
```

```
__add__ __class__ __contains__ __delattr__ __delitem__ __delsl
ice__ __doc__ __eq__ __format__ __ge__ __getattribute__ __geti
tem__ __getslice__ __gt__ __hash__ __iadd__ __imul__ __init__
__iter__ __le__ __len__ __lt__ __mul__ __ne__ __new__ __reduce
__ __reduce_ex__ __repr__ __reversed__ __rmul__ __setattr__ __
setitem__ __setslice__ __sizeof__ __str__ __subclasshook__ app
end count extend index insert pop remove reverse sort
```

NB: la notazione **qualcosa** (dove \_\_ sono due caratteri di underscore) è abbastanza classica in Python: sono metodi ed attributi privati.

## Python type system

- E' un linguaggio tipizzato
- Non fa static type checking (ma fa dynamic type checking: gli errori di tipo vengono rilevati, ma a runtime)
- Il tipo delle variabili non e' dichiarato

```
In [78]: x = 0 # x bound to an integer object  
x = "Hello" # now it's a string  
x = [1, 2, 3] # and now it's a list
```

```
In [79]: x
```

```
Out[79]: [1, 2, 3]
```

Le funzioni sono first class object: non c'è nessuna differenza tra variabili che contengono valori e quelle che contengono funzioni (callable/non callable)

```
In [80]: def f():  
    return 66  
  
x = f  
x()
```

```
Out[80]: 66
```

Posso anche passare le funzioni come parametri

```
In [82]: def f(x):  
    x()  
  
def a():  
    print "Io sono a"  
  
def b():  
    print "Io sono b"  
  
f(a)  
f(b)
```

```
Io sono a  
Io sono b
```

## Dynamic Type Checking

Ottengo un errore di tipo quando python non trova l'attributo a cui sto accedendo cioè:

- invoco un metodo non definito dell'oggetto
- leggo un campo non definito dell'oggetto

```
In [84]: class libro(object):
    def __init__(self,contenuto):
        self.contenuto = contenuto
    contenuto = "Nel mezzo del cammin di nostra vita"
    def read(self):
        return self.contenuto
    def stampaContenuto(l):
        print l.read()

x = libro("Nel mezzo del cammin di nostra vita")
y = "Questo non e' un libro"

import random

if random.random() < 0.5:
    stampaContenuto(y)
else:
    stampaContenuto(x)
```

```
-----
-----
AttributeError                                Traceback (most rec
ent call last)
<ipython-input-84-670a6ace8c88> in <module>()
      14
      15 if random.random() < 0.5:
---> 16     stampaContenuto(y)
      17 else:
      18     stampaContenuto(x)

<ipython-input-84-670a6ace8c88> in stampaContenuto(l)
      6         return self.contenuto
      7 def stampaContenuto(l):
----> 8     print l.read()
      9
     10 x = libro("Nel mezzo del cammin di nostra vita")

AttributeError: 'str' object has no attribute 'read'
```

## Dynamic Type Checking, però:

```
In [87]: class P(object):
    valore = 5

    x = P()
    x.valoree = 10
    x.valoree
    10
```

```
Out[87]: 10
```

Nessuna eccezione!! => in python se provo assegnare (binding) un attributo che non esiste python lo crea al momento!!

Quindi in questo momento la variabile x è una reference a un istanza di P con IN PIU' un attributo "valoree"

```
In [88]: type(x)
```

```
Out[88]: __main__.P
```

```
In [90]: for i in dir(x):
    print i,
```

```
__class__ __delattr__ __dict__ __doc__ __format__ __getattribu
te__ __hash__ __init__ __module__ __new__ __reduce__ __reduce_
ex__ __repr__ __setattr__ __sizeof__ __str__ __subclasshook__
__weakref__ valore valoree
```

Risoluzione degli attributi (es x.attr)

- prima cerco nell'istanza x
- poi cerco nella classe di x
- infine cerco nelle classi padre (ereditarietà)

## Assegnamento

- In Python l'accesso agli oggetti avviene tramite reference (analogo di quanto avviene in JAVA con le classi)
- Non esiste dichiarazione delle varibili: vengono istanziate quando vi si assegna un valore per la prima volta
- Non si può utilizzare una variabile prima che sia stata inizializzata

Quando si esegue un assegnamento in realtà viene copiata la reference non l'oggetto:

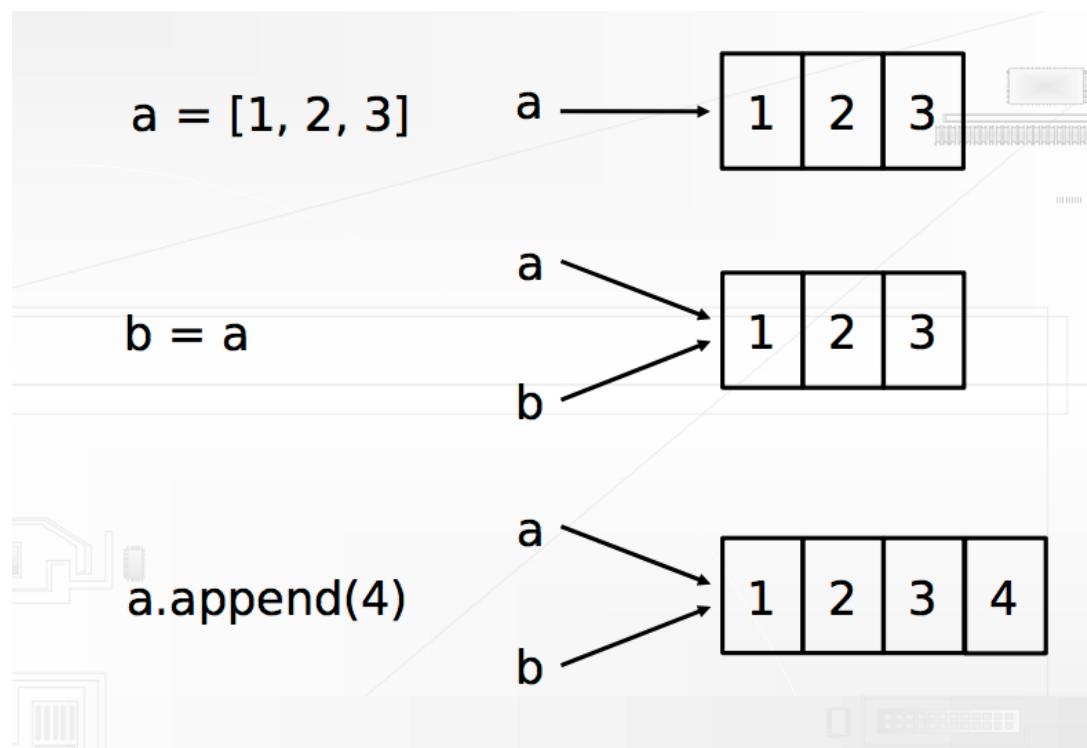
```
In [92]: a = [1,2,3]
b = a
id(a) # id(var) restituisce l'indirizzo (l-value) di var
```

```
Out[92]: 4373021184
```

```
In [94]: id(b)
```

```
Out[94]: 4373021184
```

Si crea un alias: modificando a modifco anche b!



In Python gli oggetti si dividono in:

- Oggetti mutabili il cui valore può essere modificato (liste, dizionari, classi)

```
In [96]: a = [1,2,3,4]
id(a)
```

```
Out[96]: 4373022264
```

```
In [97]: a[0] = 1
id(a)
```

```
Out[97]: 4373022264
```

- Oggetti immutabili il cui valore non può essere modificato senza creare un nuovo oggetto

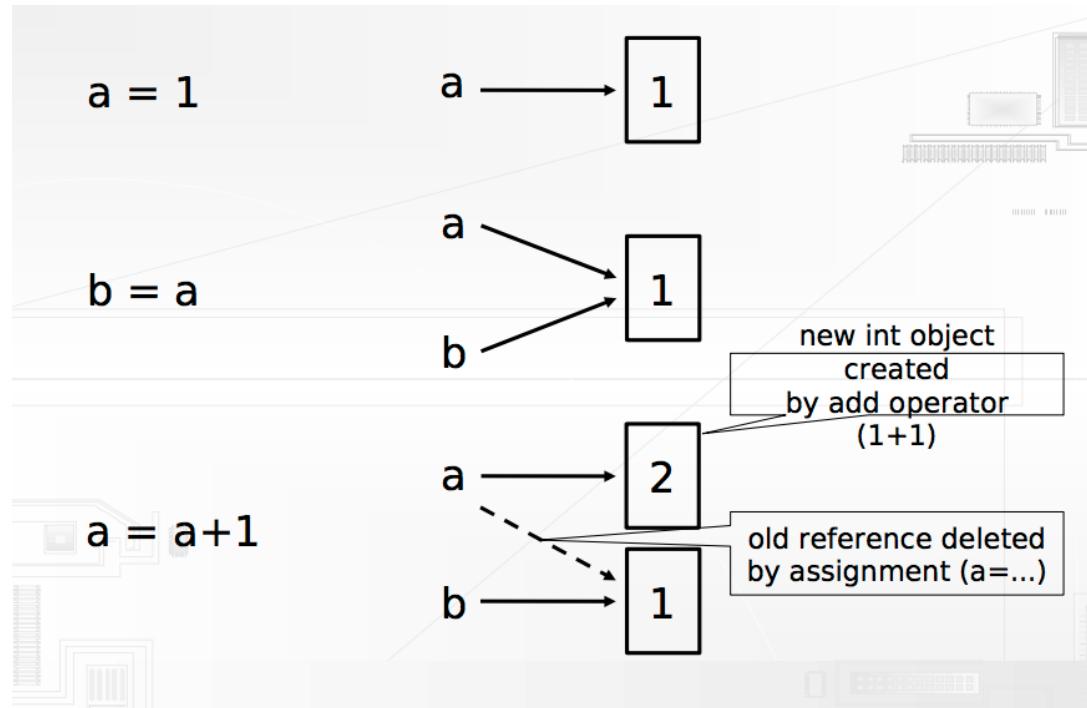
```
In [98]: a = 5
id(a)
```

```
Out[98]: 140559754770296
```

```
In [99]: a = 3
id(a)
```

```
Out[99]: 140559754770344
```

Tipo	Mutable ?
Numeri	No
Stringhe	No
Liste	Si
Dizionari	Si



## Passaggio di parametri

In Python il passaggio di parametri avviene per indirizzo: i parametri formali diventano degli alias dei parametri attuali.

```

In [100]: a = [1,2]
          def swap(x):
                  temp = x[0]
                  x[0] = x[1]
                  x[1] = temp

          swap(a)
          print a
      
```

[2, 1]

La funzione ha modificato l'oggetto passato come parametro

Se invece di modificare l'oggetto la funzione esegue un assegnamento (ovvero crea un nuovo binding), si interrompe il legame tra parametro formale e attuale che fanno ora riferimento a due celle distinte!

```
In [104]: def f(x):
    x = 1 # creo un nuovo binding. Perdo il collegamento con l'oggetto passato per parametro

a = 1000
f(a)
print a
```

1000

Le modifiche non sono visibili al chiamante

## Cosa manca

Esistono numerose caratteristiche di Python che non sono state affrontate:

- Supporto per il multi-thread (Java – style)
- Servizi del sistema operativo
- Protocolli di rete...
- Reflection (simile a Java)
- Parsing
- Debugger
- PyUnit (la versione Python di JUnit)
- Librerie grafiche 2D (Tkinter, PyQt) e anche 3D (SDL)
- ...

Noi ci concentreremo su due librerie esterne: Pandas e Numpy.

## Artificial Intelligence

Artificial intelligence (AI, also machine intelligence, MI) is intelligence demonstrated by machines, in contrast to the natural intelligence (NI) displayed by humans and other animals. In computer science AI research is defined as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving".

## Data Science

Data science is an interdisciplinary field of scientific methods, processes, algorithms and systems to extract knowledge or insights from data in various forms, either structured or unstructured, similar to data mining.

Data science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena" with data. It employs techniques and theories drawn from many fields within the broad areas of mathematics, statistics, information science, and computer science, in particular from the subdomains of machine learning, classification, cluster analysis, uncertainty quantification, computational science, data mining, databases, and visualization.

Turing award winner Jim Gray imagined data science as a "fourth paradigm" of science (empirical, theoretical, computational and now data-driven) and asserted that "everything about science is changing because of the impact of information technology" and the data deluge.

When Harvard Business Review called it "The Sexiest Job of the 21st Century" the term became a buzzword, and is now often applied to business analytics, or even arbitrary use of data, or used as a sexed-up term for statistics. While many university programs now offer a data science degree, there exists no consensus on a definition or curriculum contents. Because of the current popularity of this term, there are many "advocacy efforts" surrounding it.

## Machine Learning

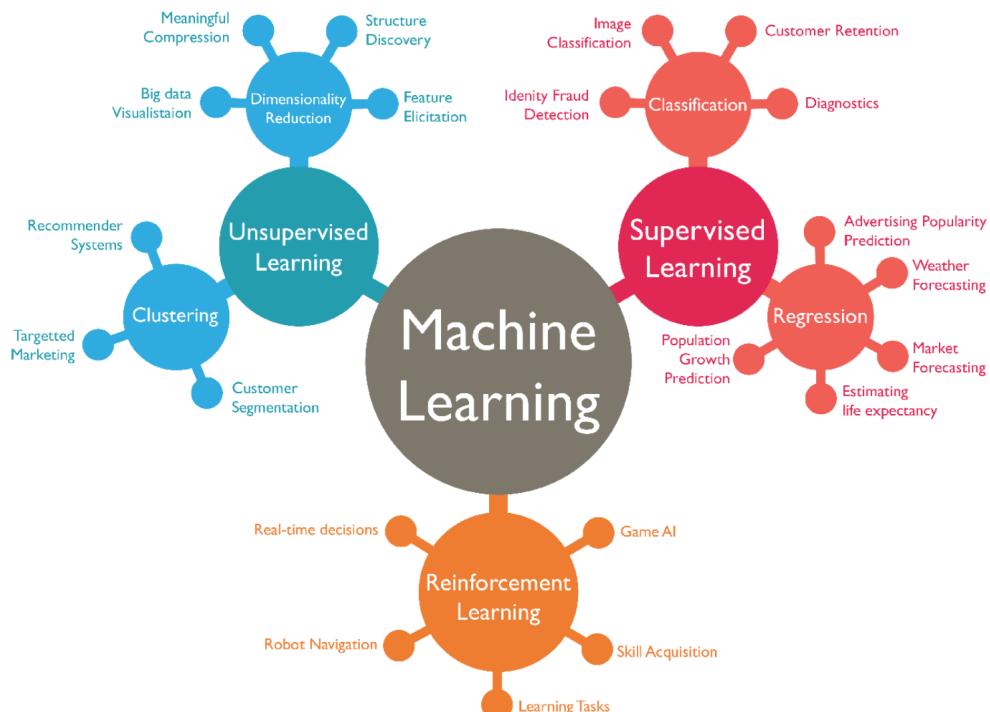
Machine learning, a fundamental concept of AI research since the field's inception, is the study of computer algorithms that improve automatically through experience.

- Unsupervised learning is the ability to find patterns in a stream of input.

- Supervised learning includes both classification and numerical regression.
  - Classification is used to determine what category something belongs in, after seeing a number of examples of things from several categories.
  - Regression is the attempt to produce a function that describes the relationship between inputs and outputs and predicts how the outputs should change as the inputs change.

Both classifiers and regression learners can be viewed as "function approximators" trying to learn an unknown (possibly implicit) function; for example, a spam classifier can be viewed as learning a function that maps from the text of an email to one of two categories, "spam" or "not spam".

- In reinforcement learning the agent is rewarded for good responses and punished for bad ones. The agent uses this sequence of rewards and punishments to form a strategy for operating in its problem space.



## Dal mondo reale ai dati

Il boom della IA è stato contemporaneo alla crescita della disponibilità (o alla possibilità di disporre) dei dati utili per le applicazioni di IA.

Training dataset ricchi e completi sono la base imprescindibile per sviluppare **agenti intelligenti** affidabili:

Data mining:

- sensori
- web (scraping, data API, ecc...)
- database
- da altri agenti
- ecc ecc

Per un agente intelligente, ovvero un software, i dati sono numeri....

...per costruire un agente dotato di IA e per alimentare le sue funzioni intelligenti, i dati sono messi sotto forma di tensori:

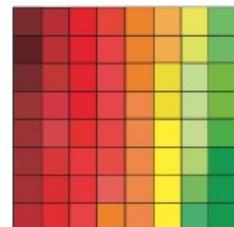
**tensor = multidimensional array**

vector



$$\mathbf{v} \in \mathbb{R}^{64}$$

matrix



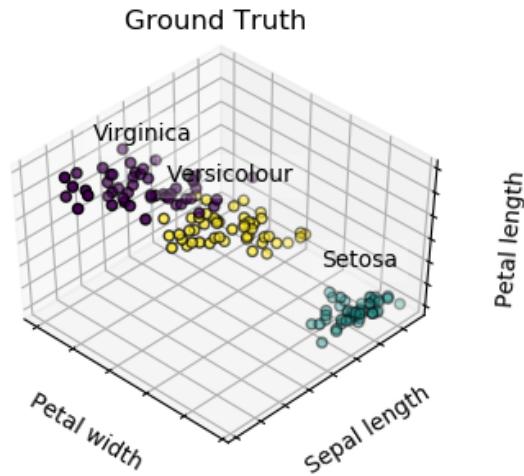
$$\mathbf{X} \in \mathbb{R}^{8 \times 8}$$

tensor



$$\mathbf{X} \in \mathbb{R}^{4 \times 4 \times 4}$$

Occorre scontrarsi (un poco :-) ) con l'algebra tensoriale e in particolare con l'astrazione richiesta da spazi a molte dimensioni come sono gli spazi usati dagli agenti intelligenti per apprendere e per esercitare le loro funzioni.



Ci sarebbe anche la features engineering...

## Features e samples (esempi)

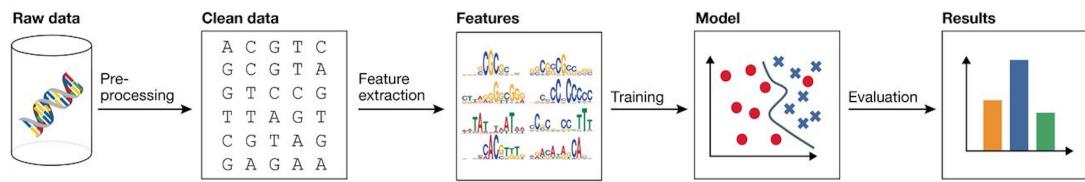
Un set di dati è composto da **esempi** ognuno dei quali è caratterizzato da più **features**:

$$X = \left( \begin{array}{cccc} & \text{n features} & & \\ & \xrightarrow{\quad\quad\quad} & & \\ \text{Feature} & \text{Feature} & \text{Feature} & \\ \#1 & \#2 & \#n & \\ \hline 1 & x_{1,1} & x_{1,2} & x_{1,n} \\ 1 & x_{2,1} & x_{2,2} & x_{2,n} \\ 1 & x_{3,1} & x_{3,2} & x_{3,n} \\ \dots & \dots & \dots & \dots \\ 1 & x_{m-1,1} & x_{m-1,2} & x_{m-1,n} \\ 1 & x_{m,1} & x_{m,2} & x_{m,n} \end{array} \right)$$

Value of feature #n for training example #1

m training examples

## Per riassumere:



## Esempio tratto dalla NAO Challenge

Sfida: riconoscere i movimenti effettuati dal braccio di una persona.

**PROBLEMA 1:** Capire come poter tradurre i movimenti in numeri

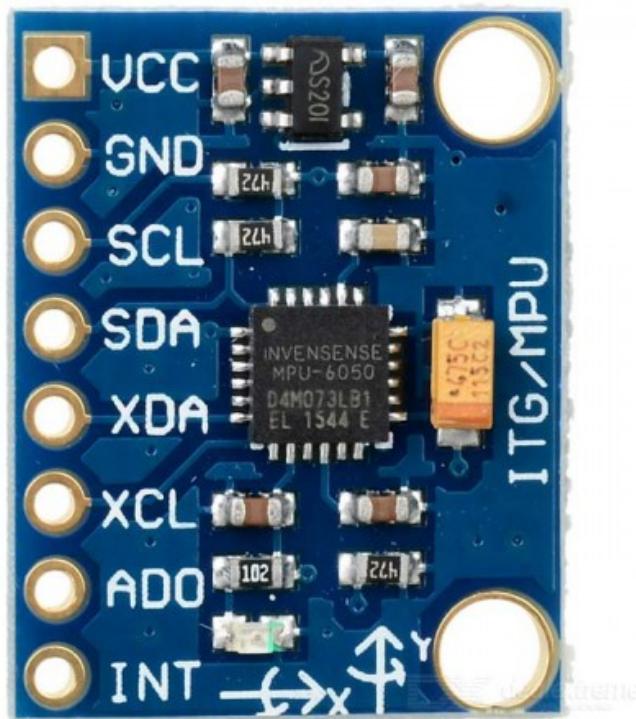
**PROBLEMA 2:** Capire come convertire i numeri in un dataset di features idoneo per:

1. Istruire l'agente IA
2. Permettere all'agente di riconoscere i movimenti una volta istruito

## PROBLEMA 1

Applico sul polso del braccio un sensore MPU-6050: accelerometro 3 assi + giroscopio 3 assi.

Il sensore si interfaccia tramite bus I2C (Raspberry PI, Arduino, Lego EV3, NAO)



E' sufficiente?

Risposta: si scopre solo provando!

Ogni 50ms il sensore misura 3 valori di accelerazione Ax, Ay, Az e 3 valori di velocità angolare Gx, Gy, Gz.

## PROBLEMA 2

Che me ne faccio dei sei numeri Ax, Ay, Az, Gx, Gy, Gz?

Essendo misure istantanee sono completamente inutili al fine di caratterizzare un intero movimento effettuato dal braccio!

## Idea!

Usiamo i sei numeri Ax, Ay, Az, Gx, Gy, Gz, registrati ogni 50ms, per alimentare sei code FIFO.

Esempio:

Ogni 50ms nella coda FIFO delle Ax trovo gli ultimi 20 valori di Ax, quindi tutta la storia delle Ax nell'ultimo secondo!

## SOTTO-PROBLEMA DEL PROBLEMA 2

Da sei code FIFO, come ottengo un tensore, o meglio una matrice bidimensionale, idoneo per istruire e successivamente alimentare un algoritmo di machine learning?

Molto semplice!

Concateno le sei code FIFO ottenendo una sola lista di 120 numeri.

Ogni 50ms ho una lista di 120 numeri che contiene i dati dell'ultimo secondo di movimento!

E quindi?

Quindi ogni 50ms ho un **campione** che è dato dalla lista di 120 numeri!

Ogni numero della lista è una **feature**!

In totale si hanno 120 feature, per cui lo spazio delle feature ha 120 dimensioni!

**ANTICIPAZIONE:** ...una volta che l'agente intelligente sarà stato istruito saprà riconoscere di che movimento si tratta per ognuno dei punti di questo spazio a 120 dimensioni.

In [ ]: