Guida a Mathematica

Parte terza. Le liste

Marcello Colozzo - http://www.extrabyte.info

Liste

Costruzioni di liste

Una lista è un insieme di "oggetti". Ad esempio:

```
lista = {-4, 5, a, x}
{-4, 5, a, x}
```

È possibile eseguire una qualunque operazione sull'oggetto **`lista`**

```
Log[lista]
{ⅈ π + Log[4], Log[5], Log[a], Log[x]}
```

$$y^{\text{lista}}$$

$$\left\{\frac{1}{y^4}, y^5, y^a, y^x\right\}$$

L'oggetto "lista" ha l'attributo **`Listable`**, nel senso che una qualunque funzione applicata a una lista, restituisce una nuova lista i cui elementi sono il risultato dell'applicazione della funzione ai singoli elementi di **`lista`**. Ad esempio:

$$f[x\_] := \text{Sin}[x + \text{ArcTan}[x]] - x\text{^}5 + 5\,x\text{^}4 + x^3 + 1$$

```
f[lista]
```

$$\{2241 - \text{Sin}[4 + \text{ArcTan}[4]],\ 126 + \text{Sin}[5 + \text{ArcTan}[5]],$$
$$1 + a^3 + 5\,a^4 - a^5 + \text{Sin}[a + \text{ArcTan}[a]],\ 1 + x^3 + 5\,x^4 - x^5 + \text{Sin}[x + \text{ArcTan}[x]]\}$$

Possiamo generare una lista di liste:

```
lista2 = {lista, {lista, lista}}
{{-4, 5, a, x}, {{-4, 5, a, x}, {-4, 5, a, x}}}
```

Table

Un potente generatore di liste è **`Table`**, la cui sintassi (esempio) è:

```
Table[
 k^2,
 {k, 10}
]
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

Qui abbiamo denotato con **k** la variabile di iterazione. Ma è chiaro che avremmo potuto utilizzare un qualunque altro simbolo. Ad esempio:

```
Table[
 x^2,
 {x, 10}
]
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

anche se è più adatto il simbolo **k**, solitamente utilizzato per indicare una variabile discreta. Negli esempi visti, la generazione della lista parte con **k=1**, che il valore di default. Naturalmente possiamo definirne un altro:

```
Table[
 k^2,
 {k, 0, 10}
]
```

```
{0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

Possiamo poi modificare il passo di iterazione:

```
Table[
 k^2,
 {k, 0, 10, 1/10}
]
```

$$\left\{0, \frac{1}{100}, \frac{1}{25}, \frac{9}{100}, \frac{4}{25}, \frac{1}{4}, \frac{9}{25}, \frac{49}{100}, \frac{16}{25}, \frac{81}{100}, 1, \frac{121}{100}, \frac{36}{25}, \frac{169}{100}, \frac{49}{25}, \frac{9}{4}, \frac{64}{25}, \frac{289}{100}, \frac{81}{25}, \frac{361}{100}, 4, \frac{441}{100}, \frac{121}{25}, \frac{529}{100}, \frac{144}{25}, \frac{25}{4}, \frac{169}{25}, \frac{729}{100}, \frac{196}{25}, \frac{841}{100}, 9, \frac{961}{100}, \frac{256}{25}, \frac{1089}{100}, \frac{289}{25}, \frac{49}{4}, \frac{324}{25}, \frac{1369}{100}, \frac{361}{25}, \frac{1521}{100}, 16, \frac{1681}{100}, \frac{441}{25}, \frac{1849}{100}, \frac{484}{25}, \frac{81}{4}, \frac{529}{25}, \frac{2209}{100}, \frac{576}{25}, \frac{2401}{100}, 25, \frac{2601}{100}, \frac{676}{25}, \frac{2809}{100}, \frac{729}{25}, \frac{121}{4}, \frac{784}{25}, \frac{3249}{100}, \frac{841}{25}, \frac{3481}{100}, 36, \frac{3721}{100}, \frac{961}{25}, \frac{3969}{100}, \frac{1024}{25}, \frac{169}{4}, \frac{1089}{25}, \frac{4489}{100}, \frac{1156}{25}, \frac{4761}{100}, 49, \frac{5041}{100}, \frac{1296}{25}, \frac{5329}{100}, \frac{1369}{25}, \frac{225}{4}, \frac{1444}{25}, \frac{5929}{100}, \frac{1521}{25}, \frac{6241}{100}, 64, \frac{6561}{100}, \frac{1681}{25}, \frac{6889}{100}, \frac{1764}{25}, \frac{289}{4}, \frac{1849}{25}, \frac{7569}{100}, \frac{1936}{25}, \frac{7921}{100}, 81, \frac{8281}{100}, \frac{2116}{25}, \frac{8649}{100}, \frac{2209}{25}, \frac{361}{4}, \frac{2304}{25}, \frac{9409}{100}, \frac{2401}{25}, \frac{9801}{100}, 100\right\}$$

Range

L'istruzione **`Range[k]`** genera la lista **`{1,2,3...,k}`**. Ad esempio:

```
Range[9]
```

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

Per **`x`** reale, **`Range[x]`** restituisce {1, 2, 3, ...[x]}:

```
Range[8.5]
```

{1, 2, 3, 4, 5, 6, 7, 8}

Anche qui è possibile definire un valore di partenza e un passo di iterazione:

```
Range[0, 10, 1/2]
```

$$\left\{0, \frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}, 3, \frac{7}{2}, 4, \frac{9}{2}, 5, \frac{11}{2}, 6, \frac{13}{2}, 7, \frac{15}{2}, 8, \frac{17}{2}, 9, \frac{19}{2}, 10\right\}$$

```
Range[x, x + 10, 2]
```

$\{1.00009 \times 10^{-12}, 2., 4., 6., 8., 10.\}$

Array

L'istruzione **`Array[f,n]`** genera una lista di n elementi :

```
Clear[f]
Array[f, 4]
{f[1], f[2], f[3], f[4]}
```

Operazioni sulle liste e manipolazioni di dati

Union

Alcuni comandi operano alla stregua delle operazioni di inclusione e di intersezione della teoria degli insiemi. Per fissare le idee, consideriamo l'esempio seguente. Siano dati gli insiemi (liste):

$$\mathbf{S1} = \left\{x^4, a, e^\pi, \sqrt{2}, \mathrm{Sin}\left[\pi^2\right]\right\}; \mathbf{S2} = \left\{b, d, a, 10^n, x^4, \mathrm{Sin}\left[\pi^2\right], \sqrt{5}\right\};$$

L'unione di tali insiemi (cioè $S_1 \bigcup S_2$) è:

```
Union[S1, S2]
```

$$\left\{\sqrt{2}, \sqrt{5}, 10^n, a, b, d, e^\pi, x^4, \mathrm{Sin}\left[\pi^2\right]\right\}$$

La funzione **`Join`** agisce in maniera simile, ma non elimina gli elementi ripetuti:

Join

```
Join[S1, S2]
```

$$\left\{x^4, a, e^\pi, \sqrt{2}, \mathrm{Sin}\left[\pi^2\right], b, d, a, 10^n, x^4, \mathrm{Sin}\left[\pi^2\right], \sqrt{5}\right\}$$

Tuttavia, l'applicazione della funzione **`Union`** elimina gli elementi ripetuti:

```
Union[Join[S1, S2]]
```

$$\left\{\sqrt{2}, \sqrt{5}, 10^n, a, b, d, e^\pi, x^4, \mathrm{Sin}\left[\pi^2\right]\right\}$$

Intersection

L'intersezione degli insiemi assegnati è $S_1 \bigcap S_2 = \left\{x^4, \sin\left(\pi^2\right)\right\}$. L'operazione di intersezione è implementata dalla funzione **`Intersection`**

```
Intersection[S1, S2]
```

$$\left\{a, x^4, \mathrm{Sin}\left[\pi^2\right]\right\}$$

Range, Complement

Siano dati gli insiemi $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $Y = \{1, 4, 7\}$.

```
X = Range[9]
{1, 2, 3, 4, 5, 6, 7, 8, 9}
Y = {1, 4, 7};
```

Evidentemente : X - Y = {2, 3, 5, 8, 9}. Questa operazione è svolta da :

```
Complement[X, Y]
```

```
{2, 3, 5, 6, 8, 9}
```

Si osservi che la funzione **Complement** opera anche su insiemi disgiunti. Ad esempio:

```
I1 = Range[3]
```

```
{1, 2, 3}
```

```
I2 = Range[4, 6]
```

```
{4, 5, 6}
```

```
Complement[I1, I2]
```

```
{1, 2, 3}
```

```
Complement[I2, I1]
```

```
{4, 5, 6}
```

In altri termini, la funzione **Complement[lista1,lista2]** restituisce gli elementi contenuti in **lista1** ma che non sono contenuti in **lista2**.

First, Last

La funzione **First** restituisce il primo elemento della lista. Ad esempio:

```
lista = {a, b, c, d, e};
```

```
First[lista]
```

```
a
```

Last restituisce l'ultimo elemento:

```
Last[lista]
```

```
e
```

Per l'estrazione dell'elemento *n-esimo*, si utilizza il costrutto **lista[[n]]**. Ad esempio:

```
Clear[lista]
```

$$lista = \left\{1, 4, 9, 16, 22, -1, 12, -\sqrt{2}\right\};$$

```
lista[[3]]
```

```
9
```

Per eseguire l'operazione inversa:

```
Table[
 lista[[n]], {n, 8}
]
```

$$\left\{1, 4, 9, 16, 22, -1, 12, -\sqrt{2}\right\}$$

Il costrutto **lista[[-n]]**restituisce l'*n-esimo* elemento enumerato dall'ultimo. Ad esempio:

```
lista[[-3]]
```

```
-1
```

Partition

Per eseguire una partizione su una lista si utilizza **Partition**

```
Clear[lista]
```

```
lista = {1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, 1728};
```

```
partizione[n_] := Partition[lista, n]
```

```
partizione[1]
```

```
{{1}, {8}, {27}, {64}, {125}, {216},
 {343}, {512}, {729}, {1000}, {1331}, {1728}}
```

```
partizione[2]
```

```
{{1, 8}, {27, 64}, {125, 216}, {343, 512}, {729, 1000}, {1331, 1728}}
```

```
partizione[3]
```

```
{{1, 8, 27}, {64, 125, 216}, {343, 512, 729}, {1000, 1331, 1728}}
```

Cioè l'intero *n* è la lunghezza delle singole sottoliste.

```
partizione[6]
```

```
{{1, 8, 27, 64, 125, 216}, {343, 512, 729, 1000, 1331, 1728}}
```

```
partizione[7]
```

```
{{1, 8, 27, 64, 125, 216, 343}}
```

Una lista di sottoliste:

```
Table[
 partizione[n],
 {n, 5}]
```

```
{{{1}, {8}, {27}, {64}, {125}, {216},
  {343}, {512}, {729}, {1000}, {1331}, {1728}},
 {{1, 8}, {27, 64}, {125, 216}, {343, 512}, {729, 1000}, {1331, 1728}},
 {{1, 8, 27}, {64, 125, 216}, {343, 512, 729}, {1000, 1331, 1728}},
 {{1, 8, 27, 64}, {125, 216, 343, 512}, {729, 1000, 1331, 1728}},
 {{1, 8, 27, 64, 125}, {216, 343, 512, 729, 1000}}}
```

Partizionamento con elementi ripetuti:

```
partizione2[n_, d_] := Partition[lista, n, d]
```

```
partizione2[4, 2]
```

```
{{1, 8, 27, 64}, {27, 64, 125, 216}, {125, 216, 343, 512},
 {343, 512, 729, 1000}, {729, 1000, 1331, 1728}}
```

```
partizione2[3, 6]
```

```
{{1, 8, 27}, {343, 512, 729}}
```

Flatten

Per eliminare eventuali livelli di parentesi si utilizza il comando **Flatten**

```
Clear[lista]
```

```
lista = {Sinh[x], {x, x^2, Log[x], Exp[x]}};
```

```
Flatten[lista]
```

$\{\mathrm{Sinh}[x], x, x^2, \mathrm{Log}[x], e^x\}$

O in notazione postfissa:

```
lista // Flatten
```

$\{\mathrm{Sinh}[x], x, x^2, \mathrm{Log}[x], e^x\}$

Il comando **Flatten** dispone di una opzione che controlla il livello di appiattimento.

```
Clear[lista]
```

```
lista = {{{{x^2, y + 1, I, {2}}}}, {2, 4}};
```

```
Flatten[lista, 2]
```

$\{\{x^2, 1 + y, i, \{2\}\}, 2, 4\}$

```
Flatten[lista, 3]
```

$\{x^2, 1 + y, i, \{2\}, 2, 4\}$

```
Flatten[lista, 4]
```

$\{x^2, 1 + y, i, 2, 2, 4\}$

```
Table[
 Flatten[lista, n],
 {n, 1, 4}
]
```

$\{\{\{\{x^2, 1 + y, i, \{2\}\}\}, 2, 4\}, \{\{x^2, 1 + y, i, \{2\}\}, 2, 4\},$
$\{x^2, 1 + y, i, \{2\}, 2, 4\}, \{x^2, 1 + y, i, 2, 2, 4\}\}$

```
%[[4]]
```

$\{x^2, 1 + y, i, 2, 2, 4\}$

Delete

Per eliminare da una lista l'elemento *i-esimo*, si utilizza il comando **Delete**. Ad esempio:

```
Clear[lista]
```

```
lista = {x^Sin[x], 2, y + c, x + 1};
Delete[lista, 3]
```

$\{x^{\mathrm{Sin}[x]}, 2, 1 + x\}$

Prepend, Append

Per inserire in una lista un nuovo elemento, si utilizza **Prepend**

```
Clear[lista]
lista = {2, 3, 4, 5};
Prepend[lista, 1]
{1, 2, 3, 4, 5}
```

In questo modo la lista rimane immutata:

```
lista
{2, 3, 4, 5}
```

Se vogliamo modificare anche la lista utilizziamo **PrependTo**

```
PrependTo[lista, 1]
{1, 2, 3, 4, 5}
lista
{1, 2, 3, 4, 5}
```

Volendo inserire un elemento in coda ad una lista, si utilizza **Append**

```
Clear[lista]
lista = Range[4]
{1, 2, 3, 4}
Append[lista, 5]
{1, 2, 3, 4, 5}
```

Osserviamo tuttavia, che la lista è immutata:

```
lista
{1, 2, 3, 4}
```

Volendo modificare anche la lista iniziale, si utilizza **AppendTo**

```
AppendTo[lista, 5]
{1, 2, 3, 4, 5}
lista
{1, 2, 3, 4, 5}
```

```
PrependTo[lista, 1]
```

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

ReplacePart

Se vogliamo sostituire l'elemento *k-esimo* con un elemento, utilizziamo la funzione **ReplacePart**

```
Clear[lista]
```

```
lista = {x + y, x + z, x + 2 * y, x + 2 * z};
```

```
ReplacePart[lista, x + 3 * z, 2]
```

```
{x + y, x + 3 z, x + 2 y, x + 2 z}
```

```
ReplacePart[lista, x + 3 * z, -2]
```

```
{x + y, x + z, x + 3 z, x + 2 z}
```

Position, MemberQ, Sort, Reverse

L'istruzione **Position** restituisce la posizione di un elemento all'interno di una lista assegnata.

```
Clear[lista]
```

```
lista = {a, a, z, x, b, {d, 2, {e, {f, g}}}};
```

```
Position[lista, d]
```

```
{{6, 1}}
```

```
Position[lista, x]
```

```
{{4}}
```

```
Position[lista, e]
```

```
{{6, 3, 1}}
```

```
Position[lista, f]
```

```
{{6, 3, 2, 1}}
```

```
Position[lista, g]
```

```
{{6, 3, 2, 2}}
```

MemberQ[lista,y] controlla se **y** appartiene alla lista, restituendo **True** in caso affermativo, **False** nel caso contrario.

```
MemberQ[lista, a]
```

```
True
```

```
MemberQ[lista, d]
```

```
False
```

Nel caso di **d** il kernel restituisce **False**, in quanto tale elemento si trova in un livello annidato.

Se una lista è costituita da parole o lettere, il comando **Sort** riordina la lista in ordine alfabetico, mentre **Reverse** inverte l'ordine.

```
Clear[lista]

lista = {x, d, g, p, q, w, e, h, d, f, j, x};

lista // Sort

{d, d, e, f, g, h, j, p, q, w, x, x}

(*eliminiamo gli elementi ripetuti*)
lista2 = lista // Union

{d, e, f, g, h, j, p, q, w, x}

(*invertiamo l'ordine*)

lista2 // Reverse

{x, w, q, p, j, h, g, f, e, d}
```

Map

Map è probabilmente il comando più potente in grado di agire sulle liste. Esaminiamo la sua sintassi. Per essere più precisi, tale comando non agisce solo sulle liste, cioè sugli insiemi discreti, ma anche su quelli continui, del tipo intervallo (limitato o illimitato) di numeri reali. Quindi consideriamo i due casi distinti: 1) X è un insieme continuo, 2) X è un insieme discreto. Nel primo caso sia $f : X \to Y$, una funzione da $X \subseteq (-\infty, +\infty)$. Il costrutto **Map[f,X]** restituisce l'insieme $f(X)$, cioè l'immagine di X attraverso f. Come è noto, tale sottoinsieme di Y si chiama *codominio* della funzione.

```
(*un intervallo [a,b] si indica -
 in codice Mathematica - con Interval[{a,b}]*)

Map[ArcSin, Interval[{-1, 1}]]
```

Interval$\left[\left\{-\frac{\pi}{2}, \frac{\pi}{2}\right\}\right]$

Una sintassi abbreviata è:

```
ArcSin@Interval[{-1, 1}]
```

Interval$\left[\left\{-\frac{\pi}{2}, \frac{\pi}{2}\right\}\right]$

Cioè anzichè scrivere **Map[f,X]**scriviamo **f@X**, oppure **f/@X**

```
ArcSin /@ Interval[{-1, 1}]
```

Interval$\left[\left\{-\frac{\pi}{2}, \frac{\pi}{2}\right\}\right]$

Passiamo al caso discreto:

```
Clear[f]

X = {a, b, c, d, e};
```

```
Map[f, X]
{f[a], f[b], f[c], f[d], f[e]}
```

```
f /@ X
{f[a], f[b], f[c], f[d], f[e]}
```

Osserviamo che **`Map[f,X]`** opera solo sul primo livello ignorando eventuali livelli annidati. Esempio:

```
Clear[f, X]
```

```
X = {{1, 2}, {4, {0, 10}}}
{{1, 2}, {4, {0, 10}}}
```

```
f /@ X
{f[{1, 2}], f[{4, {0, 10}}]}
```

Il costrutto **`Map[f,lista,r]`** applica f fino al livello r se la lista è costituita da $d > r$ livelli.

```
Map[f, X, 2]
{f[{f[1], f[2]}], f[{f[4], f[{0, 10}]}]}
```

Il comando **`MapAll[f,X]`** applica f a tutti i livelli. La sintassi equivalente è **`f//@X`**

```
f //@ X
f[{f[{f[1], f[2]}], f[{f[4], f[{f[0], f[10]}]}]}]
```

L'attributo Listable

Le funzioni built-in hanno l'attributo **`Listable`**: ignorano l'eventuale presenza di parentesti graffe (ad ogni livello) nei loro argomenti. Esempio:

```
Sin[{a, {4, pi}, {2, 1}}]
{Sin[a], {Sin[4], Sin[pi]}, {Sin[2], Sin[1]}}
```

Per default le funzioni definite dall'utente non hanno l'attributo **`Listable`**. Esempio:

```
f[{a, {4, pi}, {2, 1}}]
f[{a, {4, pi}, {2, 1}}]
```

Per assegnare l'attributo **`Listable`** si utilizza il comando **`SetAttributes[f, Listable]`**

```
SetAttributes[f, Listable]
```

```
f[{a, {4, pi}, {2, 1}}]
{f[a], {f[4], f[pi]}, {f[2], f[1]}}
```

Per quanto detto, le funzioni built-in hanno l'attributo **`Listable`**. Ad esempio, l'operazione di derivazione di una funzione reale di una o più variabili, è implementata dalla funzione built-in **`D[f,x]`**, dove **`f`** è la funzione e **`x`** la variabile rispetto a cui vogliamo determinare la derivata. Consideriamo un esempio istruttivo di operazione di derivazione, anche se l'applicazione grafica richiede la conoscenza del comando **`ParametricPlot`**. Si tratta di tracciare la traiettoria di un oscillatore armonico bidimensionale. Come è noto, per particolari valori delle frequenze dei moti componenti, si ottengono le note figure di Lissajous.

```
Clear[f, x, y]

x[t_] := Cos[3 t]; y[t_] := Cos[2 t]

posizione[t_] := {x[t], y[t]}

velocita[t_] = D[posizione[t], t]

{-3 Sin[3 t], -2 Sin[2 t]}

grafico[τ_] := ParametricPlot[
  {x[t], y[t]},
  {t, 0, 2 π},
  PlotStyle → Thickness[0.003],
  Epilog → {
    {
     Red, Arrow[{{0, 0}, {x[τ], y[τ]}}]
    },
    Point[{x[τ], y[τ]}]
   }
 ]

grafico[π]
```