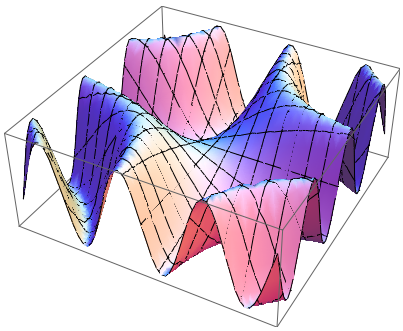
Introduzione a Mathematica

Dipartimento di Scienze Economiche, Matematiche e Statistiche
Università degli Studi di Foggia



Laboratorio per l'Analisi Quantitativa dei Dati

29 marzo 2011

dott. Crescenzio Gallo
c.gallo@unifg.it

Introduzione generale

In questa breve introduzione vedremo come interagire con *Mathematica* a livello base per eseguire alcuni semplici calcoli.

■ *Mathematica* come calcolatrice

Mathematica è in grado di eseguire le semplici operazioni aritmetiche di addizione, sottrazione, moltiplicazione e divisione. Ad es. per sommare 2+2 basta inserire il comando seguente e premere **Enter** (che si ottiene premendo contemporaneamente **Shift**+**Invio**):

```
2 + 2
```

```
4
```

Mathematica è costituito di due distinti componenti, il *Front End* ed il *Kernel*. L'input, l'output e la formattazione del testo sono gestiti dal front end. I calcoli sono effettuati nel kernel, che è un'applicazione separata. Alla prima esecuzione (anche di un semplice comando come 2+2) si potrebbe notare un leggero ritardo dovuto al tempo di attivazione del kernel; successivamente, i tempi di risposta dipendono esclusivamente dalla complessità dei calcoli richiesti al kernel.
E' possibile inserire anche commenti, racchiudendoli tra (* e *).

```
(5 * 7 - 32 / 6) / 4 (* ecco un altro complicatissimo calcolo *)
```

$$\frac{89}{12}$$

Si osservi che abbiamo ottenuto come risultato una frazione. *Mathematica* è in grado di effettuare calcoli *esatti*, piuttosto che approssimazioni decimali, quando possibile. E' comunque sempre possibile ottenere un risultato approssimato mediante la funzione **N** ("numerical"). Nel seguente esempio, **%** si riferisce al risultato precedente e **// N** passa il risultato alla funzione **N**:

```
% // N
```

```
7.41667
```

La capacità di *Mathematica* di effettuare calcoli esatti è davvero impressionante.

```
2^1000
```

```
10 715 086 071 862 673 209 484 250 490 600 018 105 614 048 117 055 336 074 437 503 883 703 510 511 249 ⋱
  361 224 931 983 788 156 958 581 275 946 729 175 531 468 251 871 452 856 923 140 435 984 577 574 698 ⋱
  574 803 934 567 774 824 230 985 421 074 605 062 371 141 877 954 182 153 046 474 983 581 941 267 398 ⋱
  767 559 165 543 946 077 062 914 571 196 477 686 542 167 660 429 831 652 624 386 837 205 668 069 376
```

Mathematica conosce anche le usuali funzioni matematiche. Nota: *i nomi degli "oggetti" nativi iniziano sempre per lettera maiuscola e gli argomenti delle funzioni sono racchiusi tra parentesi quadre.*

```
Sin[π/3]
```

$$\frac{\sqrt{3}}{2}$$

Qui viene nuovamente utilizzata la funzione **N** con un secondo argomento per ottenere la precisione desiderata per π:

```
N[π, 1000]
```

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862
 80348253421170679821480865132823066470938446095505822317253594081284811174502841027
 01938521105559644622948954930381964428810975665933446128475648233786783165271201909
 91456485669234603486104543266482133936072602491412737245870066063155881748815209209
 62829254091715364367892590360011330530548820466521384146951941511609433057270365759
 59195309218611738193261179310511854807446237996274956735188575272489122793818301194
 91298336733624406566430860213949463952247371907021798609437027705392171762931767523
 84674818467669405132000568127145263560827785771342757789609173637178721468440901224
 95343014654958537105079227968925892354201995611212902196086403441815981362977477130
 99605187072113499999983729780499510597317328160963185950244594553469083026425223082
 53344685035261931188171010003137838752886587533208381420617177669147303598253490428
 75546873115956286388235378759375195778185778053217122680661300192787661119590921642
 0199
```

Esercizio. Provare ad ottenere le prime 10 cifre decimali di e.

■ Parentesi tonde, quadre e graffe

Le parentesi hanno vari scopi in *Mathematica*. Le parentesi quadre **[]** sono usate per racchiudere gli argomenti di una funzione come in:

```
N[Tan[1]]
```

```
1.55741
```

Le parentesi tonde **()** servono a raggruppare espressioni matematiche come:

```
(2 * 5 - 1) / 3
```

```
3
```

Le parentesi graffe **{}** servono per le liste, che rivestono un importante ruolo in *Mathematica* essendo una struttura dati fondamentale. Molte funzioni possono operare su liste:

```
N[{Pi, E, Sqrt[2]}]
```

```
{3.14159, 2.71828, 1.41421}
```

Alcune funzioni restituiscono liste:

```
cifre = IntegerDigits[2^1000]
```

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

Ed alcune funzioni manipolano liste in modi interessanti. Cosa fa la seguente funzione?

```
Tally[cifre]
```

```
{{1, 34}, {0, 28}, {7, 35}, {5, 35}, {8, 30}, {6, 34}, {2, 23}, {3, 25}, {9, 23}, {4, 35}}
```

Se si desiderano ulteriori dettagli sulla funzione **Tally** basta digitare il seguente comando:

```
? Tally
```

Tally[*list*] tallies the elements in *list*, listing all distinct elements together with their multiplicities.
Tally[*list*, *test*] uses *test* to determine whether pairs of elements should be considered equivalent, and gives a list of the first representatives of each equivalence class, together with their multiplicities. ≫

Esercizio.

(a) Quante cifre ha il numero $2^{1\,000\,000}$? (Suggerimento: utilizzare il comando **Length** insieme a **IntegerDigits**).

(b) Qual è la frequenza della cifra meno ripetuta in $2^{1\,000\,000}$?

■ Algebra

Mathematica può fare molto più che semplici calcoli. Può manipolare espressioni algebriche, risolvere equazioni, calcolare derivate ed integrali e molto di più. E' anche possibile assegnare complicate espressioni a variabili per un loro successivo utilizzo. Ecco un esempio.

```
(* Assegno un valore alla variabile expr *)
```

```
expr = (x^2 - 1) (x^2 - 2)^2 (x^2 - 4)^3
```

$$(-4+x^2)^3\,(-2+x^2)^2\,(-1+x^2)$$

```
expr^2
```

$$(-4+x^2)^6\,(-2+x^2)^4\,(-1+x^2)^2$$

```
Expand[expr^2]
```

$$65\,536 - 360\,448\,x^2 + 880\,640\,x^4 - 1\,265\,664\,x^6 + 1\,193\,728\,x^8 - 779\,648\,x^{10} + 362\,128\,x^{12} - 120\,704\,x^{14} + 28\,696\,x^{16} - 4752\,x^{18} + 521\,x^{20} - 34\,x^{22} + x^{24}$$

```
Factor[%]
```

$$(-2+x)^6\,(-1+x)^2\,(1+x)^2\,(2+x)^6\,(-2+x^2)^4$$

```
Solve[expr == 0, x]
```

$$\{\{x\to -2\},\{x\to -2\},\{x\to -2\},\{x\to -1\},\{x\to 1\},\{x\to 2\},\{x\to 2\},\{x\to 2\},\{x\to -\sqrt{2}\},\{x\to -\sqrt{2}\},\{x\to \sqrt{2}\},\{x\to \sqrt{2}\}\}$$

Si osservi che **Solve** trova soluzioni *esatte.* Per ottenere un'approssimazione numerica è possibile utilizzare la funzione **NSolve**:

```
NSolve[expr == 0, x]
```

$$\{\{x\to -2.\},\{x\to -2.\},\{x\to -2.\},\{x\to -1.41421\},\{x\to -1.41421\},\{x\to -1.\},\{x\to 1.\},\{x\to 1.41421\},\{x\to 1.41421\},\{x\to 2.\},\{x\to 2.\},\{x\to 2.\}\}$$

Esercizio. Sia $p(x) = x^4 + 2\,x^3 - 8\,x - 16$.
(a) Trovare le soluzioni dell'equazione $p(x) = 0$
(b) Scomporre (fattorizzare) $p(x)$
(c) Trovare la forma espansa di $p(x)^{100}$
Che relazione esiste tra (a) e (b)?

■ Definizione di polinomi

Con *Mathematica* possiamo facilmente esaminare alcune funzioni che altrimenti non vorremmo proprio trattare. Vediamo un esempio. Consideriano la funzione polinominale f di grado 19 il cui coefficiente di x^k è il k −esimo numero primo: $f(x) = 2\,x + 3\,x^2 + 5\,x^3 + 7\,x^4 + 11\,x^5 + ... + 67\,x^{19}$ che possiamo anche abbreviare in $f(x) = \sum_{k=1}^{19} p_k\,x^k$, dove p_k denota il k −esimo numero primo.

```
f[x_] := Sum[Prime[k] x^k, {k, 1, 19}]
```

Dalla precedente definizione si deduce che la sintassi generale per la dichiarazione di una funzione è **f[x_]:=...** Il trattino di sottolineatura (**x_**) indica un *pattern*. Il comando **Prime** restituisce semplicemente il corrispondente numero primo; così, **Prime[1]=2**, **Prime[2]=3** etc. Il comando **Sum** è un po' più complesso. Osserviamo che **Sum[a[k], {k, inizio, fine}]** in *Mathematica* equivale a: $\sum_{k=\text{inizio}}^{\text{fine}} a(k)$.
Ora che abbiamo definito la funzione, possiamo utilizzarla:

```
f[1] (* restituisce la somma dei primi 19 numeri primi *)
```

```
568
```

Calcoliamone la derivata prima:

```
f'[1]
```

```
7799
```

Possiamo elevarla alla quarta potenza e svilupparla:

```
f[x]^4 // Expand
```

```
16 x^4 + 96 x^5 + 376 x^6 + 1160 x^7 + 3121 x^8 + 7532 x^9 + 16 754 x^10 + 34 796 x^11 + 68 339 x^12 + 127 952 x^13 +
  229 956 x^14 + 398 688 x^15 + 669 781 x^16 + 1 094 076 x^17 + 1 742 710 x^18 + 2 713 604 x^19 +
  4 139 111 x^20 + 6 195 712 x^21 + 9 115 304 x^22 + 13 196 800 x^23 + 18 820 889 x^24 + 26 463 844 x^25 +
  36 713 358 x^26 + 50 278 092 x^27 + 68 005 411 x^28 + 90 883 128 x^29 + 120 054 216 x^30 + 156 806 992 x^31 +
  202 573 437 x^32 + 258 909 980 x^33 + 327 475 746 x^34 + 409 985 468 x^35 + 508 161 863 x^36 +
  623 677 392 x^37 + 758 047 316 x^38 + 912 553 696 x^39 + 1 088 104 897 x^40 + 1 285 098 300 x^41 +
  1 503 370 286 x^42 + 1 742 001 524 x^43 + 1 999 279 089 x^44 + 2 272 563 568 x^45 + 2 558 435 452 x^46 +
  2 852 420 272 x^47 + 3 149 308 967 x^48 + 3 442 868 940 x^49 + 3 726 213 442 x^50 + 3 991 979 740 x^51 +
  4 232 345 581 x^52 + 4 439 472 128 x^53 + 4 605 790 984 x^54 + 4 724 319 160 x^55 + 4 788 734 419 x^56 +
  4 794 521 756 x^57 + 4 738 842 478 x^58 + 4 621 223 644 x^59 + 4 444 544 777 x^60 + 4 212 295 576 x^61 +
  3 931 161 512 x^62 + 3 608 975 120 x^63 + 3 255 458 087 x^64 + 2 879 424 164 x^65 + 2 491 892 982 x^66 +
  2 103 038 284 x^67 + 1 724 817 717 x^68 + 1 369 234 208 x^69 + 1 043 863 140 x^70 + 757 464 960 x^71 +
  514 811 139 x^72 + 318 463 596 x^73 + 171 201 482 x^74 + 73 386 172 x^75 + 20 151 121 x^76
```

Esercizio. Consideriamo la funzione: $f(x) = 1 + 2\,x + 4\,x^2 + 8\,x^3 + 16\,x^4 + ... + 2\,147\,483\,648\,x^{31}$ dove i coefficienti sono potenze di 2.
(a) Fattorizzare $f(x)$
(b) Sviluppare (trovare l'espansione di) $f(x)^{32}$
Si osservi che il termine costante non è nullo; ciò vuol dire che il comando **Sum** avrà un diverso punto di partenza.

■ Analisi

Mathematica può calcolare derivate, integrali ed essenzialmente fare tutte le manipolazioni formali richieste per completare i calcoli richiesti nell'analisi. Per calcolare le derivate si utilizza il comando **D**.

```
D[x^2, x]
```

```
2 x
```

```
D[x^2, {x, 2}]
```

```
2
```

Se definiamo una funzione f possiamo usare la notazione $f'(x)$, $f''(x)$.

```
f[x_] := x^2;
f'[x]
```

```
2 x
```

```
f''[x]
```

```
2
```

Naturalmente, *Mathematica* può calcolare derivate più complesse.

```
D[x^2 Sin[x Cos[x]], x]
```

```
x^2 Cos[x Cos[x]] (Cos[x] - x Sin[x]) + 2 x Sin[x Cos[x]]
```

Possiamo integrare questo risultato per riottenere la funzione originaria applicando il comando **Integrate**.

```
Integrate[%, x]
```

```
x^2 Sin[x Cos[x]]
```

L'integrazione è naturalmente un'attività un po' più complessa. Vi sono integrali che non possono essere espressi in forma chiusa:

```
Integrate[x^2 Sin[x Cos[x]], x]
```

$$\int x^2 \text{Sin}[x \text{Cos}[x]]\, dx$$

Quando **Integrate** non dà un risultato, come in questo caso, vuol dire che *Mathematica* non ha potuto calcolare l'integrale. La versione numerica può però venirci incontro per calcolare l'integrale definito:

```
NIntegrate[x^2 Sin[x Cos[x]], {x, 0, 1}]
```

```
0.163828
```

Esercizio. Valutare i seguenti integrali:

(a) $\int \frac{1-x^2+x^4}{1+x^3}\, dx$

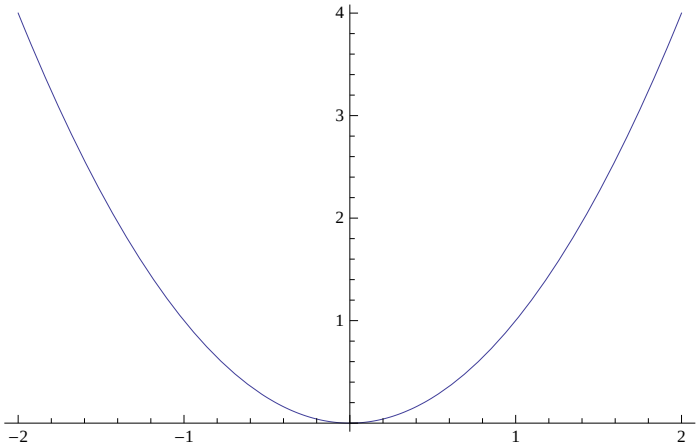
(b) $\int_{-\pi}^{\pi} \cos(\sin(\cos(x)))\, dx$

Esercizio. Trovare i punti critici della funzione $f(x) = 5\,x^6 - 168\,x^5 + 2160\,x^4 - 13\,580\,x^3 + 43\,905\,x^2 - 69\,300\,x + 1$. Cioé, trovare le radici dell'equazione $f'(x) = 0$.

■ Grafici di funzioni

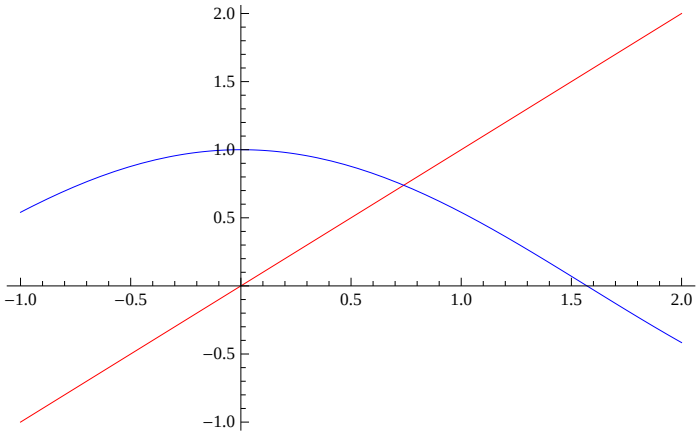
Lo strumento principale per disegnare grafici di funzioni è il comando **Plot**.

```
Plot[x^2, {x, -2, 2}]
```



Possiamo "plottare" più di una funzione contemporaneamente mediante una lista di funzioni. Ad esempio, il seguente comando illustra la soluzione dell'equazione $\cos(x) = x$.

```
Plot[{x, Cos[x]}, {x, -1, 2}, PlotStyle → {Red, Blue}]
```

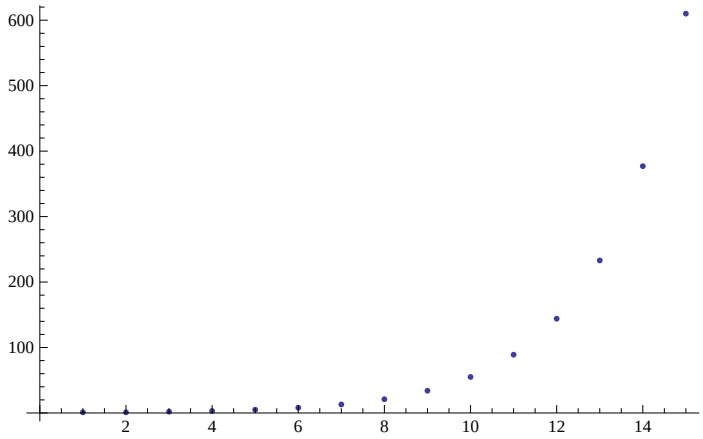


Questo comando illustra un primo uso di un'opzione, cioé **PlotStyle->{Red,Blue}** in modo da poter distinguere i grafici. Vi sono molte opzioni grafiche e molti comandi grafici. Esaminiamo ad esempio il comando **ListPlot**, un comando per grafica discreta. Data una lista di numeri, **ListPlot** ne traccia il grafico rispetto alla loro posizione nella lista. Ad esempio, generiamo i primi numeri della successione di Fibonacci:

```
fibs = Table[Fibonacci[k], {k, 1, 15}]
```

```
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610}
```

```
ListPlot[fibs]
```



Esercizio. Data $f(x) = x \sin\left(x^2\right)$ tracciare i grafici di $f(x)$ ed $f'(x)$ sugli stessi assi.

La notazione

Mathematica è un enorme programma ed apprenderlo tutto può essere un'impresa ardua. Comunque, un aspetto positivo di *Mathematica* è la consistenza del suo progetto. Consideriamo il seguente insieme di comandi, tutti validi.

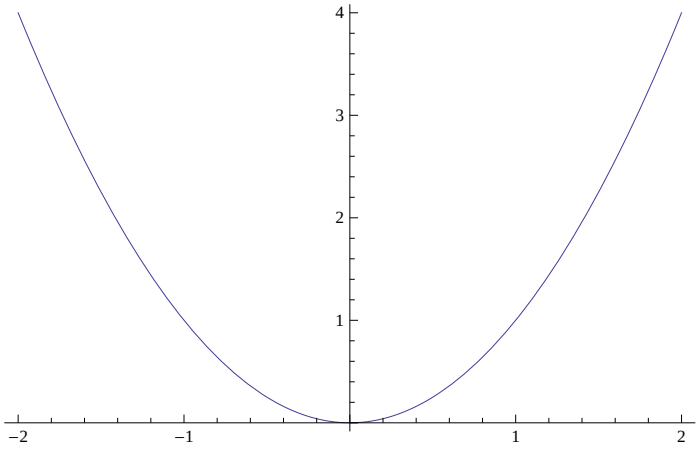
```
Integrate[x^2, {x, -2, 2}]
```

$$\frac{16}{3}$$

```
NIntegrate[x^2, {x, -2, 2}]
```

```
5.33333
```

```
Plot[x^2, {x, -2, 2}]
```




```
Table[x^2, {x, -2, 2}]
```

```
{4, 1, 0, 1, 4}
```

```
Sum[x^2, {x, -2, 2}]
```

```
10
```

```
Product[x^2, {x, -2, 2}]
```

```
0
```

```
Do[x^2, {x, -2, 2}]
```

Osserviamo che i comandi su esposti (ed altri oggetti predefiniti) obbediscono allo stesso schema. Inoltre, ciascun comando ha la stessa forma: **Comando[***espressione***,** *iteratore***]**. Mantenere questa consistenza rende alla lunga il linguaggio più semplice da apprendere e da utilizzare.

Calcolo numerico e simbolico

La prima versione di *Mathematica* fu chiamata “un sistema per fare matematica con il computer”. Come tale, il calcolo numerico e simbolico è alla sua base. Per calcolo numerico di solito intendiamo semplici operazioni aritmetiche. Ad esempio:

```
2 + 2
```

```
4
```

```
5.73 * 42
```

```
240.66
```

Come vedremo, vi è una distinzione importante tra questi due esempi a dispetto della loro (apparente) semplicità.

■ Alcuni concetti fondamentali

■ Tipi di numeri: 1 e 1.0

Mathematica può effettuare due principali tipi di calcolo numerico: *esatto* ed *approssimato*. Interi, frazioni e costanti matematiche come π ed e sono numeri esatti. Quando si eseguono calcolo con numeri esatti, *Mathematica* restituisce risultati esatti. Ad esempio:

```
2/3
```

$$\frac{2}{3}$$

Si osservi che il risultato è 2/3. Sulla maggior parte delle calcolatrici ci si aspetterebbe 0.666667. Ogni volta che *Mathematica* esegue un calcolo, lo fa alla massima precisione possibile. I numeri esatti effettivamente hanno una precisione infinita. L'approssimazione decimale ha la precisione della macchina; il seguente calcolo è più da "calcolatrice":

```
2 / 3.0
```

```
0.666667
```

Ha senso? La massima precisione possibile è stata limitata dal numero in rappresentazione decimale approssimata (formato macchina) 3.0. Ecco una lista di esempi simili.

```
lista = {(1 - 4)/2, (1 - 4)/2.0, 4 + π, 4.0 + π, Sin[π/3], Sin[π/3.0]}
```

$$\left\{-\frac{3}{2}, -1.5, 4+\pi, 7.14159, \frac{\sqrt{3}}{2}, 0.866025\right\}$$

Al comando **N** può essere passato qualsiasi numero o lista di numeri per generare un'approssimazione numerica:

```
N[lista]
```

```
{-1.5, -1.5, 7.14159, 7.14159, 0.866025, 0.866025}
```

Possiamo anche utilizzare il secondo argomento opzionale di **N** per ottenere la precisione desiderata.

```
N[lista, 20]
```

```
{-1.5000000000000000000, -1.5, 7.1415926535897932385,
 7.14159, 0.86602540378443864676, 0.866025}
```

Perché le coppie di risultati non sono più le stesse?

■ Manipolazione algebrica

Mathematica può eseguire ben più che semplici calcoli con numeri; può anche manipolare simboli ed espressioni più complesse. Ecco un semplice esempio:

```
Expand[(1 + x)^5]
```

$1 + 5\,x + 10\,x^2 + 10\,x^3 + 5\,x^4 + x^5$

Questa capacità di eseguire manipolazioni simboliche è in realtà la chiave per l'aritmetica esatta. Lo si può capire meglio dal seguente esempio:

```
Expand[(1 + π)^5]
```

$1 + 5\,\pi + 10\,\pi^2 + 10\,\pi^3 + 5\,\pi^4 + \pi^5$

Il senso dovrebbe essere chiaro... La controparte di **Expand** è **Factor**.

```
Factor[%]
```

$(1 + \pi)^5$

Probabilmente la manipolazione algebrica più importante è quella realizzata dal comando **Simplify**. Per illustrarlo, produciamo prima un'orribile espressione attraverso il comando **TrigExpand**.

```
TrigExpand[Cos[5 x]]
```

$\text{Cos}[x]^5 - 10\,\text{Cos}[x]^3\,\text{Sin}[x]^2 + 5\,\text{Cos}[x]\,\text{Sin}[x]^4$

Simplify, in questo caso, ritorna all'espressione compatta:

```
Simplify[%]
```

$\text{Cos}[5\,x]$

Occasionalmente, può essere più utile semplificare un'espressione con il comando **FullSimplify**:

```
expr = Cosh[x] - Sinh[x];
{Simplify[expr], FullSimplify[expr]}
```

$\{\text{Cosh}[x] - \text{Sinh}[x],\ e^{-x}\}$

Quindi la manipolazione algebrica può risultare complicata, anche con un sistema di algebra computazionale su computer. Avere un tale sistema a disposizione semplicemente innalza il livello dei problemi che tendiamo ad affrontare. Ecco un esempio (un po' surreale, per la verità):

```
expr = ArcCsch[2] / (ArcSinh[2] - ArcCsch[2]) /
  Sin[Pi ArcCsch[2] / (ArcSinh[2] - ArcCsch[2])]
```

$(\text{ArcCsch}[2]\,\text{Csc}[(\pi\,\text{ArcCsch}[2]) / (-\text{ArcCsch}[2] + \text{ArcSinh}[2])]) / (-\text{ArcCsch}[2] + \text{ArcSinh}[2])$

Cosa potrebbe rappresentare questo "mostro"? Controlliamone il valore numerico:

```
N[expr]
```

```
0.5
```

Vi è un forte indizio che il risultato *potrebbe* essere 1/2. Controlliamo con una maggiore precisione:

```
N[expr, 50]
```

```
0.50000000000000000000000000000000000000000000000000
```

Non è così facile da provare, comunque. Infatti:

```
FullSimplify[expr]
```

$(\text{ArcCsch}[2]\,\text{Csc}[(\pi\,\text{ArcCsch}[2]) / (\text{ArcCsch}[2] - \text{ArcSinh}[2])]) / (\text{ArcCsch}[2] - \text{ArcSinh}[2])$

Per questo particolare problema la semplificazione può essere tentata dopo aver provato con la sostituzione esplicita **ArcSinh[2]→3*ArcCsch[2]**. Verifichiamo prima che la sostituzione ipotizzata sia valida:

```
FullSimplify[ArcSinh[2] - 3 * ArcCsch[2]]
```

```
0
```

Ora possiamo utilizzare una regola per effettuare la sostituzione e *quindi* eseguire la semplificazione.

```
FullSimplify[expr /. ArcSinh[2] → 3 * ArcCsch[2]]
```

$$\frac{1}{2}$$

```
?/.
```

expr /. *rules* applies a rule or list of rules in an attempt to transform each subpart of an expression *expr*. ≫

Esercizio. $e^{\pi\sqrt{163}}$ può essere espresso come un numero intero di 18 cifre. Verificarlo mediante **N**.

Esercizio. Provare a semplificare la seguente espressione:

$$\tan^{-1}\left(3 + 2\sqrt{2} + \sqrt{14 + 10\sqrt{2}}\right) + \frac{1}{2}\cot^{-1}\left(1 + \sqrt{2} + \sqrt{4 + 4\sqrt{2}}\right)$$

Suggerimento: utilizzare **FullSimplify** insieme a **TrigToExp** (leggere l'help, se necessario).

■ Risoluzione algebrica e numerica di equazioni

Abbiamo già esaminato brevemente i comandi **Solve** ed **NSolve**. Ora li approfondiremo, insieme al comando **FindRoot** ad essi correlato.
Solve è un comando puramente algebrico che trova soluzioni di un limitato insieme di equazioni. Funziona molto bene su polinomi di grado basso.

```
p = x^2 - x - 1;
soluzioni = Solve[p == 0, x]
```

$$\left\{\left\{x \to \frac{1}{2}\left(1-\sqrt{5}\right)\right\}, \left\{x \to \frac{1}{2}\left(1+\sqrt{5}\right)\right\}\right\}$$

Le soluzioni vengono restituite in termini di regole. Una regola è della forma *variabile* → *sostituzione*. Una regola può essere applicata ad un'espressione mediante l'operatore di sostituzione (**/.**). Ad esempio:

```
p /. soluzioni
```

$$\left\{-1+\frac{1}{4}\left(1-\sqrt{5}\right)^2+\frac{1}{2}\left(-1+\sqrt{5}\right), -1+\frac{1}{2}\left(-1-\sqrt{5}\right)+\frac{1}{4}\left(1+\sqrt{5}\right)^2\right\}$$

Strano: il risultato dovrebbe essere zero, essendo le radici del polinomio! Semplifichiamo:

```
Simplify[%]
```

```
{0, 0}
```

Ok. Proviamo ora un'equazione cubica:

```
Solve[x^3 - x - 1 == 0, x]
```

$$\left\{\left\{x \to \frac{1}{3}\left(\frac{27}{2}-\frac{3\sqrt{69}}{2}\right)^{1/3}+\frac{\left(\frac{1}{2}\left(9+\sqrt{69}\right)\right)^{1/3}}{3^{2/3}}\right\},$$

$$\left\{x \to -\frac{1}{6}\left(1+i\sqrt{3}\right)\left(\frac{27}{2}-\frac{3\sqrt{69}}{2}\right)^{1/3}-\left(\left(1-i\sqrt{3}\right)\left(\frac{1}{2}\left(9+\sqrt{69}\right)\right)^{1/3}\right)\Big/\left(2\times 3^{2/3}\right)\right\},$$

$$\left\{x \to -\frac{1}{6}\left(1-i\sqrt{3}\right)\left(\frac{27}{2}-\frac{3\sqrt{69}}{2}\right)^{1/3}-\left(\left(1+i\sqrt{3}\right)\left(\frac{1}{2}\left(9+\sqrt{69}\right)\right)^{1/3}\right)\Big/\left(2\times 3^{2/3}\right)\right\}\Big\}$$

Il risultato è abbastanza complesso, come può spesso capitare anche per semplici espressioni algebriche. In questi casi è meglio provare con il comando **NSolve**, che utilizza una combinazione di funzioni numeriche ed algebriche per trovare approssimazioni numeriche alle soluzioni dell'equazione.

```
NSolve[x^3 - x - 1 == 0, x]
```

```
{{x → -0.662359 - 0.56228 i}, {x → -0.662359 + 0.56228 i}, {x → 1.32472}}
```

Ora il risultato è decisamente più leggibile. Non sempre però questa strada è praticabile. Ad esempio:

```
Solve[Cos[x] == x, x]
```

```
Solve::nsmet : This system cannot be solved with the methods available to Solve. ≫
```

```
Solve[Cos[x] == x, x]
```

Provare con **NSolve** non cambia la situazione:

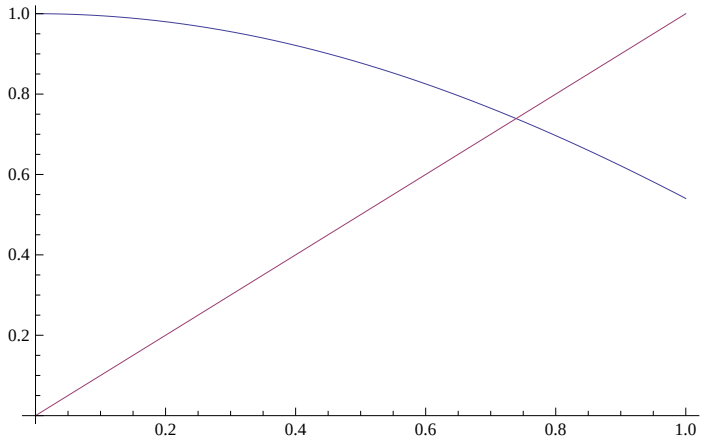
```
NSolve[Cos[x] == x, x]
```

```
NSolve::nsmet : This system cannot be solved with the methods available to NSolve. ≫
```

```
NSolve[Cos[x] == x, x]
```

E' però facile vedere che c'è una soluzione dal semplice esame del grafico seguente:

```
Plot[{Cos[x], x}, {x, 0, 1}]
```



Il punto di intersezione indica una soluzione, poco più grande di 0.7. In questa situazione, la funzione puramente numerica **FindRoot** può essere di aiuto.

La sintassi è **`FindRoot[`***eq*`, {`*var*`, `*init*`}]`, dove *eq* è l'equazione nella variabile *var* rispetto alla quale risolvere. Il parametro *init* è un valore iniziale che dovrebbe essere vicino alla soluzione; tipicamente tale valore viene trovato mediante un grafico come quello visto. **FindRoot** utilizza quindi il metodo di Newton/Raphson per arrivare alla soluzione. Ad esempio:

```
FindRoot[Cos[x] == x, {x, 0.7}]
```

```
{x → 0.739085}
```

Questa è una procedura completamente numerica, quindi il risultato sarà sempre fornito in forma decimale approssimata.

Esercizio. Si consideri l'equazione $x \sin\left(x^3\right) = e^{-x^2}$.

(a) Disegnare il grafico di entrambe le funzioni sull'intervallo $\left[0, \sqrt[3]{6\,\pi}\right]$. Si dovrebbero osservare 6 punti di intersezione.

(b) Utilizzare **FindRoot** con i valori iniziali ottenuti dal grafico precedente per trovare buone approssimazioni decimali alle tre più piccole soluzioni positive dell'equazione.

■ Ancora un po' di analisi

Ricordiamo che è possibile definire una funzione, differenziarla e trovarne i punti critici:

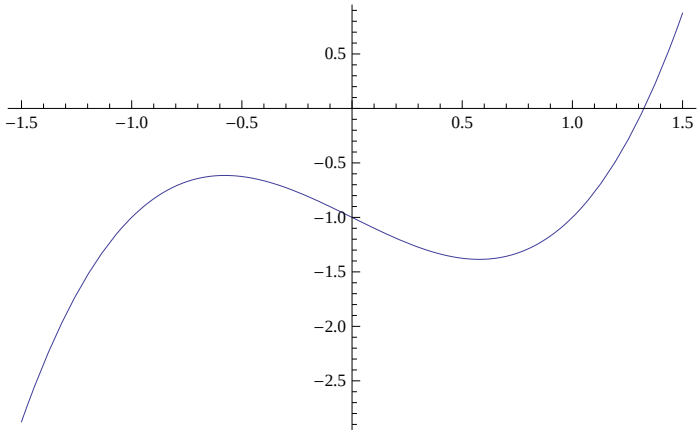
```
f[x_] := x^3 - x - 1;
Solve[f'[x] == 0, x]
NSolve[f'[x] == 0, x]
```

$$\left\{\left\{x \to -\frac{1}{\sqrt{3}}\right\}, \left\{x \to \frac{1}{\sqrt{3}}\right\}\right\}$$

```
{{x → -0.57735}, {x → 0.57735}}
```

Queste dovrebbero essere le ascisse dei due estremi, che possiamo visualizzare come segue:

```
Plot[f[x], {x, -1.5, 1.5}]
```



Esercizio. Utilizzare una combinazione di tecniche grafiche ed algebriche per trovare i due più piccoli valori positivi di $f(x) = x\sin(x^2)$. Procedere osservando il grafico della funzione e determinare (possibilmente in forma approssimata) dove la derivata della funzione è nulla.
Nota: Avrai probabilmente bisogno di usare **FindRoot** per questo problema, dato che l'equazione risultante non può essere risolta per via algebrica. Avrai quindi necessità di tracciare prima il grafico con il comando **Plot** per ottenere il valore iniziale.

Le liste

In questa sezione tratteremo la manipolazione di liste, la struttura dati composta fondamentale. Iniziamo con un breve sguardo ad alcune interessanti applicazioni che si possono realizzare con le liste.

■ Generazione e manipolazione di liste

■ Generare liste

Anche se è possibile scrivere una lista utilizzando le parentesi graffe nella forma `{a,...}`, è certamente più efficiente generare una lunga lista per mezzo di comandi. Il più utile è `Table`, che ha la seguente sintassi: **Table[a_i, {*i*, min, max}]**. Calcoliamo ad esempio i primi 100 quadrati:

```
Table[k^2, {k, 1, 100}]
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400,
 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225,
 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304,
 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 3600, 3721,
 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476,
 5625, 5776, 5929, 6084, 6241, 6400, 6561, 6724, 6889, 7056, 7225, 7396, 7569,
 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801, 10000}
```

Se invece vogliamo solo i quadrati dei numeri dispari utilizzeremo un passo 2:

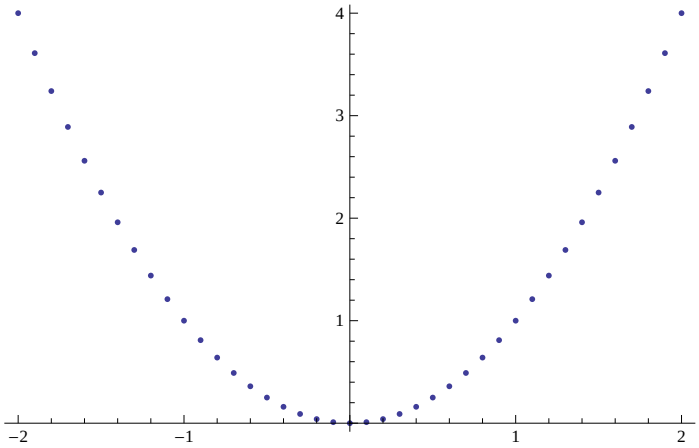
```
Table[k^2, {k, 1, 100, 2}]
```

```
{1, 9, 25, 49, 81, 121, 169, 225, 289, 361, 441, 529, 625, 729,
 841, 961, 1089, 1225, 1369, 1521, 1681, 1849, 2025, 2209, 2401, 2601,
 2809, 3025, 3249, 3481, 3721, 3969, 4225, 4489, 4761, 5041, 5329, 5625,
 5929, 6241, 6561, 6889, 7225, 7569, 7921, 8281, 8649, 9025, 9409, 9801}
```

Osserviamo che gli elementi della lista possono a loro volta essere delle liste o espressioni più complesse. Inoltre, il passo non deve necessariamente essere un numero intero. In quest'altro esempio generiamo una lista di 41 punti sul grafico di $y = x^2$ e passiamo il risultato a **ListPlot**.

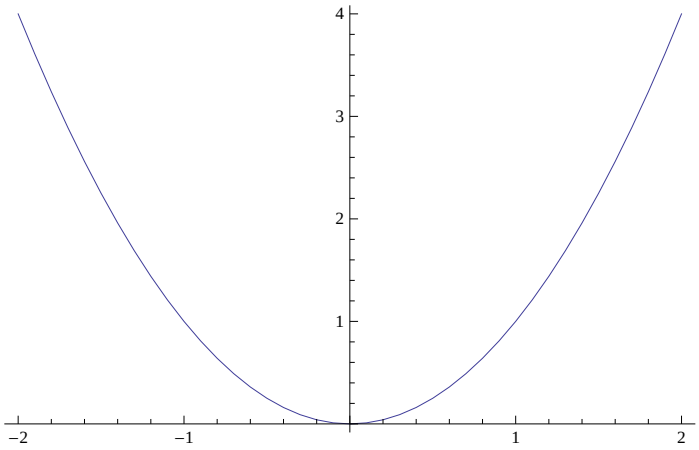
```
punti = Table[{x, x^2}, {x, -2, 2, 0.1}]
ListPlot[punti]
```

```
{{-2., 4.}, {-1.9, 3.61}, {-1.8, 3.24}, {-1.7, 2.89}, {-1.6, 2.56}, {-1.5, 2.25},
 {-1.4, 1.96}, {-1.3, 1.69}, {-1.2, 1.44}, {-1.1, 1.21}, {-1., 1.}, {-0.9, 0.81},
 {-0.8, 0.64}, {-0.7, 0.49}, {-0.6, 0.36}, {-0.5, 0.25}, {-0.4, 0.16},
 {-0.3, 0.09}, {-0.2, 0.04}, {-0.1, 0.01}, {0., 0.}, {0.1, 0.01}, {0.2, 0.04},
 {0.3, 0.09}, {0.4, 0.16}, {0.5, 0.25}, {0.6, 0.36}, {0.7, 0.49}, {0.8, 0.64},
 {0.9, 0.81}, {1., 1.}, {1.1, 1.21}, {1.2, 1.44}, {1.3, 1.69}, {1.4, 1.96},
 {1.5, 2.25}, {1.6, 2.56}, {1.7, 2.89}, {1.8, 3.24}, {1.9, 3.61}, {2., 4.}}
```



Per vedere la linea che congiunge i punti, si può utilizzare **ListLinePlot**:

```
ListLinePlot[punti]
```



- **Funzioni che operano su liste**

Andremo ora ad eseguire alcuni esercizi operando sulla seguente lista:

```
lista = Table[i, {i, 1, 21}]
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21}
```

Molte funzioni accettano una lista come argomento. La statistica è piena di queste funzioni. Ad esempio, potremmo voler calcolare la media della lista:

```
Mean[lista]
```

```
11
```

Analogamente per i comandi **Median**, **Variance** e **StandardDeviation**:

```
Median[lista]
```

```
11
```

```
Variance[lista]
```

$$\frac{77}{2}$$

```
StandardDeviation[lista]
```

$$\sqrt{\frac{77}{2}}$$

E' anche semplice sommare o invertire gli elementi della lista:

```
Total[lista]
```

```
231
```

```
Reverse[lista]
```

```
{21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
```

- **Utilizzo di funzioni pure con Map e Select**

Molte funzioni operano naturalmente su liste piuttosto che numeri. Ad esempio, per calcolare i quadrati degli elementi della lista possiamo semplicemente scrivere:

```
lista^2
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441}
```

Anche funzioni molto più complesse operano allo stesso modo. Ad esempio, la funzione **EvenQ** restituisce il valore **True** (vero) o **False** (falso) a seconda se il numero in input è pari o dispari. Se l'input è una lista di numeri, **EvenQ** restituisce una lista di valori logici:

```
EvenQ[lista]
```

```
{False, True, False, True, False, True, False, True, False, True,
 False, True, False, True, False, True, False, True, False, True, False}
```

E se invece abbiamo una funzione più complessa che non si applica automaticamente alla lista? Consideriamo la seguente funzione:

```
f[n_] := If[EvenQ[n], n/2, 3 n + 1]
```

f divide un numero pari per 2 e lo trasforma in $3n + 1$ se è dispari. Se proviamo **f[lista]** non produce alcun risultato sensato:

```
f[lista]
```

```
If[{False, True, False, True, False, True, False, True, False, True,
   False, True, False, True, False, True, False, True, False, True, False},
 1/2 {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21},
 3 {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21} + 1]
```

Possiamo invece "mappare" la funzione sulla lista attraverso l'operatore **Map** (**/@**). Vi sono due forme equivalenti: **Map[f,lista]** e **f/@lista**. Proviamo:

```
f /@ lista
```

```
{4, 1, 10, 2, 16, 3, 22, 4, 28, 5, 34, 6, 40, 7, 46, 8, 52, 9, 58, 10, 64}
```

Talvolta è utile mappare una funzione su una lista senza creare esplicitamente una nuova funzione. Allo scopo si utilizzano le cosiddette funzioni *pure* (o anonime). Una funzione pura ha la forma **corpo&**, dove **corpo** è un'espressione contenente il simbolo **#** e **&** indica che quest'espressione è una funzione. Ad esempio, **#^2&** rappresenta la funzione quadrato:

```
#^2 &[5]
```

```
25
```

Può sembrare criptico, ma è spesso utile. Possiamo quindi calcolare i quadrati degli elementi della nostra listra semplicemente con:

```
#^2 &[lista]
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441}
```

oppure con:

```
#^2 & /@ lista
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441}
```

o equivalentemente:

```
Map[#^2 &, lista]
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441}
```

Anche la precedente funzione **f** può essere trasformata allo stesso modo:

```
If[EvenQ[#], #/2, 3 # + 1] & /@ lista
```

```
{4, 1, 10, 2, 16, 3, 22, 4, 28, 5, 34, 6, 40, 7, 46, 8, 52, 9, 58, 10, 64}
```

Le funzioni pure sono anche convenienti quando occorre selezionare gli elementi di una lista che soddisfano determinati criteri. La funzione **Select** ha la forma **Select[lista,test]** dove **test** rappresenta una funzione booleana che descrive i criteri da soddisfare. Ecco come selezionare gli elementi pari della nostra lista:

```
Select[lista, EvenQ]
```

```
{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
```

Nel caso che la funzione-criterio non sia già disponibile è sempre possibile costruirne una al volo. Vediamo ad esempio come selezionare gli elementi della lista divisibili per tre:

```
Select[lista, Mod[#, 3] == 0 &]
```

```
{3, 6, 9, 12, 15, 18, 21}
```

Possiamo contare quanti sono con la funzione **Length**:

```
Length[%]
```

```
7
```

- **Un esempio più complesso**

Si supponga di voler trovare la somma di tutti i multipli di 3 o 5 inferiori a 1000. Elenchiamo prima i numeri che ci interessano (senza farli visualizzare):

```
lista = Table[i, {i, 1, 999}];
```

Poi selezioniamo solo quelli che sono divisibili per 3 o per 5 (l'operatore **||** sta per "oppure"):

```
multipli = Select[lista, Mod[#, 3] == 0 || Mod[#, 5] == 0 &];
```

Ed infine li sommiamo:

```
Total[multipli]
```

```
233 168
```

Spesso risulta più utile combinare tutti i passi in uno solo:

```
Total[Select[Table[i, {i, 1, 999}], Mod[#, 3] == 0 || Mod[#, 5] == 0 &]]
```

```
233 168
```

Esercizio. Trovare la somma di tutti i termini di valore pari della successione di Fibonacci che non superi 4 000 000. Utilizzare allo scopo la funzione **Fibonacci[n]** per ottenere l'*n* −mo termine della successione.

Esercizio. Un palindromo è un numero le cui cifre si leggono allo stesso modo da entrambi i lati. Ad esempio, 101 e 29 792 sono palindromi, mentre 42 non lo è. Si utilizzino le funzioni **IntegerDigits** e **Reverse** per scrivere una funzione che controlli se un numero è palindromo. Quanti palindromi vi sono tra 42 e 42 000?

Grafici “a ragnatela”

Ancora sulla manipolazione di liste

Utilizziamo ancora la lista:

```
lista = Table[i, {i, 1, 21}]
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21}
```

Il comando **Partition** suddivide una lista in parti di uguale lunghezza. Ad esempio:

```
Partition[lista, 3]
```

```
{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}, {13, 14, 15}, {16, 17, 18}, {19, 20, 21}}
```

Il comando **Flatten** fa il contrario, nel senso che rimuove tutte le sotto-liste:

```
Flatten[%]
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21}
```

Possiamo rimescolare il tutto mediante il comando **Transpose**:

```
Transpose[Partition[lista, 3]]
```

```
{{1, 4, 7, 10, 13, 16, 19}, {2, 5, 8, 11, 14, 17, 20}, {3, 6, 9, 12, 15, 18, 21}}
```

```
Flatten[Transpose[Partition[lista, 3]]]
```

```
{1, 4, 7, 10, 13, 16, 19, 2, 5, 8, 11, 14, 17, 20, 3, 6, 9, 12, 15, 18, 21}
```

Le funzioni *List

Si provi a digitare ?*List:

```
? *List
```

▼ System`

BinaryReadList	FixedPointList	PowerModList
CoefficientList	FoldList	PropertyList
ComposeList	HistogramList	ReadList
DateList	List	ReplaceList
DMSList	MessageList	SampledSoundList
EdgeList	MonomialList	SingularValueList
FactorList	NestList	StringReplaceList
FactorSquareFreeList	NestWhileList	TrigFactorList
FactorTermsList	ParentList	VertexList
FindList	PermutationList	$MessageList

Il comando genera un elenco di funzioni, molte delle quali generano liste. **CoefficientList** ad esempio genera la lista dei coefficienti di un polinomio:

```
CoefficientList[x^5 - 2 x^4 + 3 x^3 - 4 x^2 + 5 x - 6, x]
```

```
{-6, 5, -4, 3, -2, 1}
```

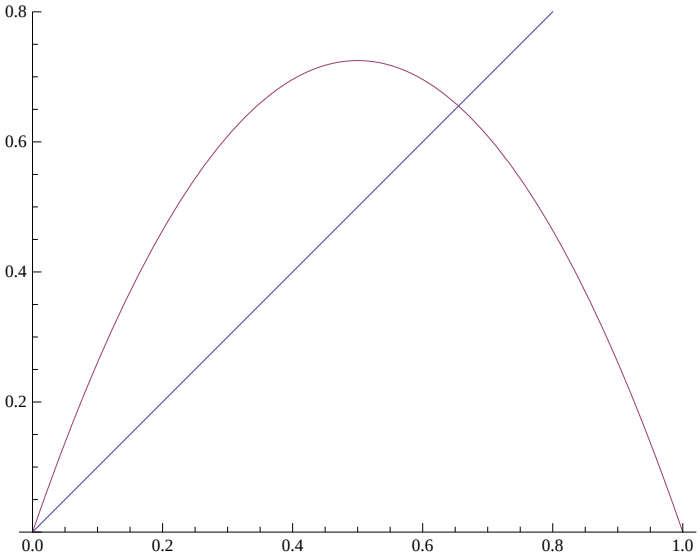
```
? NestList
```

NestList[*f*, *expr*, *n*] gives a list of the results of applying *f* to *expr* 0 through *n* times. ≫

Costruzione di un grafico “a ragnatela”

Iniziamo l’esercizio con la definzione di una funzione f e la visualizzazione del suo grafico insieme a quello della retta $y = x$.

```
f[x_] := 2.9 x (1 - x);
grafici = Plot[{x, f[x]}, {x, 0, 1}, AspectRatio → Automatic, PlotRange → {0, 0.8}]
```



Ora utilizziamo il comando **NestList** per generare la traiettoria. Genereremo solo alcuni termini per osservarne il comportamento.

```
orbit = NestList[f, 0.1, 5]
```

```
{0.1, 0.261, 0.559349, 0.714785, 0.591215, 0.700871}
```

La traiettoria generata comprende i singoli punti, ma abbiamo bisogno della sequenza $\{(x_0, x_0), (x_0, x_1), (x_1, x_1), (x_1, x_2), ...\}$. Si osservi che occorrono quattro copie di ciascun termine x_i tranne il primo. Abbiamo allora bisogno di partizionare la lista in sottoliste lunghe 2.
Prima di tutto, quadruplichiamo ciascun termine mediante il mapping di una funzione pura:

```
{#, #, #, #} & /@ orbit
```

```
{{0.1, 0.1, 0.1, 0.1}, {0.261, 0.261, 0.261, 0.261},
 {0.559349, 0.559349, 0.559349, 0.559349}, {0.714785, 0.714785, 0.714785, 0.714785},
 {0.591215, 0.591215, 0.591215, 0.591215}, {0.700871, 0.700871, 0.700871, 0.700871}}
```

Ora rimuoviamo le sottoliste:

```
Flatten[%]
```

```
{0.1, 0.1, 0.1, 0.1, 0.261, 0.261, 0.261, 0.261, 0.559349, 0.559349,
 0.559349, 0.559349, 0.714785, 0.714785, 0.714785, 0.714785, 0.591215,
 0.591215, 0.591215, 0.591215, 0.700871, 0.700871, 0.700871, 0.700871}
```

Cancelliamo il primo termine:

```
Drop[%, 1]
```

```
{0.1, 0.1, 0.1, 0.261, 0.261, 0.261, 0.261, 0.559349, 0.559349,
 0.559349, 0.559349, 0.714785, 0.714785, 0.714785, 0.714785, 0.591215,
 0.591215, 0.591215, 0.591215, 0.700871, 0.700871, 0.700871, 0.700871}
```

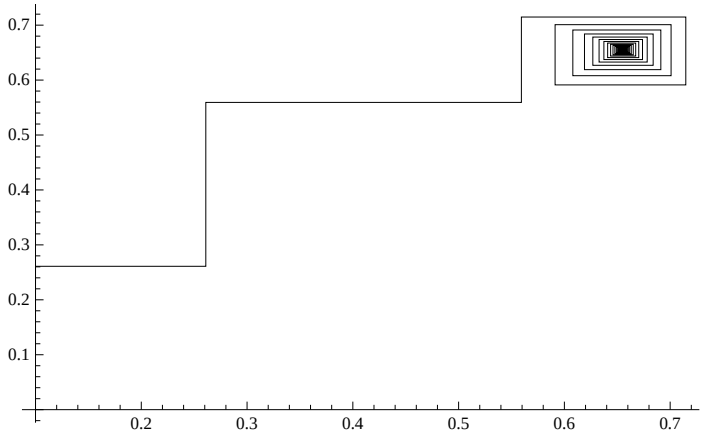
E infine partizioniamo a coppie il risultato:

```
Partition[%, 2]
```

```
{{0.1, 0.1}, {0.1, 0.261}, {0.261, 0.261}, {0.261, 0.559349}, {0.559349, 0.559349},
 {0.559349, 0.714785}, {0.714785, 0.714785}, {0.714785, 0.591215},
 {0.591215, 0.591215}, {0.591215, 0.700871}, {0.700871, 0.700871}}
```

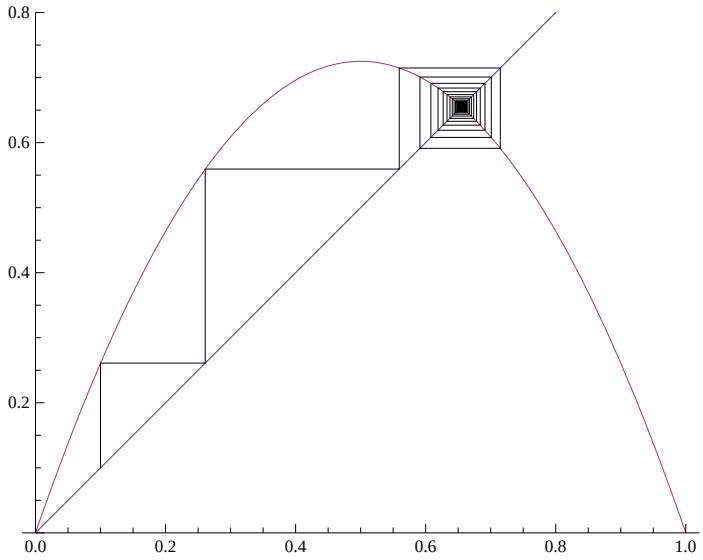
Questa sequenza di punti forma la linea che dobbiamo tracciare per generare il grafico “a ragnatela”. Prima di questo, combiniamo il tutto in un unico comando e generiamo tutti i dati.

```
linee = Partition[Drop[Flatten[{#, #, #, #} & /@ NestList[f, 0.1, 100]], 1], 2];
ragnatela = ListLinePlot[linee, PlotRange → All, PlotStyle → Black]
```



Infine visualizziamo il tutto:

```
Show[grafici, ragnatela]
```



Esercizio. In questo esercizio tracceremo alcuni punti generati con una procedura iterativa nel piano complesso. Per fare ciò abbiamo bisogno di estrarre da un numero complesso $a + b\,i$ la coppia dei suoi termini reali (a, b). Le funzioni **Re** e **Im** estraggono la parte reale ed immaginaria. Ad esempio:

```
Im[1 + 2 i]
```

```
2
```

```
Re[1 + 2 i]
```

```
1
```

(a) Utilizzare **Re** e **Im** per scrivere una funzione che trasformi un numero complesso in una coppia di numeri reali. Mappare quindi la funzione sulla lista **{1,i,1-i}** per verificarne il funzionamento. Il risultato dovrebbe essere **{{1,0},{0,1},{1,-1}}**.

(b) Ora, sia $f(z) = z^2$ e si generi la traiettoria ottenuta iterando f 500 volte partendo da $z_0 = e^i$ (*Mathematica* eseguirà tutti i calcoli necessari!).

(c) Mappare la funzione del punto (a) sulla traiettoria del punto (b) e passare il risultato a **ListLinePlot**. Il risultato finale dovrebbe essere piacevole da osservare...

Importazione, esportazione, consultazione di dati

Molte applicazioni matematiche, statistiche e scientifiche in genere richiedono analisi di dati. Senza la capacità di leggere e scrivere dati da e verso l'esterno, gli strumenti di *Mathematica* sarebbero inutili per l'analisi dei dati. Come ogni buon linguaggio di programmazione, naturalmente, *Mathematica* possiede una varietà di strumenti per questo scopo.

- **Importazione**
- **Importare dati da Internet**

Il comando **Import** è un utile strumento ad alto livello per introdurre dati nell'ambiente *Mathematica.* Vediamo ad esempio come importare una immagine JPEG da Internet:

```
img = Import["http://www.dsems.unifg.it/images/ATENEO45.JPG", "JPEG"]
```

Il primo argomento è una stringa contenente l'indirizzo web del file da importare. Il secondo argomento specifica il formato del file (nel nostro caso, un'immagine JPEG).

- **Importare fogli elettronici**

Un tipico "contenitore" di dati da analizzare è il foglio elettronico. In *Mathematica* vi è una cartella chiamata ExampleData, nella quale vi è una raccolta di file di vari formati utili per illustrare l'importazione di dati. Uno di questi file si chiama mountains.sxc (un foglio elettronico OpenOffice). Possiamo importare il foglio in una lista di liste (una matrice) e visualizzarlo in forma tabellare:

```
foglioImportato = Import["ExampleData/mountains.sxc"]
First[foglioImportato] // TableForm
```

```
{{{Rank, Name, Height, First, Ascend}, {1, Mount Everest, 8848, 1953, },
  {2, K2, 8611, 1954, }, {3, Kangchenjunga, 8586, 1955, },
  {4, Lhotse, 8516, 1956, }, {5, Makalu, 8485, 1955, }, {6, Cho Oyu, 8188, 1954, },
  {7, Dhaulagiri, 8167, 1960, }, {8, Manaslu, 8163, 1956, },
  {9, Nanga Parbat, 8125, 1953, }, {10, Annapurna I, 8091, 1950, }}, {{}}, {{}}}
```

Rank	Name	Height	First	Ascend
1	Mount Everest	8848	1953	
2	K2	8611	1954	
3	Kangchenjunga	8586	1955	
4	Lhotse	8516	1956	
5	Makalu	8485	1955	
6	Cho Oyu	8188	1954	
7	Dhaulagiri	8167	1960	
8	Manaslu	8163	1956	
9	Nanga Parbat	8125	1953	
10	Annapurna I	8091	1950	

Possiamo naturalmente analizzare i dati importati. Ad esempio, troviamo l'altezza media delle 10 montagne più alte:

```
Mean[Rest[Map[#[[3]] &, First[foglioImportato]]]]
```

```
8378
```

Ecco in dettaglio i singoli comandi, da cui si comprende la procedura di caoclo seguita:

```
foglioImportato
```

```
{{{Rank, Name, Height, First, Ascend}, {1, Mount Everest, 8848, 1953, },
  {2, K2, 8611, 1954, }, {3, Kangchenjunga, 8586, 1955, },
  {4, Lhotse, 8516, 1956, }, {5, Makalu, 8485, 1955, }, {6, Cho Oyu, 8188, 1954, },
  {7, Dhaulagiri, 8167, 1960, }, {8, Manaslu, 8163, 1956, },
  {9, Nanga Parbat, 8125, 1953, }, {10, Annapurna I, 8091, 1950, }}, {{}}, {{}}}
```

```
First[foglioImportato]
```

```
{{Rank, Name, Height, First, Ascend}, {1, Mount Everest, 8848, 1953, },
 {2, K2, 8611, 1954, }, {3, Kangchenjunga, 8586, 1955, },
 {4, Lhotse, 8516, 1956, }, {5, Makalu, 8485, 1955, }, {6, Cho Oyu, 8188, 1954, },
 {7, Dhaulagiri, 8167, 1960, }, {8, Manaslu, 8163, 1956, },
 {9, Nanga Parbat, 8125, 1953, }, {10, Annapurna I, 8091, 1950, }}
```

```
Map[#[[3]] &, First[foglioImportato]]
```

```
{Height, 8848, 8611, 8586, 8516, 8485, 8188, 8167, 8163, 8125, 8091}
```

```
Rest[Map[#[[3]] &, First[foglioImportato]]]
```

```
{8848, 8611, 8586, 8516, 8485, 8188, 8167, 8163, 8125, 8091}
```

■ Esportazione

Esportare dati da *Mathematica* è altrettanto importante che importarli. **Export** è la funzione ad alto livello che viene utilizzata.

■ Esportare in LaTeX

ExportString è simile a **Export**; differisce semplicemente nel fatto che manda il suo output ad una stringa, che può essere successivamente utilizzata. Esiste anche il complementare comando **ImportString**.
Un formato particolarmente utile in ambito scientifico è il LaTeX. Di solito risulta comodo preparare una serie di calcoli in *Mathematica* e poi esportare il risultato in LaTeX. Vediamo un esempio:

```
ExportString[" ∫_-∞^∞ e^-x^2 dx=√π ", "TeX"]
```

```
%% AMS-LaTeX Created by Wolfram Mathematica 8.0 : www.wolfram.com

\documentclass{article}
\usepackage{amsmath, amssymb, graphics, setspace}

\newcommand{\mathsym}[1]{{}}
\newcommand{\unicode}[1]{{}}

\newcounter{mathematicapage}
\begin{document}

\[\int _{-\infty }^{\infty }e^{-x^2}\text{dx=}\sqrt{\pi }\]

\end{document}
```

Le funzioni Data

Esaminiamo ora le funzioni **Data**. Wolfram Research gestisce una serie di server che forniscono accesso ad una *enorme* mole di dati accessibili tramite *Mathematica*. Vi sono dati matematici, geografici, finanziari, linguistici, scientifici etc. Vedremo alcuni esempi, il cui funzionamento si osservi richiede una connessione Internet funzionante.

Dati chimici

Diamo un'occhiata al set di dati **ChemicalData**. Il seguente comando potrebbe durare un po'; esso restituisce la lista di informazioni chimiche disponibili attraverso la funzione **ChemicalData**:

```
ChemicalData[]
```

A very large output was generated. Here is a sample of it:

```
{LiquidHydrogen, MolecularHydrogen, DeuteriumHydride,
 LiquidHelium, Helium, TritiumHydride, <<43120>>,
 3Hexaprenyl4Hydroxybenzoate, 16:0,18:1Phosphatidylcholine,
 NMethylvaline, Isowillardiine, 7Methylinosine, Ununseptium}
```

Show Less | Show More | Show Full Output | Set Size Limit...

Quali proprietà chimiche possiamo esaminare? Questo ce lo dirà il seguente comando:

```
ChemicalData["Properties"]
```

```
{AcidityConstant, AcidityConstants, AdjacencyMatrix, AlternateNames, AtomPositions,
 AutoignitionPoint, BeilsteinNumber, BoilingPoint, BondTally, CASNumber,
 CHColorStructureDiagram, CHStructureDiagram, CIDNumber, Codons, ColorStructureDiagram,
 CombustionHeat, CompoundFormulaDisplay, CompoundFormulaString, CriticalPressure,
 CriticalTemperature, Density, DensityGramsPerCC, DielectricConstant,
 DOTHazardClass, DOTNumbers, EdgeRules, EdgeTypes, EGECNumber, ElementMassFraction,
 ElementTally, ElementTypes, EUNumber, FlashPoint, FlashPointFahrenheit,
 FormalCharges, FormattedName, GmelinNumber, HBondAcceptorCount, HBondDonorCount,
 HenryLawConstant, HildebrandSolubility, HildebrandSolubilitySI, InChI,
 IonEquivalents, Ions, IonTally, IsoelectricPoint, IsomericSMILES, IUPACName,
 LogAcidityConstant, LowerExplosiveLimit, MDLNumber, MeltingBehavior, MeltingPoint,
 Memberships, MolarVolume, MolecularFormulaDisplay, MolecularFormulaString,
 MolecularWeight, MoleculePlot, Name, NFPAFireRating, NFPAHazards, NFPAHealthRating,
 NFPALabel, NFPAReactivityRating, NonHydrogenCount, NonStandardIsotopeCount,
 NonStandardIsotopeNumbers, NonStandardIsotopeTally, NSCNumber, OdorThreshold,
 OdorType, PartitionCoefficient, pH, Phase, RefractiveIndex, Resistivity,
 RotatableBondCount, RTECSClasses, RTECSNumber, SideChainAcidityConstant, SMILES,
 Solubility, SolubilityType, SpaceFillingMoleculePlot, StandardName, StructureDiagram,
 SurfaceTension, TautomerCount, ThermalConductivity, TopologicalPolarSurfaceArea,
 UpperExplosiveLimit, VanDerWaalsConstants, VaporDensity, VaporizationHeat,
 VaporPressure, VaporPressureTorr, VertexCoordinates, VertexTypes, Viscosity}
```

Ecco un elemento chimico di una certa importanza:

```
ChemicalData["Water"]
```

Ecco come ottenere il punto di ebollizione dell'acqua:

```
ChemicalData["Water", "BoilingPoint"]
```

```
100
```

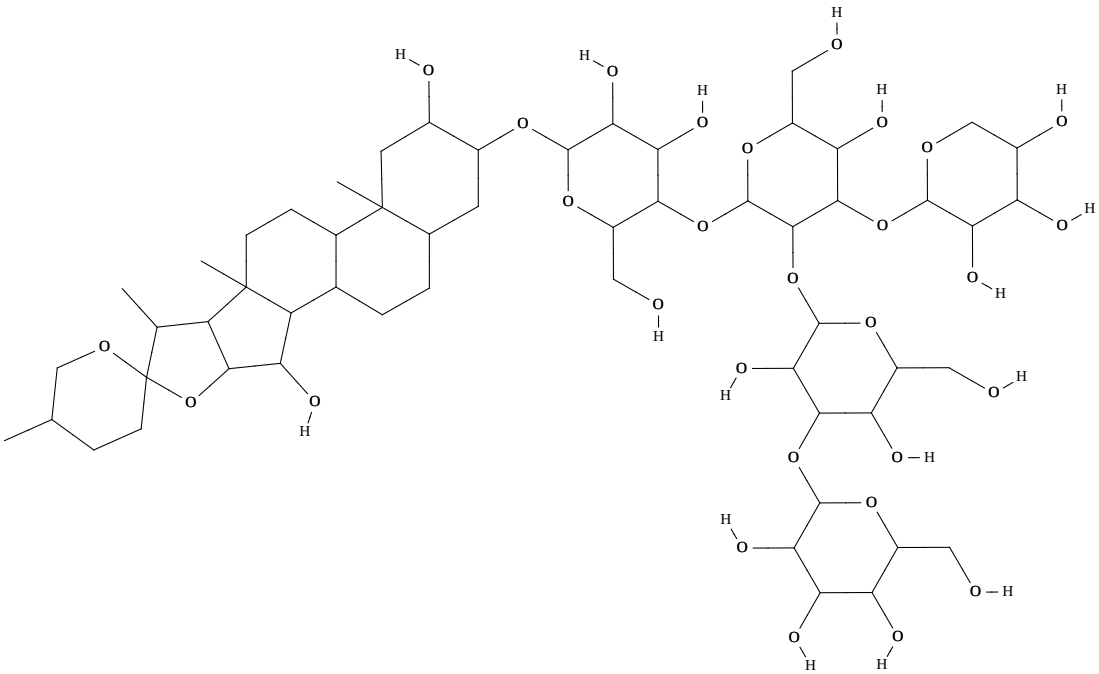
Meno frequente è sicuramente la richiesta di conoscere il punto di fusione della digitonina (un glicoside digitalico che viene principalmente sfruttato per il suo potere tensioattivo):

```
ChemicalData["Digitonin", "MeltingPoint"]
```

```
235.
```

Le complesse molecole di questo gruppo di sostanze sono caratterizzate da una struttura complessa che possiamo esaminare:

```
ChemicalData["Digitonin", "StructureDiagram"]
```



- **Dati finanziari**

Ed ora esaminiamo il data set **FinancialData**. Elenchiamo nuovamente tutte le entità disponibili nel data set in esame:

```
FinancialData[]
```

A very large output was generated. Here is a sample of it:

```
{^A1BSC, ^A1CYC, ^A1DOW, ^A1ENE, ^A1FIN, ^A1HCR, ^A1IDU, ^A1NCY,
 ^A1SGI, ^A1SGITR, ^A1TEC, <<145 481>>, Z:XBNCI, Z:XBPHF, Z:XBPHG,
 Z:XBPHH, Z:XBPHI, Z:XBREF, Z:XBREG, Z:XBREH, Z:XBREI, Z:ZZ6P, Z:ZZ6Q}
```

Show Less | Show More | Show Full Output | Set Size Limit...

Elenchiamo le proprietà finanziarie che *Mathematica* conosce:

```
FinancialData["Properties"]
```

```
{Ask, AskSize, Average200Day, Average50Day, AverageVolume3Month, Bid, BidSize,
 BookValuePerShare, Change, Change200Day, Change50Day, ChangeHigh52Week,
 ChangeLow52Week, CIK, Close, Company, CumulativeFractionalChange, CumulativeReturn,
 CUSIP, Dividend, DividendPerShare, DividendYield, EarningsPerShare, EBITDA,
 Exchange, FloatShares, ForwardEarnings, ForwardPERatio, FractionalChange,
 FractionalChange200Day, FractionalChange50Day, FractionalChangeHigh52Week,
 FractionalChangeLow52Week, High, High52Week, ISIN, LastTradeSize, LatestTrade,
 Lookup, Low, Low52Week, MarketCap, Name, OHLC, OHLCV, Open, PEGRatio, PERatio,
 Price, PriceTarget, PriceToBookRatio, PriceToSalesRatio, QuarterForwardEarnings,
 Range, Range52Week, RawClose, RawHigh, RawLow, RawOHLC, RawOpen, RawRange, Return,
 Sector, SEDOL, ShortRatio, SICCode, StandardName, Symbol, Volatility20Day,
 Volatility50Day, Volume, Website, YearEarningsEstimate, YearPERatioEstimate}
```

Per controllare le quotazioni di borsa abbiamo bisogno di conoscere le sigle ufficiali delle aziende quotate. Ecco i valori attuali di due società ben note in campo informatico:

```
{FinancialData["AAPL"], FinancialData["MSFT"]} (* Apple, Microsoft *)
```

```
{350.44, 25.41}
```

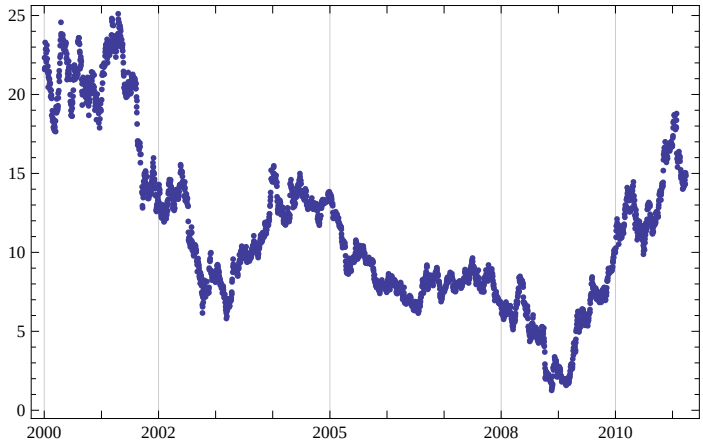
Andiamo a casa nostra:

```
FinancialData["F"](* FIAT *)
```

```
14.86
```

Esaminiamo l'andamento del titolo FIAT negli anni:

```
DateListPlot[FinancialData["F", "Jan. 1, 2000"]]
```



Equazioni differenziali

Anche se *Mathematica* può trattare un'enormità di dati ed argomenti scientifici, il suo scopo primario è "fare" matematica: le equazioni differenziali sono un argomento che trae notevole beneficio dall'uso della potenza computazionale. Nel seguito utilizzeremo la terminologia tipica anglosassone, con DE per Differential Equation ed IVP per Initial Value Problem.

■ Concetti fondamentali

■ Alcune definizioni

Un'*equazione differenziale ordinaria del primo ordine* ha la forma $y'(t) = f(y, t)$. Una soluzione dell'equazione differenziale è una funzione $y(t)$ che soddisfa l'equazione. L'esempio più semplice è dato da $f(y, t) = y$; quindi la DE è $y' = y$. Una soluzione è data da $y(t) = e^t$.
Più in generale, qualsiasi funzione della forma $y(t) = c\, e^t$ soddisfa $y' = y$; diciamo che $y = c\, e^t$ è la *soluzione generale* di $y' = y$. Possiamo specificare la soluzione mediante una *condizione iniziale*. Ad esempio, $y(0) = 2$ insieme a $y' = y$ implica che $y(t) = 2\, e^t$. Per trovarla, dobbiamo prima scrivere $y(t) = c\, e^t$ e poi utilizzare la condizione iniziale $y(0) = 2$ per trovare che $c = 2$. La coppia di equazioni $y' = y,\ y(0) = 2$ viene chiamata "problema del valore iniziale" (IVP, *initial value problem*).

■ DSolve

Quando risolviamo una normale equazione facciamo uso dei comandi **Solve** (risolutore esatto di equazioni algebriche) ed **NSolve** (risolutore numerico approssimato). Vi sono due comandi analoghi a questi per le equazioni differenziali, cioé **DSolve** e **NDSolve**.
DSolve tenta di trovare una soluzione esatta di un'equazione differenziale o di un sistema di equazioni differenziali. La sintassi è **DSolve[***eqs*, *funcs*, *var***]**, dove *eqs* è la lista delle equazioni da risolvere, *funcs* sono le funzioni non note e *var* è la variabile della funzione non nota. ecco alcuni esempi.

L'esempio più semplice:

```
DSolve[y'[t] == y[t], y[t], t]
```

```
{{y[t] → e^t C[1]}}
```

Possiamo aggiungere una condizione iniziale:

```
DSolve[{y'[t] == y[t], y[0] == 2}, y[t], t]
```

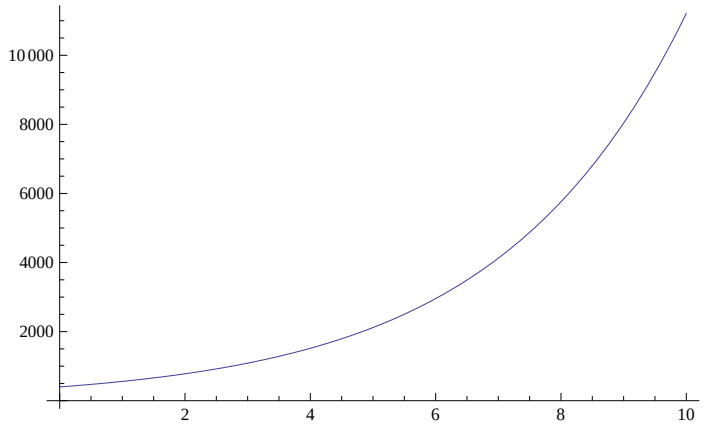
```
{{y[t] → 2 e^t}}
```

Ed ecco un esempio un po' più complesso. E' bene "ripulire" la funzione prima di procedere, per evitare qualsiasi problema di sovrapposizione con oggetti definiti in precedenza.

```
Clear[y, t];
DSolve[{y'[t] == 1/3 y[t], y[0] == 400}, y[t], t]
```

```
{{y[t] → 400 e^(t/3)}}
```

```
Plot[y[t] /. %, {t, 0, 10}]
```



Un altro esempio, dove abbiamo una famiglia di soluzioni dipendenti dalla costante **C[1]**.

```
Clear[y, t];
DSolve[{y'[t] == -1/5 y[t] + 100}, y[t], t]
```

$$\{\{y[t] \to 500 + e^{-t/5}\, C[1]\}\}$$

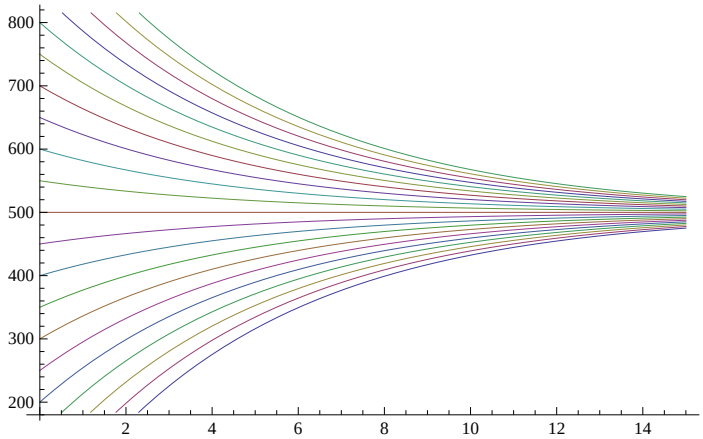
Possiamo ad esempio ottenere diverse soluzioni per valori di C[1] tra -500 e 500 in incrementi di 50:

```
sols = Table[y[t] /. %[[1]] /. C[1] → n, {n, -500, 500, 50}]
```

$$\{500 - 500\, e^{-t/5},\ 500 - 450\, e^{-t/5},\ 500 - 400\, e^{-t/5},\ 500 - 350\, e^{-t/5},\ 500 - 300\, e^{-t/5},$$
$$500 - 250\, e^{-t/5},\ 500 - 200\, e^{-t/5},\ 500 - 150\, e^{-t/5},\ 500 - 100\, e^{-t/5},\ 500 - 50\, e^{-t/5},$$
$$500,\ 500 + 50\, e^{-t/5},\ 500 + 100\, e^{-t/5},\ 500 + 150\, e^{-t/5},\ 500 + 200\, e^{-t/5},\ 500 + 250\, e^{-t/5},$$
$$500 + 300\, e^{-t/5},\ 500 + 350\, e^{-t/5},\ 500 + 400\, e^{-t/5},\ 500 + 450\, e^{-t/5},\ 500 + 500\, e^{-t/5}\}$$

Abbiamo ora una lista di 21 funzioni, ciascuna soluzione dell'equazione differenziale e ciascuna corrispondente ad un diverso valore numerico di **C[1]**. Tracciamo il grafico delle soluzioni sugli stessi assi:

```
Plot[sols, {t, 0, 15}]
```



■ NDSolve

Utilizziamo **NDSolve** in situazioni dove **DSolve** non è in grado di fornire una soluzione algebrica esatta o nessuna soluzione! E' necessario specificare anche una condizione iniziale ed un iteratore per l'intervallo numerico:

```
sol = DSolve[{y'[t] == 0.05 y[t] - 0.0001 y[t]^2, y[0] == 10}, y[t], t]
```

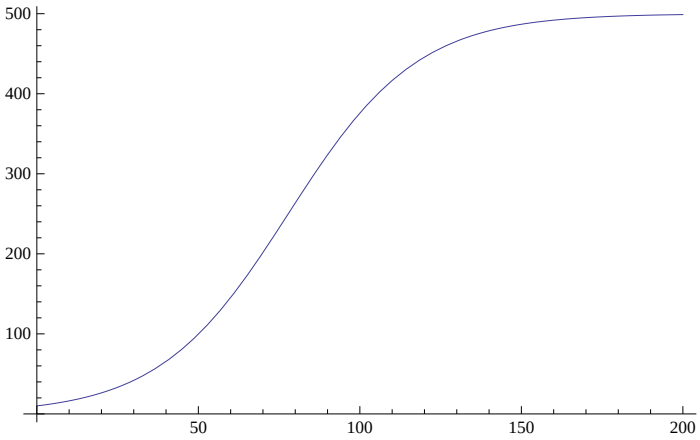
Solve::ifun : Inverse functions are being used by Solve, so some
solutions may not be found; use Reduce for complete solution information. >>

$$\left\{\left\{y[t] \to \frac{500. \times 2.71828^{0.05\, t}}{49. + 2.71828^{0.05\, t}}\right\}\right\}$$

```
sol = NDSolve[{y'[t] == 0.05 y[t] - 0.0001 y[t]^2, y[0] == 10}, y[t], {t, 0, 200}]
```

```
{{y[t] → InterpolatingFunction[{{0., 200.}}, <>][t]}}
```

```
Plot[y[t] /. sol[[1]], {t, 0, 200}]
```



Possiamo anche tabularne alcuni valori:

```
data = Table[{t, y[t] /. sol[[1]]}, {t, 0, 200, 20}];
Text@Grid[Prepend[data, {"t", "y(t)"}], Alignment → ".", Dividers → Gray]
```

t	y(t)
0	10.
20	26.2797
40	65.5185
60	145.367
80	263.509
100	375.895
120	445.848
140	478.614
160	491.914
180	496.995
200	498.89