Wolfram *Mathematica*

Il software di riferimento per la Didattica, la Ricerca e lo Sviluppo

ADALTA

Distributore ufficiale per l'Italia di Wolfram Research

www.adalta.it/wolfram

Lezione 4

Strutture dati

Crescenzio Gallo – Università di Foggia

crescenzio.gallo@unifg.it

Note:

- Il materiale visualizzato durante questo seminario è disponibile per il download all'indirizzo http://www.crescenziogallo.it/unifg/seminario-mathematica-2014/
- Il materiale utilizzato è tratto dai webinar pubblicati da Adalta e prodotti dal dott. Roberto Cavaliere (*Mathematica* Technical Sales Manager, *r.cavaliere@adalta.it*)

Agenda

Introduzione

- *Mathematica* è un linguaggio
- *Mathematica* è un linguaggio funzionale e simbolico
- *Funzioni e data set*

Manipolazione dei dati

- Manipolazione basata sulla posizione dei dati
- Manipolazione basata sulla qualità dei dati
- Manipolazione basata sulla struttura dei dati
- Alcune macro

Conclusioni

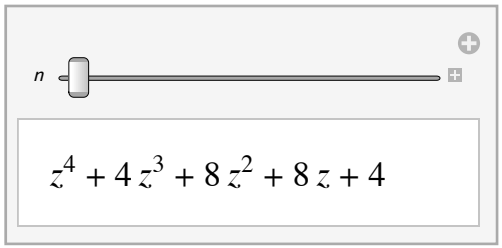
Introduzione: *Mathematica* è un linguaggio

Poichè siamo abituati ad usare *Mathematica* attraverso il front-end, capita spesso di pensare di non considerare la vera natura di *Mathematica* ossia quella di linguaggio di programmazione.
Già nel fare un semplice calcolo e renderlo parametrico e farlo computare al variare del parametro in effetti abbiamo sfruttato la capacità di programmazione del linguaggio *Mathematica*:

```
Expand[((z + 1)^2 + 1)^2]
```

$$z^4 + 4\,z^3 + 8\,z^2 + 8\,z + 4$$

```
Manipulate[Expand[((z + 1)^2 + 1)^n], {n, 2, 10, 1}]
```



Introduzione: *Mathematica* è un linguaggio funzionale e simbolico

Il principio di base di *Mathematica* è “Everything is an expression”.

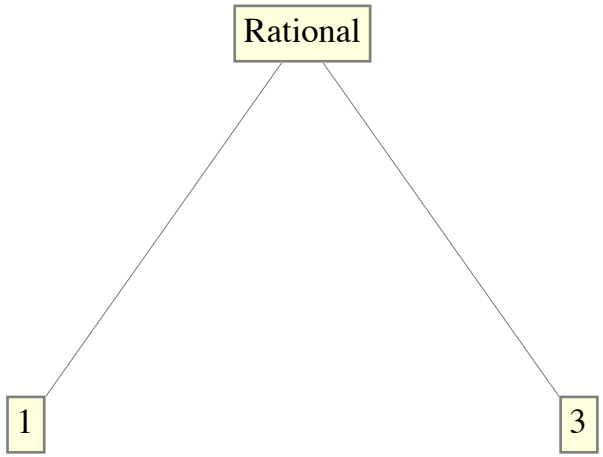
Ad eccezione delle entità atomiche (numeri, stringhe, simboli), tutto viene convertito internamente in espressioni o nidificazio di espressioni.

La manipolazione delle espressioni permette a *Mathematica* di eseguire calcoli simbolici, di modificare interattivamente grafici strutture dati, di animare o consentire all'utente di manipolare oggetti attraverso semplici interfacce.
Vediamo come viene rappresentato internamente un numero razionale:

```
FullForm[1/3]
```

Rational[1, 3]

```
TreeForm[1/3]
```

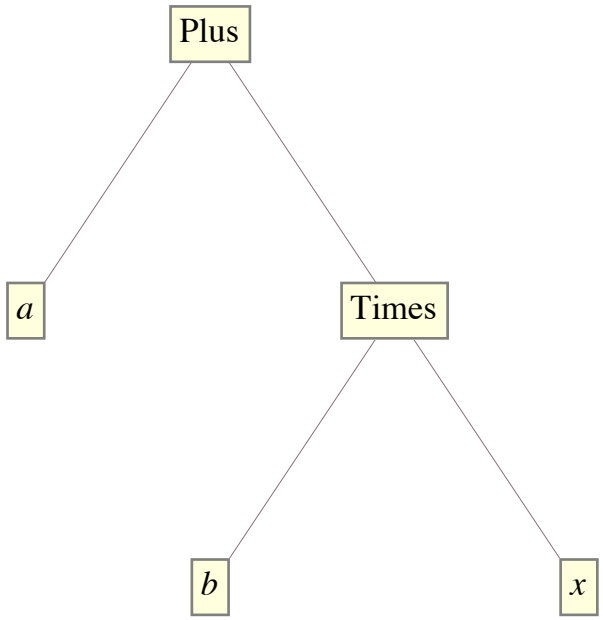


Un binomio:

```
FullForm[a + b x]
```

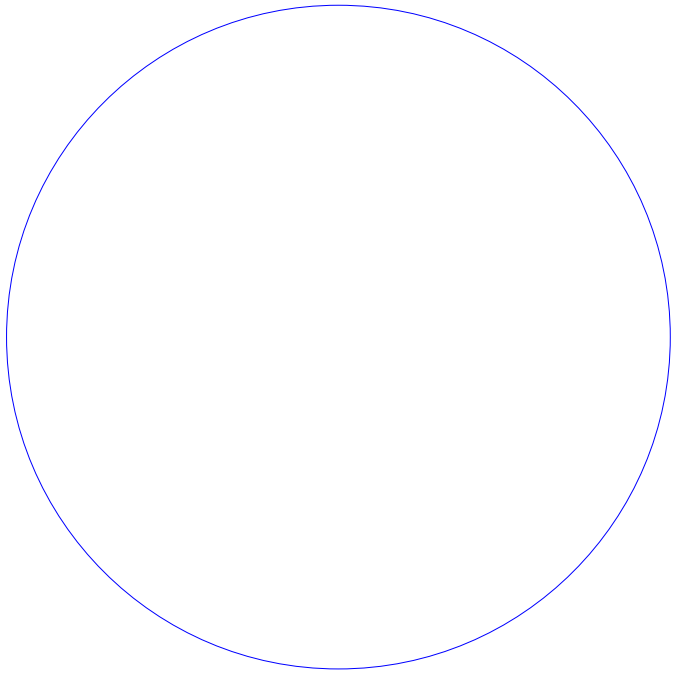
Plus[a, Times[b, x]]

```
TreeForm[a + b x]
```



Un grafico:

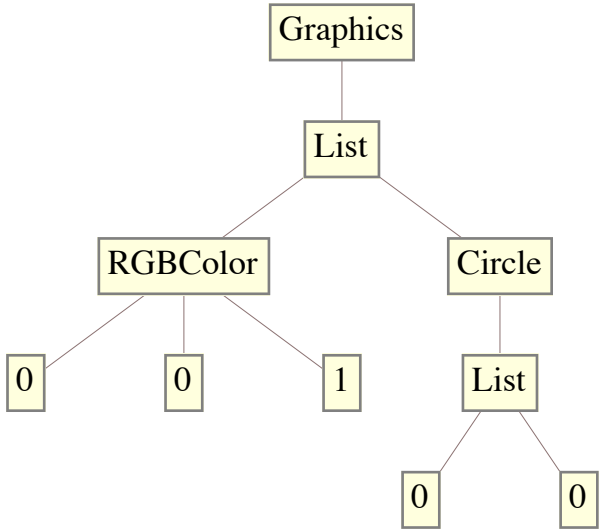
```
gr = Graphics[{Blue, Circle[{0, 0}]}]
```



```
FullForm[gr]
```

```
Graphics[List[RGBColor[0, 0, 1], Circle[List[0, 0]]]]
```

TreeForm[gr]



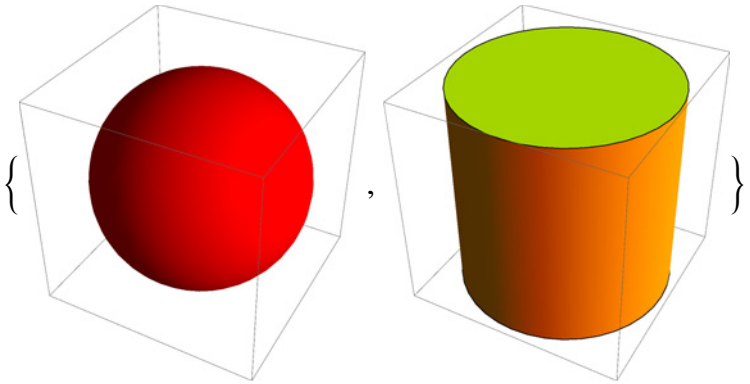
Introduzione: *Mathematica* è un linguaggio funzionale e simbolico

Le implicazioni pratiche.

Qualsiasi espressione può essere inclusa in qualsiasi altra espressione.

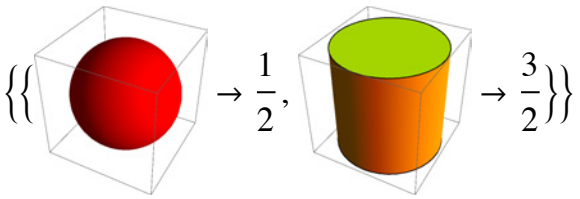
Esempio 1

```
{X, Y} = {Graphics3D[{Red, Sphere[]}], Graphics3D[{Yellow, Cylinder[]}]}
```



Queste due forme grafiche sono internamente rappresentate come qualsiasi altra espressione simbolica, pertanto possono esse usate esattamente come simboli:

```
Solve[{X + Y == 2, Y - X == 1}]
```



```
FullForm[X]
```

Graphics3D[List[RGBColor[1, 0, 0], Sphere[List[0, 0, 0]]]]

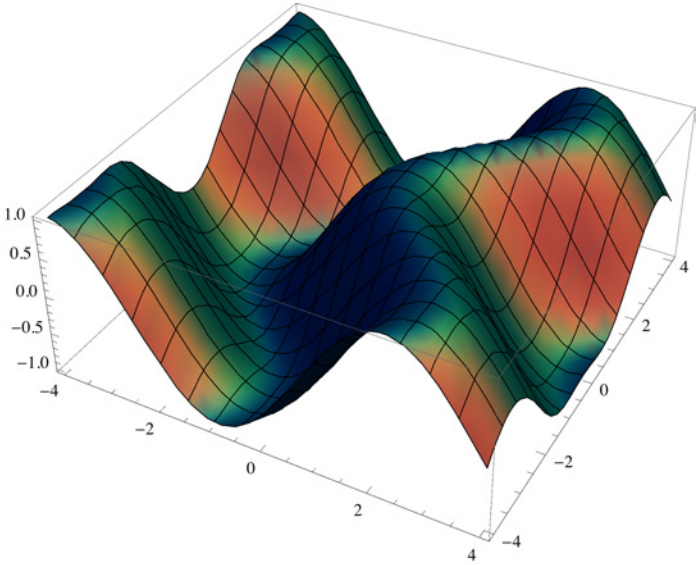
```
FullForm[Y]
```

Graphics3D[List[RGBColor[1, 1, 0], Cylinder[List[List[0, 0, −1], List[0, 0, 1]]]]]

Esempio 2

Gli elementi di interfaccia utente sono a loro volta rappresentati mediante espressioni e pertanto possono anche comparire direttamente dentro le righe di codice:

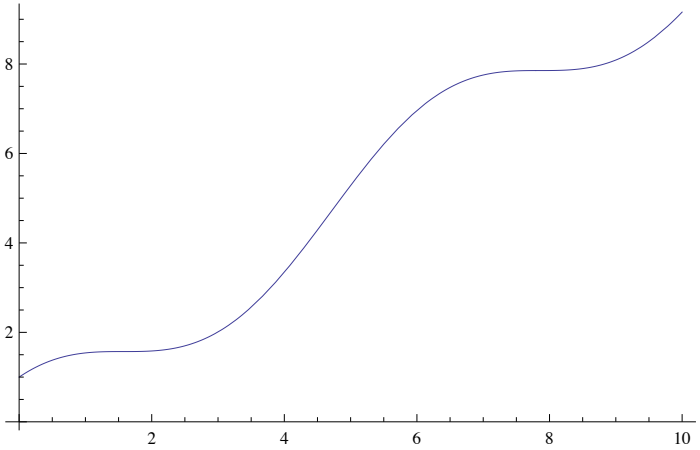
```
Plot3D[Sin[x + Cos[y]], {x, -4, 4}, {y, -4, 4}, PlotStyle → {■, Specularity[■, 10 ∨]}, Axes → ☑]
```



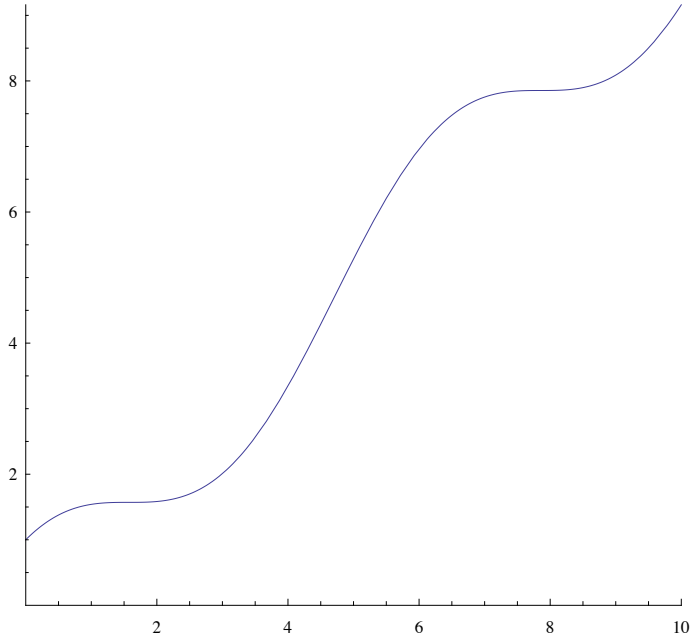
Esempio 3

Manipolare l'espressione di un grafico dopo averlo realizzato, ad esempio per ribaltare l'asse delle *x* rispetto a quello delle *y:*

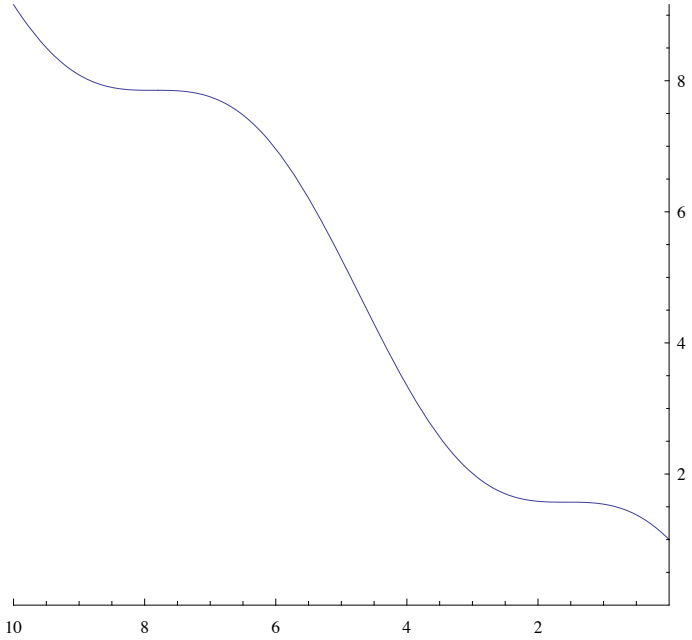
```
Plot[Cos[x] + x, {x, 0, 10}]
```




```
s = FullGraphics[Plot[Cos[x] + x, {x, 0, 10}]]
```



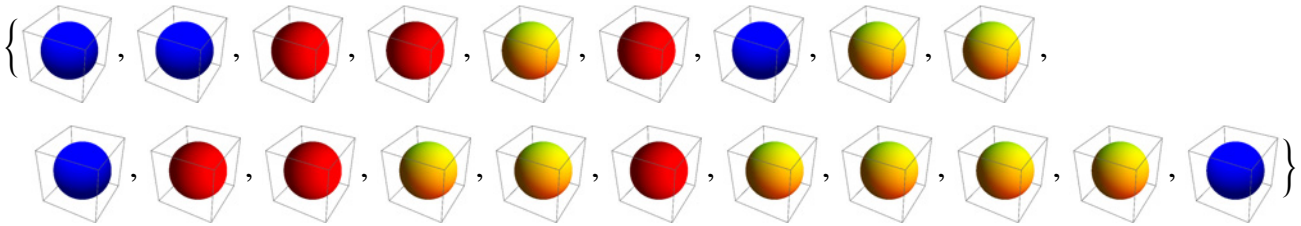

```
ReplaceAll[s, {x_Real, y_Real} → {-x, y}]
```



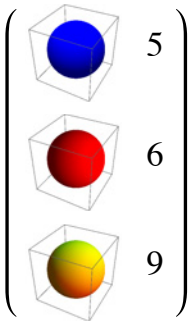
Esempio4

Possiamo eseguire operazioni su liste di oggetti di qualsiasi natura, ad esempio contare le ricorrenze di ciascun elemento grafico i una lista di 20 elementi:

```
spheres = RandomChoice[{ , , }, 20]
```



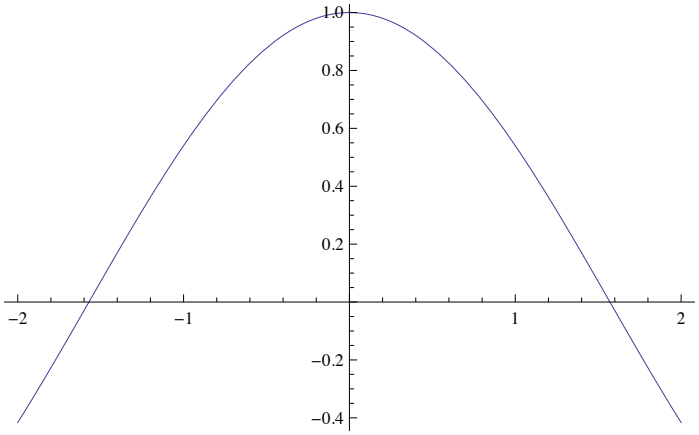
```
Tally[spheres]
```


5
6
9

Introduzione: funzioni e data set

Nel paradigma funzionale il modo più naturale per eseguire operazioni è quello di definire funzioni che operano su data set.
Come visto nella slide precedente, in *Mathematica* i data set non sono altro che generiche espressioni, sebbene in molti ca pratici si lavora su liste.

```
gr = Plot[Cos[x], {x, -2, 2}]
```




```
esp1 = -4 + 2 x - 3 x^2 + 2 x^3
```

$2\,x^3 - 3\,x^2 + 2\,x - 4$

```
list1 = {a, b, c, d}
```

$\{a, b, c, d\}$

```
Length[gr]
```

2

```
Length[esp1]
```

4

```
Length[list1]
```

4

```
Part[gr, 2]
```

$$\left\{\text{AspectRatio} \to \frac{1}{\phi},\ \text{Axes} \to \text{True},\ \text{AxesLabel} \to \{\text{None}, \text{None}\},\ \text{AxesOrigin} \to \{0, 0\},\ \text{Method} \to \{\},\right.$$
$$\left.\text{PlotRange} \to \begin{pmatrix} -2 & 2 \\ -0.416147 & 1. \end{pmatrix},\ \text{PlotRangeClipping} \to \text{True},\ \text{PlotRangePadding} \to \{\text{Scaled}[0.02], \text{Scaled}[0.02]\}\right\}$$

```
Part[esp1, 2]
```

$2\,x$

```
Part[list1, 2]
```

b

```
FullForm[gr]
```

```
Graphics[List[List[List[], List[], List[Hue[0.67`, 0.6`, 0.6`],
        Line[List[List[-1.999999918367347`, -0.4161467623187796`], List[-1.9987731283177614`, -0.41503093236829597`],
          List[-1.997546338268176`, -0.4139144777905993`], List[-1.995092758169005`, -0.41167970147562205`],
          List[-1.9901855979706633`, -0.4072027273140776`], List[-1.9803712775739797`, -0.39821947049719264`],
          List[-1.9607426367806124`, -0.3801387561945811`], List[-1.9214853551938782`, -0.34354498048351506`],
          List[-1.8363665716581652`, -0.26245956457718206`],
          List[-1.7568884657193913`, -0.18501992655153862`], List[-1.6789694047244954`, -0.10796223831452838`],
          List[-1.594446123367355`, -0.02364759202814696`], List[-1.515563519607154`, 0.05520472869086768`],
          List[-1.4300766954847084`, 0.14025566888491495`], List[-1.3461489163061406`, 0.2227626460928716`],
          List[-1.2678618147245122`, 0.2983223766816859`], List[-1.1829704927806393`, 0.37817661231503763`],
          List[-1.1037198484337056`, 0.4502778345056738`], List[-1.0260282490306498`, 0.5182197535308473`],
          List[-0.9417324292653497`, 0.5883881033758412`], List[-0.8630772870969888`, 0.6501022835146517`],
          List[-0.7778179245663834`, 0.7124464530479571`], List[-0.6941176069796561`, 0.7686184743509266`],
          List[-0.6160579669898679`, 0.8161625868387635`], List[-0.5313941066378353`, 0.8621014649094217`],
          List[-0.4523709238827418`, 0.899413302314375`], List[-0.45103299570903344`, 0.8999973044777072`],
          List[-0.449695067535325`, 0.9005796955994867`], List[-0.4470192111879082`, 0.9017396405512327`],
          List[-0.4416674984930746`, 0.9040401522790932`], List[-0.43096407310340734`, 0.9085634333957794`],
          List[-0.4095572223240729`, 0.9172972282209694`], List[-0.3667435207654039`, 0.9334999931683206`],
          List[-0.3654299526689436`, 0.9339702035268338`], List[-0.3641163845724833`, 0.9344388023562828`],
          List[-0.36148924837956264`, 0.9353711621965747`], List[-0.35623497599372145`, 0.9372165081387078`],
```

List[−0.34572643122203894`, 0.9408295276987386`], List[−0.32470934167867394`, 0.9477434961393536`],
List[−0.3233957735822136`, 0.9481617503754047`], List[−0.3220822054857533`, 0.9485783685954325`],
List[−0.3194550692928327`, 0.9494066941148086`], List[−0.3142007969069914 3`, 0.9510436815207947`],
List[−0.3036922521353089`, 0.9542388445779049`], List[−0.282675162591944`, 0.9603127040073371`],
List[−0.28144973008293583`, 0.9606537874205473`], List[−0.28022429757392775`, 0.9609934282347146`],
List[−0.2777734325559 1147`, 0.961668380027954`], List[−0.2728717025198789`, 0.9630009501296866`],
List[−0.2630682424478138`, 0.9655966450622685`], List[−0.2434613223036836`, 0.9705093924184968`],
List[−0.24223588979467547`, 0.9708040704613284`], List[−0.24101045728566733`, 0.9710972906625929`],
List[−0.23855959226765105`, 0.9716793557813134`], List[−0.23365786223161852`, 0.9728259726924399`],
List[−0.22385440215955343`, 0.9750490575225859`], List[−0.20424748201542325`, 0.9792138952410068`],
List[−0.20291885856325503`, 0.9794825160362572`], List[−0.20159023511108684`, 0.9797494078097734`],
List[−0.19893298820675043`, 0.980278002410144`], List[−0.1936184943980776`, 0.981314422825036`],
List[−0.18298950678073195`, 0.983304087085177`], List[−0.18166088332856373`, 0.9835449887084587`],
List[−0.18033225987639553`, 0.983784154138767`], List[−0.17767501297205912`, 0.9842572747347906`],
List[−0.1723605191633863`, 0.9851826632404749`], List[−0.16173153154604064`, 0.9869499390600767`],
List[−0.16040290809387242`, 0.9871630126516816`], List[−0.15907428464170423`, 0.9873743436636324`],
List[−0.15641703773736781`, 0.9877917764594483`], List[−0.151102543928695`, 0.9886057148872087`],
List[−0.14047355631134933`, 0.9901498036617327`], List[−0.11921558107665804`, 0.9929022349209508`],
List[−0.11797509321194202`, 0.9930490063694967`], List[−0.11673460534722599`, 0.9931942497043554`],
List[−0.11425362961779392`, 0.9934801511413562`], List[−0.10929167815892979`, 0.9940336069746212`],
List[−0.09936777524120155`, 0.9950670835759523`], List[−0.09812728737648552`, 0.9951893797029018`],
List[−0.09688679951176948`, 0.9953101444225365`], List[−0.09440582378233742`, 0.9955470788988824`],
List[−0.0894438723234733`, 0.9960025629450259`], List[−0.07951996940574504`, 0.9968399529529888`],
List[−0.07827948154102901`, 0.9969377255831501`], List[−0.07703899367631298`, 0.9970339641156246`],
List[−0.07455801794688091`, 0.9972218382975028`], List[−0.06959606648801679`, 0.9975791711311267`],
List[−0.06835557862330077`, 0.9976646669654211`], List[−0.06711509075858474`, 0.9977486275834041`],
List[−0.06463411502915267`, 0.9979119426560005`], List[−0.05967216357028854`, 0.9982201446789857`],
List[−0.05843167570557250 6`, 0.9982933552975234`], List[−0.05719118784085647`, 0.9983650297323179`],
List[−0.05471021211142441`, 0.9985037696118675`], List[−0.04974826065256028 6`, 0.9987628104715415`],
List[−0.04850777278784425`, 0.9988237286643182`], List[−0.04726728492312822 6`, 0.9988831098572079`],
List[−0.04478630919369617`, 0.9989972608801855`], List[−0.03982435773483204`, 0.9992071150654167`],
List[−0.0386082299473641`, 0.9992547948636786`], List[−0.03739210215989616`, 0.999300996797461`],
List[−0.03495984658496029`, 0.9993889668004479`], List[−0.03374371879749235 6`, 0.9994307347395478`],

List[−0.03252759101002441`, 0.999471024553959`], List[−0.030095335435088535`, 0.9995471695725516`],
List[−0.0288792076476206`, 0.999583024664117 2`], List[−0.02766307986015266`, 0.9996174014057623`],
List[−0.025230824285216787`, 0.999681719638108 8`], List[−0.024014696497748848`, 0.9997116610336859`],
List[−0.02279856871028091`, 0.999740123889093 6`], List[−0.020366313135345034`, 0.9997926138132047`],
List[−0.019150185347877095`, 0.999816640804277 5`], List[−0.01793405756040916`, 0.9998391890999191`],
List[−0.01671792977294122`, 0.999860258666781 5`], List[−0.015501801985473282`, 0.9998798494737036`],
List[−0.014285674198005345`, 0.99989796149171 1`], List[−0.013069546410537407`, 0.9999145946940168`],
List[−0.011853418623069469`, 0.99992974905602 1`], List[−0.010637290835601531`, 0.9999434245553109`],
List[−0.009421163048133594`, 0.99995562117166 08`], List[−0.008205035260665655`, 0.9999663388870323`],
List[−0.006988907473197718`, 0.99997557768557 44`], List[−0.00577277968572978`, 0.9999833375536231`],
List[−0.004556651898261842`, 0.99998961847970 17`], List[−0.00334052411079390 4`, 0.9999944204545211`],
List[−0.002124396323325966 4`, 0.999997743470979 4`], List[−0.000908268535858028 4`, 0.9999995875241617`],
List[0.000307859251609909 5`, 0.99999995261134 1`], List[0.001523987039077847 4`, 0.999998838731977 1`],
List[0.002740114826545785`, 0.99999624588771 75`], List[0.003956242614013723`, 0.999992174082397`],
List[0.005172370401481661`, 0.99998662332203 76`], List[0.006388498188949598`, 0.999979593614848 7`],
List[0.007604625976417536 5`, 0.99997108497122 7`], List[0.008820753763885475`, 0.999961097403756 4`],
List[0.010036881551353412`, 0.99994963092720 83`], List[0.011253009338821 35`, 0.999936685558541 2`],
List[0.012469137126289288`, 0.99992226131690 09`], List[0.013685264913757225`, 0.999906358223620 4`],
List[0.014901392701225164`, 0.99988897630221 96`], List[0.016117520488693103`, 0.999870115578406 2`],
List[0.018549776063628978`, 0.99982795783730 53`], List[0.019765903851096916`, 0.999804660882367 7`],
List[0.020982031638564852`, 0.99977988524971 71`], List[0.023414287213500 73`, 0.999725898100032 4`],
List[0.024630415000968665`, 0.99969668666284 37`], List[0.025846542788436604`, 0.999665996707632 2`],
List[0.028278798363372 48`, 0.99960018142688 47`], List[0.029494926150840418`, 0.999565056198687 4`],
List[0.030711053938308357`, 0.99952845264714 43`], List[0.033143309513244235`, 0.999450810792748 7`],
List[0.038007820663115 98`, 0.99927778973224 46`], List[0.039327139393743 99`, 0.999226787716975 4`],
List[0.040646458124371995`, 0.99917404644590 02`], List[0.043285095585628014`, 0.999063346506565 6`],
List[0.048562370508140 05`, 0.99882107980030 55`], List[0.049881689238768 06`, 0.99875616647854 3`],
List[0.051201007969396 07`, 0.99868951472013 88`], List[0.053839645430652 08`, 0.99855099636048 7`],
List[0.059116920353164 11`, 0.99825310370881 66`], List[0.060436239083792115`, 0.998174286311741 9`],
List[0.061755557814420124`, 0.99809373149084 68`], List[0.064394195275676 14`, 0.997927410141475 5`],
List[0.069671470198188 17`, 0.99757392472888 78`], List[0.080226020043212 23`, 0.996783618519351 6`],
List[0.081545338773840 24`, 0.99667702085973 34`], List[0.082864657504468 25`, 0.996568688382437 7`],
List[0.085503294965724 26`, 0.99634681973208 79`], List[0.090780569888823 63`, 0.995882273118334 8`],

List[0.10133511973326036`, 0.9948699889334509`], List[0.10265443846388836`, 0.9947356585088126`],
List[0.10397375719451638`, 0.9945995966456355`], List[0.10661239465577238`, 0.9943222795539954`],
List[0.11188966957828442`, 0.9937468787306158`], List[0.12244421942330848`, 0.9925130676214851`],
List[0.12367540256648432`, 0.9923619405786739`], List[0.12490658570966015`, 0.9922093093019823`],
List[0.1273689519960118`, 0.9918995349746786`], List[0.13229368456871515`, 0.9912619458460844`],
List[0.1421431497141218`, 0.9899146605802462`], List[0.16184208000493513`, 0.986932131700442`],
List[0.163073263148111`, 0.9867329952224582`], List[0.16430444629128682`, 0.9865323630430154`],
List[0.16676681257763848`, 0.986126612798502`], List[0.1716915451503418`, 0.9852971774251202`],
List[0.18154101029574846`, 0.9835666383085802`], List[0.20123994058656178`, 0.9798194863349199`],
List[0.20257431467289766`, 0.9795518935369391`], List[0.20390868875923357`, 0.9792825565939768`],
List[0.2065774369319054`, 0.9787386521944879`], List[0.21191493327724897`, 0.9776299349647307`],
List[0.2225899259679362`, 0.9753289783747889`], List[0.24393991134931065`, 0.970393911017271`],
List[0.24527428543564656`, 0.970070758813328`], List[0.24660865952198247`, 0.9697458793460754`],
List[0.24927740769465426`, 0.9690909409385764`], List[0.25461490403999787`, 0.9677603626222693`],
List[0.2652898967306851`, 0.9650165336885793`], List[0.28663988211205954`, 0.9591992974218294`],
List[0.2879498961211473`, 0.9588280931546002`], List[0.2892599101302351`, 0.9584552434075228`],
List[0.29187993814841073`, 0.9577046100360904`], List[0.297119994184762`, 0.9561836258779972`],
List[0.3076001062574645`, 0.9530629350191506`], List[0.3285603304028694`, 0.9465078778568997`],
List[0.3704807786936793`, 0.9321533808648682`], List[0.3717026571153149`, 0.9317102873878936`],
List[0.3729245355369506`, 0.9312658028798596`], List[0.37536829238022185`, 0.9303726634271269`],
List[0.38025580606676435`, 0.9285697214867874`], List[0.39003083343984946`, 0.9248973369010297`],
List[0.40958088818601956`, 0.9172878041419392`], List[0.44868099767835984`, 0.901020039451627`],
List[0.533485437025285`, 0.8610398286554971`], List[0.6126491987752708`, 0.8181275035996156`],
List[0.6902539155813786`, 0.7710843634278312`], List[0.7744628527497309`, 0.7147967815387625`],
List[0.853031112321144`, 0.6577029022372834`], List[0.9382035922548017`, 0.5912377762305614`],
List[1.01773539459152`, 0.5252942960131319`], List[1.095708151984360 6`, 0.4574168586182373`],
List[1.1802851297394454`, 0.3806611762146488 4`], List[1.2592214298975912`, 0.3065580846894268`],
List[1.3447619504179815`, 0.2241145465940652 7`], List[1.4287434259944938`, 0.1415756343823171 6`],
List[1.5070842239740 67`, 0.0636690078668007 3`], List[1.5920292423158844`, −0.021231320127351763`],
List[1.6713335830607627`, −0.10036797443529705`], List[1.749078878861763`, −0.17733960990458003`],
List[1.8334283950250079`, −0.2596232628159326`], List[1.9121372335913134`, −0.3347509380903199`],
List[1.9135100880409 39`, −0.3360442720943074 3`], List[1.9148829424905645`, −0.33733697274589514`],
List[1.9176286513898155`, −0.3399204642475082`], List[1.9231200691883177`, −0.3450797399013008`],

```
            List[1.9341029047853218`, −0.3553669176884197`], List[1.9560685759793301`, −0.3758114429436639`],
            List[1.9574414304289558`, −0.3770833073292904`], List[1.9588142848785812`, −0.3783544610150554`],
            List[1.9615599937778323`, −0.3808946267052246 7`], List[1.9670514115763345`, −0.3859663243631962`],
            List[1.9780342471733385`, −0.3960746499591388`], List[1.979407101622964`, −0.3973348564141901`],
            List[1.9807799560725896`, −0.3985943140006975`], List[1.9835256649718405`, −0.4011109730745506`],
            List[1.9890170827703426`, −0.4061352004504471`], List[1.9903899372199683`, −0.4073893495721328`],
            List[1.9917627916695937`, −0.408642730875279`], List[1.9945085005688448`, −0.4111471805782688`],
            List[1.9958813550184704`, −0.4123982442579033`], List[1.9972542094680958`, −0.413648530678589`],
            List[1.9986270639177213`, −0.4148980374838749 4`], List[1.999999918367347`, −0.4161467623187796`]]]]]],
  List[Rule[AspectRatio, Power[GoldenRatio, −1]], Rule[Axes, True],
    Rule[
      AxesLabel,
      List[None, None]], Rule[
      AxesOrigin,
      List[0, 0]], Rule[
      Method,
      List[]],
    Rule[PlotRange, List[List[−2, 2], List[−0.4161467623187796`, 0.999999952611341`]]],
    Rule[
      PlotRangeClipping,
      True], Rule[
      PlotRangePadding,
      List[Scaled[0.02`], Scaled[0.02`]]]]]
```

`FullForm[esp1]`

```
Plus[−4, Times[2, x], Times[−3, Power[x, 2]], Times[2, Power[x, 3]]]
```

`FullForm[list1]`

```
List[a, b, c, d]
```

Una cosa cui bisogna fare sempre attenzione: *Mathematica* gestisce tutti gli oggetti come strutture dati ancor prima di riconosce gli significati teorici. Questo significa che dobbiamo fare attenzione noi al significato dei risultati e non pensare che *Mathematic* faccia alcun tipo di “ragionamento” sui nostri dati a meno che non glielo chiediamo esplicitamente.

```
A = RandomInteger[{1, 5}, {5, 5}];
MatrixForm[A]
```

$$\begin{pmatrix} 3 & 4 & 1 & 1 & 1 \\ 3 & 3 & 5 & 4 & 3 \\ 5 & 4 & 4 & 5 & 1 \\ 2 & 5 & 4 & 5 & 1 \\ 3 & 4 & 1 & 5 & 5 \end{pmatrix}$$

```
B = RandomInteger[{1, 5}, {5, 5}];
MatrixForm[B]
```

$$\begin{pmatrix} 5 & 3 & 5 & 5 & 4 \\ 2 & 2 & 4 & 1 & 3 \\ 1 & 3 & 1 & 2 & 4 \\ 3 & 5 & 4 & 3 & 2 \\ 5 & 2 & 2 & 1 & 3 \end{pmatrix}$$

```
MatrixForm[A/B]
```

$$\begin{pmatrix} \frac{3}{5} & \frac{4}{3} & \frac{1}{5} & \frac{1}{5} & \frac{1}{4} \\ \frac{3}{2} & \frac{3}{2} & \frac{5}{4} & 4 & 1 \\ 5 & \frac{4}{3} & 4 & \frac{5}{2} & \frac{1}{4} \\ \frac{2}{3} & 1 & 1 & \frac{5}{3} & \frac{1}{2} \\ \frac{3}{5} & 2 & \frac{1}{2} & 5 & \frac{5}{3} \end{pmatrix}$$

La divisione tra matrici non è una operazione definita, ma in questo caso *Mathematica* sta facendo un'operazione lecita ossia opera con *A* e *B* come strutture dati.
Dunque, in *Mathematica* qualsiasi cosa può rappresentare un data set.

Introduzione: funzioni e data set

Applicare funzioni su liste.

In *Mathematica* difficilmente si ricorre ai cicli del tipo **For**, **While** o **Table** per due motivi principali:
- molte funzioni native hanno uno speciale attributo chiamato **Listable** che gli consente di operare direttamente su liste oltre cł su singoli valori;
- si usano operatori quali **Apply**, **Map**, **Replace** per applicare funzioni su interi data set.

Esempio 1: il calcolo di una funzione trigonometrica sui valori di un array.

Tutte le funzioni trigonometriche sono **Listable**

```
Attributes[Sin]
```

{Listable, NumericFunction, Protected}

```
a = RandomReal[{0, 100}, {100 000}];
```

```
(* approccio procedurale *)
res = Table[Sin[a[[i]]], {i, 1, Length[a]}];
```

```
(* uso dell'attributo Listable *)
res1 = Sin[a];
```

```
res === res1
```

True

Anche **Power** è **Listable**

```
Attributes[Power]
```

{Listable, NumericFunction, OneIdentity, Protected}

```
a = {2, 3, 4, 5, 6, 7, 8, 9, 10}
```

{2, 3, 4, 5, 6, 7, 8, 9, 10}

```
x^a
```

$\{x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9, x^{10}\}$

Esempio 2: sommare gli elementi di una lista.

```
a = RandomReal[{0, 100}, {100 000}];
```

```
(* approccio procedurale *)
res = 0;
Table[res = res + a[[i]], {i, 1, Length[a]}];
```

```
(* approccio funzionale *)
res1 = Apply[Plus, a];
```

```
res
```

5.01845×10^6

```
res1
```

5.01845×10^6

```
Plus @@ a
```

5.01845×10^6

Misuriamo i tempi

```
AbsoluteTiming[res1 = Apply[Plus, a];]
```

{0.024807, Null}

```
AbsoluteTiming[
 res = 0;
 Table[res = res + a[[i]], {i, 1, Length[a]}];]
```

{0.116081, Null}

Manipolazione dei dati: basata sulla posizione dei dati

Vediamo praticamente alcune operazioni sui dati, rimarcando ancora una volta che le stesse operazioni si possono eseguire s generiche espressioni oltre che su liste.

```
Clear[a]
```

```
lista = {simbolo, 2, 3, "stringa", x^2 - 4 == 0, a, w, π, {0, -4}}
```

$\{\text{simbolo}, 2, 3, \text{stringa}, x^2 - 4 = 0, a, w, \pi, \{0, -4\}\}$

```
esp = -4 + 2 x - 3 x^2 + 2 x^3
```

$2\,x^3 - 3\,x^2 + 2\,x - 4$

```
Join[lista, Reverse[lista]]
```

$\{\text{simbolo}, 2, 3, \text{stringa}, x^2 - 4 = 0, a, w, \pi, \{0, -4\}, \{0, -4\}, \pi, w, a, x^2 - 4 = 0, \text{stringa}, 3, 2, \text{simbolo}\}$

```
Join[esp, Reverse[esp]]
```

$4\,x^3 - 6\,x^2 + 4\,x - 8$

Proviamo a cambiare la Head di esp e capire cosa ha fatto la **Join** nel caso di espressioni:

```
esp1 = Apply[pippo, esp]
```

$\text{pippo}\left(-4, 2\,x, -3\,x^2, 2\,x^3\right)$

```
Join[esp1, Reverse[esp1]]
```

$\text{pippo}\left(-4, 2\,x, -3\,x^2, 2\,x^3, 2\,x^3, -3\,x^2, 2\,x, -4\right)$

Le funzioni che seguono operano sulle espressioni in base alla posizione degli elementi da manipolare:

```
(* elimina i primi tre elementi *)
Column[{Drop[lista, 3], Drop[esp, 3]}]
```

$\{\text{stringa}, x^2 - 4 = 0, a, w, \pi, \{0, -4\}\}$

$2\,x^3$

```
(* prende i primi tre elementi *)
Column[{Take[lista, 3], Take[esp, 3]}]
```

{simbolo, 2, 3}

$-3\,x^2 + 2\,x - 4$

```
(* elimina il terzo elemento a partire dalla fine *)
Column[{Delete[lista, -3], Delete[esp, -3]}]
```

$\{\text{simbolo}, 2, 3, \text{stringa}, x^2 - 4 = 0, a, \pi, \{0, -4\}\}$

$2\,x^3 - 3\,x^2 - 4$

```
(* inserisce un elemento al terzo posto *)
Column[{Insert[lista, pippo, 3], Insert[esp, pippo, 3]}]
```

$\{\text{simbolo}, 2, \text{pippo}, 3, \text{stringa}, x^2 - 4 = 0, a, w, \pi, \{0, -4\}\}$

$\text{pippo} + 2\,x^3 - 3\,x^2 + 2\,x - 4$

Manipolazione dei dati: basata sulla qualità dei dati

Se non conosco a priori la posizione dei dati nella struttura, ma ne conosco alcune proprietà posso usare l'istruzione **Select**.
La chiamata standard è **Select**[*espressione*, *criterio*].

```
lista = {simbolo, 2, 3, "stringa", x^2 - 4 == 0, a, w, π, {0, -4}}
```

$\{\text{simbolo}, 2, 3, \text{stringa}, x^2 - 4 = 0, a, w, \pi, \{0, -4\}\}$

```
Select[lista, IntegerQ]
```

{2, 3}

```
Select[lista, StringQ]
```

{stringa}

```
Select[lista, ListQ]
```

(0 −4)

Un esempio più articolato

```
lista = RandomReal[{-5, 5}, {10000}];
res = Select[lista, # > 0 &];
Length[res]
```

4948

```
(* approccio procedurale *)
res1 = {};
Table[If[lista[[i]] > 0, AppendTo[res1, lista[[i]]], {i, Length[lista]}];
res == res1
```

True

Misuriamo i tempi

```
AbsoluteTiming[
 (* approccio procedurale *)
 res1 = {};
 Table[If[lista[[i]] > 0, AppendTo[res1, lista[[i]]], {i, Length[lista]}];]
```

{0.114112, Null}

```
(* approccio funzionale *)
AbsoluteTiming[res = Select[lista, # > 0 &];]
```

{0.007830, Null}

Dunque, la **Select** fa uso di predicati del tipo **ListQ**, **IntegerQ**, ecc.

```
?? System`*Q
```

▼ System`

AcyclicGraphQ	DirectoryQ	ImageQ	MachineNumberQ	PlanarGraphQ	StringQ
AlgebraicIntegerQ	DiscreteTimeModelQ	IndependentEdgeSetQ	MarcumQ	PolynomialQ	SymmetricMatrixQ
AlgebraicUnitQ	DistributionParameterQ	IndependentVertexSetQ	MatchLocalNameQ	PositiveDefiniteMatrixQ	SyntaxQ
ArgumentCountQ	DuplicateFreeQ	InexactNumberQ	MatchQ	PossibleZeroQ	TautologyQ
ArrayQ	EdgeCoverQ	IntegerQ	MatrixQ	PrimePowerQ	TensorQ
AtomQ	EdgeQ	IntervalMemberQ	MemberQ	PrimeQ	TreeGraphQ
BinaryImageQ	EllipticNomeQ	InverseEllipticNomeQ	NameQ	ProcessParameterQ	TrueQ
BipartiteGraphQ	EmptyGraphQ	IrreduciblePolynomialQ	NumberQ	QHypergeometricPFQ	UnateQ
	EulerianGraphQ	IsomorphicGraphQ		QuadraticIrrationalQ	UndirectedGraphQ

BusinessDayQ	Q	phQ	NumericQ	onalQ	phQ
CompatibleUnitQ	EvenQ	KnownUnitQ	ObservableModelQ	QuantityQ	UnsameQ
CompleteGraphQ	ExactNumberQ	LeapYearQ	OddQ	RootOfUnityQ	UpperCaseQ
ConnectedGraphQ	FileExistsQ	LegendreQ	OptionQ	SameQ	ValueQ
ContinuousTimeModelQ	FreeQ	LetterQ	OrderedQ	SatisfiableQ	VectorQ
ControllableModelQ	GraphQ	LinkConnectedQ	OutputControllableModelQ	ScheduledTaskActiveQ	VertexCoverQ
CoprimeQ	GroupElementQ	LinkReadyQ	PartitionsQ	SimpleGraphQ	VertexQ
DayMatchQ	HamiltonianGraphQ	ListQ	PathGraphQ	SquareFreeQ	WeaklyConnectedGraphQ
DigitQ	HermitianMatrixQ	LoopFreeGraphQ	PermutationCyclesQ	StringFreeQ	WeightedGraphQ
DirectedGraphQ	HypergeometricPFQ	LowerCaseQ	PermutationListQ	StringMatchQ	

Oppure utilizza funzioni pure per definire una condizione da verificare per ciascun elemento da selezionare.

Manipolazione dei dati: basata sulla struttura dei dati

Un altro modo di selezionare elementi da una struttura è dato da **Cases**, la quale a differenza di **Select** permette di specificare u pattern e non un criterio.

La chiamata standard è **Cases**[*espressione*, *pattern*].

```
lista = RandomInteger[{-10, 10}, {100}]
```

{6, −10, −5, −7, 10, 0, 1, 7, −2, 2, −10, 9, −4, −2, 3, −9, −8, −3, −5, 6, −6, −4, −4, 0, 6, 5, 6, 4, −10, 4, 1, −7, −6, −10, −6, −7, 8, 9, −2, 8, −10, 8, 9, 1, 1, −3, −4, −1, 3, 3, −4, −5, −8, 3, −4, 2, −8, −1, −8, −8, −4, 4, 9, −4, 6, 0, −10, −3, 6, −5, 0, −3, 7, −5, 4, −7, 10, −5, −1, 0, −3, 10, 10, 6, 4, 0, −9, −7, 4, 1, 5, −10, −3, 7, 10, 10, 1, 8, 8, 4}

```
Cases[lista, Except[0]]
```

{6, −10, −5, −7, 10, 1, 7, −2, 2, −10, 9, −4, −2, 3, −9, −8, −3, −5, 6, −6, −4, −4, 6, 5, 6, 4, −10, 4, 1, −7, −6, −10, −6, −7, 8, 9, −2, 8, −10, 8, 9, 1, 1, −3, −4, −1, 3, 3, −4, −5, −8, 3, −4, 2, −8, −1, −8, −8, −4, 4, 9, −4, 6, −10, −3, 6, −5, −3, 7, −5, 4, −7, 10, −5, −1, −3, 10, 10, 6, 4, −9, −7, 4, 1, 5, −10, −3, 7, 10, 10, 1, 8, 8, 4}

```
(* tutti i numeri pari > 2 *)
Cases[lista, _?(# > 2 && EvenQ[#] &)]
```

{6, 10, 6, 6, 6, 4, 4, 8, 8, 8, 4, 6, 6, 4, 10, 10, 10, 6, 4, 4, 10, 10, 8, 8, 4}

```
lista = RandomInteger[{0, 1}, {100, 2}];
```

```
Cases[lista, {x_, x_}]
```

$$\begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix}$$

1	1
1	1
0	0
0	0
1	1
1	1
0	0
0	0
1	1
1	1
1	1
0	0
1	1
0	0
1	1
1	1
0	0
0	0
1	1
0	0
1	1
1	1
0	0
1	1
0	0
1	1
0	0
0	0
0	0
1	1
1	1
1	1

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$$

La **Cases** permette anche di specificare cosa fare quando si trova un elemento che soddisfa il pattern:

```
Cases[lista, {x_, x_} :> x]
```

{0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1}

Proviamo con l'approccio procedurale:

```
res = {};
Table[If[lista[[i, 1]] === lista[[i, 2]], AppendTo[res, lista[[i, 1]]]], {i, Length[lista]}];
```

Misuriamo i tempi

```
lista = RandomInteger[{0, 1}, {10^4, 2}];
```

```
AbsoluteTiming[res = Cases[lista, {x_, x_} :> x];]
```

{0.009344, Null}

```
AbsoluteTiming[
 res1 = {};
 Table[If[lista[[i, 1]] === lista[[i, 2]], AppendTo[res1, lista[[i, 1]]]], {i, Length[lista]}];]
```

{0.112522, Null}

```
res === res1
```

True

Manipolazione dei dati: alcune macro

La comodità di *Mathematica* è anche quella che dispone di numerose macro funzioni che eseguono operazioni anche complesse sui data set e dunque sulle liste. Vediamo alcune.

TakeWhile e **DropWhile**

Supponiamo da avere una lista di 100 date a partire dalla data 20 Ottobre 2011 e vogliamo selezionare solo le date fino ad un cer giorno, diciamo fino al 1° Gennaio 2012

```
(* Numero di secondi trascorsi dal 1 gennaio 1900 *)
AbsoluteTime[DateList[]]
```

$3.611509232330332 \times 10^{9}$

```
date = Table[Take[DateList[AbsoluteTime[DateList[{2011, 10, 20}]] + 86 400 * i], 3], {i, 1, 100}]
```

2011	10	21
2011	10	22
2011	10	23
2011	10	24
2011	10	25
2011	10	26
2011	10	27
2011	10	28
2011	10	29
2011	10	30
2011	10	31
2011	11	1
2011	11	2
2011	11	3
2011	11	4
2011	11	5
2011	11	6
2011	11	7

2011	11	7
2011	11	8
2011	11	9
2011	11	10
2011	11	11
2011	11	12
2011	11	13
2011	11	14
2011	11	15
2011	11	16
2011	11	17
2011	11	18
2011	11	19
2011	11	20
2011	11	21
2011	11	22
2011	11	23
2011	11	24
2011	11	25
2011	11	26
2011	11	27
2011	11	28
2011	11	29
2011	11	30
2011	12	1
2011	12	2
2011	12	3
2011	12	4
2011	12	5
2011	12	6
2011	12	7
2011	12	8

2011	12	9
2011	12	10
2011	12	11
2011	12	12
2011	12	13
2011	12	14
2011	12	15
2011	12	16
2011	12	17
2011	12	18
2011	12	19
2011	12	20
2011	12	21
2011	12	22
2011	12	23
2011	12	24
2011	12	25
2011	12	26
2011	12	27
2011	12	28
2011	12	29
2011	12	30
2011	12	31
2012	1	1
2012	1	2
2012	1	3
2012	1	4
2012	1	5
2012	1	6
2012	1	7
2012	1	8

2012	1	9
2012	1	10
2012	1	11
2012	1	12
2012	1	13
2012	1	14
2012	1	15
2012	1	16
2012	1	17
2012	1	18
2012	1	19
2012	1	20
2012	1	21
2012	1	22
2012	1	23
2012	1	24
2012	1	25
2012	1	26
2012	1	27
2012	1	28

```
TakeWhile[date, DateDifference[#, {2012, 1, 1}] > 0 &]
```

2011	10	21
2011	10	22
2011	10	23
2011	10	24
2011	10	25
2011	10	26
2011	10	27
2011	10	28
2011	10	29
2011	10	30

2011	10	31
2011	11	1
2011	11	2
2011	11	3
2011	11	4
2011	11	5
2011	11	6
2011	11	7
2011	11	8
2011	11	9
2011	11	10
2011	11	11
2011	11	12
2011	11	13
2011	11	14
2011	11	15
2011	11	16
2011	11	17
2011	11	18
2011	11	19
2011	11	20
2011	11	21
2011	11	22
2011	11	23
2011	11	24
2011	11	25
2011	11	26
2011	11	27
2011	11	28
2011	11	29
2011	11	30

$$\begin{pmatrix} 2011 & 12 & 1 \\ 2011 & 12 & 2 \\ 2011 & 12 & 3 \\ 2011 & 12 & 4 \\ 2011 & 12 & 5 \\ 2011 & 12 & 6 \\ 2011 & 12 & 7 \\ 2011 & 12 & 8 \\ 2011 & 12 & 9 \\ 2011 & 12 & 10 \\ 2011 & 12 & 11 \\ 2011 & 12 & 12 \\ 2011 & 12 & 13 \\ 2011 & 12 & 14 \\ 2011 & 12 & 15 \\ 2011 & 12 & 16 \\ 2011 & 12 & 17 \\ 2011 & 12 & 18 \\ 2011 & 12 & 19 \\ 2011 & 12 & 20 \\ 2011 & 12 & 21 \\ 2011 & 12 & 22 \\ 2011 & 12 & 23 \\ 2011 & 12 & 24 \\ 2011 & 12 & 25 \\ 2011 & 12 & 26 \\ 2011 & 12 & 27 \\ 2011 & 12 & 28 \\ 2011 & 12 & 29 \\ 2011 & 12 & 30 \\ 2011 & 12 & 31 \end{pmatrix}$$

Come fare se vogliamo, invece, le date dal 1° Gennaio in poi?

Soluzione 1: invertiamo la lista di partenza e usiamo ancora **TakeWhile** con un criterio analogo (uso ≤ al posto di >)

```
TakeWhile[Reverse[date], DateDifference[#, {2012, 1, 1}] ≤ 0 &]
```

$$\begin{pmatrix} 2012 & 1 & 28 \\ 2012 & 1 & 27 \\ 2012 & 1 & 26 \\ 2012 & 1 & 25 \\ 2012 & 1 & 24 \\ 2012 & 1 & 23 \\ 2012 & 1 & 22 \\ 2012 & 1 & 21 \\ 2012 & 1 & 20 \\ 2012 & 1 & 19 \\ 2012 & 1 & 18 \\ 2012 & 1 & 17 \\ 2012 & 1 & 16 \\ 2012 & 1 & 15 \\ 2012 & 1 & 14 \\ 2012 & 1 & 13 \\ 2012 & 1 & 12 \\ 2012 & 1 & 11 \\ 2012 & 1 & 10 \\ 2012 & 1 & 9 \\ 2012 & 1 & 8 \\ 2012 & 1 & 7 \\ 2012 & 1 & 6 \\ 2012 & 1 & 5 \\ 2012 & 1 & 4 \\ 2012 & 1 & 3 \\ 2012 & 1 & 2 \\ 2012 & 1 & 1 \end{pmatrix}$$

Oppure mi costruisco una funzione chiamata DropWhile (che non esiste). Prima devo capire come usare `LengthWhile`

```
(* calcolo la lunghezza della lista di elementi che soddisfano il criterio,
a partire dal primo *)
LengthWhile[{1, 2, 3, 4, 5, 6}, # < 4 &]
```

3

Quindi dato un criterio, posso invocare LengthWhile per capire quanti elementi soddisfano il criterio e poi tagliare tali elementi

```
Drop[{1, 2, 3, 4, 5, 6}, LengthWhile[{1, 2, 3, 4, 5, 6}, # < 4 &]]
```

{4, 5, 6}

Proviamo sulle date

```
Drop[date, LengthWhile[date, (DateDifference[#, {2012, 1, 1}] ≥ 0) &]]
```

$$\begin{pmatrix} 2012 & 1 & 2 \\ 2012 & 1 & 3 \\ 2012 & 1 & 4 \\ 2012 & 1 & 5 \\ 2012 & 1 & 6 \\ 2012 & 1 & 7 \\ 2012 & 1 & 8 \\ 2012 & 1 & 9 \\ 2012 & 1 & 10 \\ 2012 & 1 & 11 \\ 2012 & 1 & 12 \\ 2012 & 1 & 13 \\ 2012 & 1 & 14 \\ 2012 & 1 & 15 \\ 2012 & 1 & 16 \\ 2012 & 1 & 17 \\ 2012 & 1 & 18 \\ 2012 & 1 & 19 \\ 2012 & 1 & 20 \\ 2012 & 1 & 21 \\ 2012 & 1 & 22 \\ 2012 & 1 & 23 \\ 2012 & 1 & 24 \\ 2012 & 1 & 25 \\ 2012 & 1 & 26 \\ 2012 & 1 & 27 \\ 2012 & 1 & 28 \end{pmatrix}$$

Definiamo la funzione DropWhile per usarla in altri casi

```
DropWhile[data_List, crit_] := Drop[data, LengthWhile[data, crit]];
```

```
DropWhile[date, (DateDifference[#, {2012, 1, 1}] >= 0) &]
```

$$\begin{pmatrix} 2012 & 1 & 2 \\ 2012 & 1 & 3 \\ 2012 & 1 & 4 \\ 2012 & 1 & 5 \\ 2012 & 1 & 6 \\ 2012 & 1 & 7 \\ 2012 & 1 & 8 \\ 2012 & 1 & 9 \\ 2012 & 1 & 10 \\ 2012 & 1 & 11 \\ 2012 & 1 & 12 \\ 2012 & 1 & 13 \\ 2012 & 1 & 14 \\ 2012 & 1 & 15 \\ 2012 & 1 & 16 \\ 2012 & 1 & 17 \\ 2012 & 1 & 18 \\ 2012 & 1 & 19 \\ 2012 & 1 & 20 \\ 2012 & 1 & 21 \\ 2012 & 1 & 22 \\ 2012 & 1 & 23 \\ 2012 & 1 & 24 \\ 2012 & 1 & 25 \\ 2012 & 1 & 26 \\ 2012 & 1 & 27 \\ 2012 & 1 & 28 \end{pmatrix}$$

BinCount

Conta il numero di elementi inclusi in intervalli assegnati

```
punti = RandomReal[{-5, 5}, {1000, 2}];
```

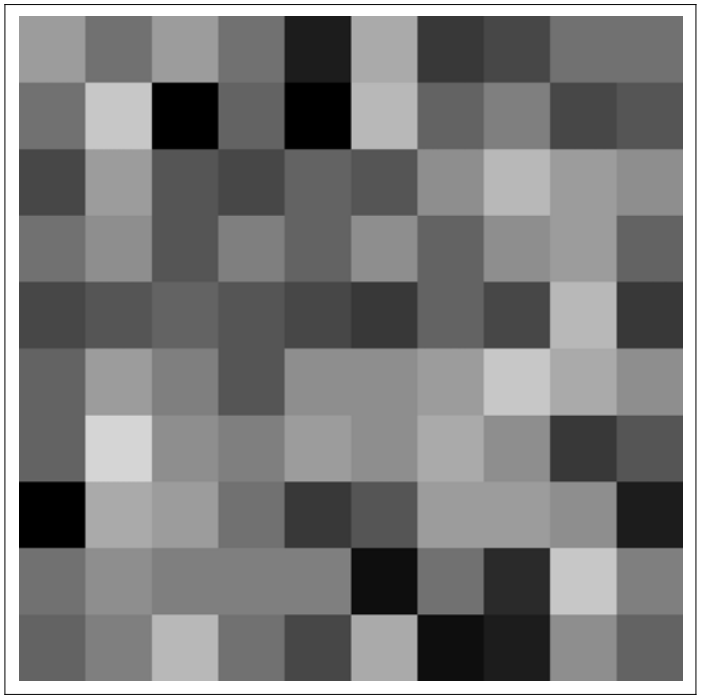
```
punti[[1 ;; 10]]
```

$$\begin{pmatrix} 4.84926 & -3.87524 \\ 2.66607 & -4.96132 \\ -0.0108596 & -0.54135 \\ 4.00817 & 2.47081 \\ 0.487482 & -4.70039 \\ -1.89368 & 3.0596 \\ -2.75624 & 1.25019 \\ 4.89099 & 4.86881 \\ -4.20549 & 2.83288 \\ 0.150837 & -3.22192 \end{pmatrix}$$

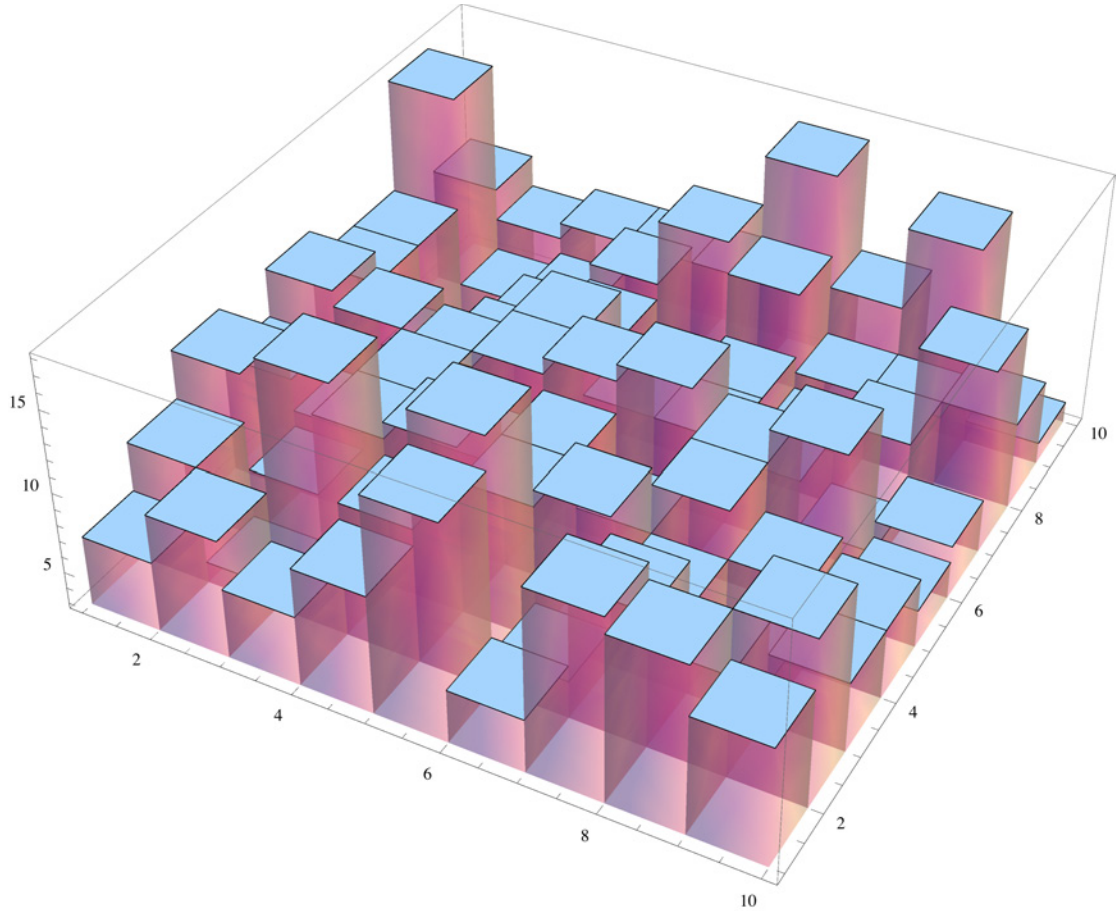
```
BinCounts[punti] // MatrixForm
```

$$\begin{pmatrix} 7 & 10 & 7 & 10 & 16 & 6 & 14 & 13 & 10 & 10 \\ 10 & 4 & 18 & 11 & 18 & 5 & 11 & 9 & 13 & 12 \\ 13 & 7 & 12 & 13 & 11 & 12 & 8 & 5 & 7 & 8 \\ 10 & 8 & 12 & 9 & 11 & 8 & 11 & 8 & 7 & 11 \\ 13 & 12 & 11 & 12 & 13 & 14 & 11 & 13 & 5 & 14 \\ 11 & 7 & 9 & 12 & 8 & 8 & 7 & 4 & 6 & 8 \\ 11 & 3 & 8 & 9 & 7 & 8 & 6 & 8 & 14 & 12 \\ 18 & 6 & 7 & 10 & 14 & 12 & 7 & 7 & 8 & 16 \\ 10 & 8 & 9 & 9 & 9 & 17 & 10 & 15 & 4 & 9 \\ 11 & 9 & 5 & 10 & 13 & 6 & 17 & 16 & 8 & 11 \end{pmatrix}$$

```
ArrayPlot[BinCounts[punti]]
```



```
ListPlot3D[BinCounts[punti], InterpolationOrder → 0, Filling → Axis, Mesh → None, ImageSize → Large]
```



Tally

```
SetDirectory[NotebookDirectory[]]
```

/Users/cgallo/Box Sync/Documenti/Universita/UNIFG/Didattica/Seminario Mathematica 2014

Conta le frequenze di ciascun elemento della lista

```
testo = Import["testo.txt"];
```

```
StringTake[testo, 191]
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget
dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

```
(* conta la frequenza di ogni singola lettera *)
Tally[Characters[testo]]
```

L	1
o	93
r	132
e	299
m	116
	424
i	232
p	63
s	191
u	229
d	64
l	136
t	184
a	171
,	63
c	103
n	144
g	33
.	64
A	8
C	8
q	27
b	28
D	8
f	16
N	10
j	4
v	27
I	4

h	14
V	6
P	8
Q	1
E	4
M	3
S	6
F	1
y	3
;	2

SortBy

Nell'esempio di prima proviamo a contare le frequenze delle parole e non delle lettere (**WordCharacter..** indica un carattere ripetuto una o più volte).

```
words = ToLowerCase[StringCases[testo, WordCharacter ..]];
```

```
Length[words]
```

425

```
Length[Union[words]]
```

143

Sort invocato su coppie di elementi utilizza il primo valore per eseguire l'ordinamento

```
res = Take[Sort[Tally[words]], -50]
```

orci	6
parturient	1
pede	4
pellentesque	2
penatibus	1
phasellus	6
placerat	1
porttitor	2
posuere	6

praesent	2
pretium	4
primis	2
pulvinar	1
purus	1
quam	4
quis	9
quisque	1
rhoncus	3
ridiculus	1
risus	1
rutrum	2
sagittis	2
sapien	3
scelerisque	1
sed	9
sem	4
semper	2
sit	6
sociis	1
sodales	2
suscipit	1
tellus	4
tempus	3
tincidunt	4
tortor	2
turpis	3
ullamcorper	2
ultrices	3
ultricies	5
ut	6

varius	1
vel	3
velit	2
venenatis	2
vestibulum	7
vitae	5
vivamus	1
viverra	2
volutpat	1
vulputate	4

SortBy permette di specificare quale elemento considerare per ordinare le coppie

```
res = Take[SortBy[Tally[words], Last], -50]
```

turpis	3
ultrices	3
vel	3
a	4
ac	4
consectetuer	4
dolor	4
dui	4
etiam	4
eu	4
faucibus	4
fringilla	4
hendrerit	4
id	4
justo	4
libero	4
lorem	4
metus	4
nullam	4

pede	4
pretium	4
quam	4
sem	4
tellus	4
tincidunt	4
vulputate	4
arcu	5
leo	5
nisi	5
ultricies	5
vitae	5
amet	6
ante	6
donec	6
ipsum	6
nec	6
nunc	6
orci	6
phasellus	6
posuere	6
sit	6
ut	6
aenean	7
et	7
in	7
vestibulum	7
eget	8
imperdiet	8
quis	9
sed	9

GatherBy

GatherBy permette di raggruppare elementi per i quali il risultato di una funzione assegnata restituisce uno stesso valore.

Automaticamente seleziona i dispari ed i pari:

```
dati = RandomInteger[{-5, 5}, {10}];
```

```
GatherBy[dati, EvenQ]
```

$$\{\{2, 0, 0, -4\}, \{-3, -1, 3, -1, 1, -1\}\}$$

Se i dati sono strutturati in liste di liste, sarebbe più difficile ancora raggrupparli in funzione di una caratteristica dei sottoelementi:

```
dati = RandomInteger[{-5, 5}, {10, 3}]
```

$$\begin{pmatrix} -5 & 3 & -1 \\ 1 & -3 & -3 \\ 1 & 3 & -5 \\ 4 & 2 & -2 \\ 1 & -4 & 1 \\ 2 & -1 & 4 \\ 3 & 1 & -4 \\ -2 & 0 & 4 \\ -5 & 4 & -3 \\ -5 & -1 & -1 \end{pmatrix}$$

Raggruppa le terne che hanno il terzo elemento dispari:

```
GatherBy[dati, EvenQ[Last[#]] &]
```

$$\left\{\begin{pmatrix} -5 & 3 & -1 \\ 1 & -3 & -3 \\ 1 & 3 & -5 \\ 1 & -4 & 1 \\ -5 & 4 & -3 \\ -5 & -1 & -1 \end{pmatrix}, \begin{pmatrix} 4 & 2 & -2 \\ 2 & -1 & 4 \\ 3 & 1 & -4 \\ -2 & 0 & 4 \end{pmatrix}\right\}$$

oppure il secondo:

```
GatherBy[dati, EvenQ[#[[2]]] &]
```

$$\left\{\begin{pmatrix} -5 & 3 & -1 \\ 1 & -3 & -3 \\ 1 & 3 & -5 \\ 2 & -1 & 4 \\ 3 & 1 & -4 \\ -5 & -1 & -1 \end{pmatrix}, \begin{pmatrix} 4 & 2 & -2 \\ 1 & -4 & 1 \\ -2 & 0 & 4 \\ -5 & 4 & -3 \end{pmatrix}\right\}$$

Conclusioni

Nonostante *Mathematica* sia un linguaggio funzionale, di fatto mette a disposizione set di funzioni dedicati ai vari approcci d programmazione

Functional Programming
Procedural Programming
Rule-Based Programming
Graph Programming
CUDA Programming

Dunque un'altra delle sua caratteristiche è che si presenta come un linguaggio **pluriparadigmatico**.