Lezione 2

In[1]:=

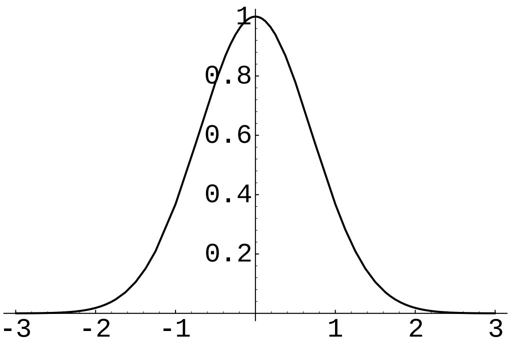
```
f[x_] = E^(-x^2) (* per definire f come funzione di una variabile *)
```

Out[1]=

$$E^{-x^2}$$

In[2]:=

```
Plot[f[x],{x,-3,3}] (* per ottenere il grafico della funzione *)
```



Out[2]=

-Graphics-

In[3]:=

```
f'[x] (* la derivata prima si denota un apice ... *)
```

Out[3]=

$$\frac{-2\ x}{E^{x^2}}$$

In[4]:=

```
f''[x] (* la derivata seconda con due, eccetera ... *)
```

Out[4]=

$$\frac{-2}{E^{x^2}} + \frac{4\ x^2}{E^{x^2}}$$

In[5]:=

```
Integrate[f[x],x] (* integrale indefinito *)
```

Out[5]=

$$\frac{\mathrm{Sqrt[Pi]\ Erf[x]}}{2}$$

In[6]:=

```
Integrate[f[x],{x,-3,3}] (* integrale definito *)
```

Out[6]=

Sqrt[Pi] Erf[3]

In[7]:=

```
N[%]
```

Out[7]=

```
1.77241
```

In[8]:=

```
Integrate[f[x],{x,-Infinity,Infinity}] (* integrale illimitato *)
```

Out[8]=

```
Sqrt[Pi]
```

In[9]:=

```
fn[x_] = f[x]/Sqrt[Pi] (* normalizziamo per avere integrale 1 *)
```

Out[9]=

```
      1
------------
  2
 x
E   Sqrt[Pi]
```

In[10]:=

```
Integrate[fn[x],{x,-Infinity,Infinity}] (* controlliamo ... *)
```

Out[10]=

```
1
```

In[11]:=

```
fn[x_,y_] = fn[x] fn[y] (* fn anche come funzione di due variabili *)
```

Out[11]=

```
   2    2
 -x  - y
E
---------
   Pi
```

In[12]:=

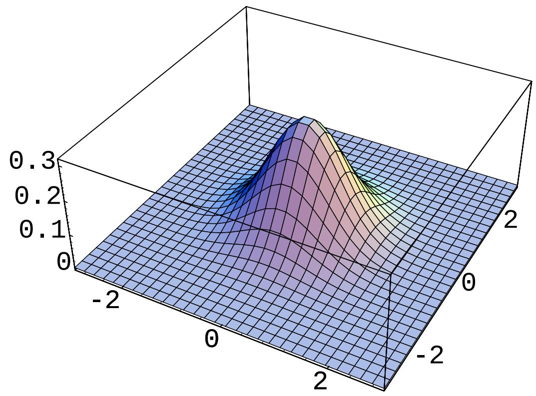
```
?? fn
```

```
Global`fn

fn[x_] = 1/(E^x^2*Pi^(1/2))

fn[x_, y_] = E^(-x^2 - y^2)/Pi
```

In[13]:=
```
Plot3D[fn[x,y],{x,-3,3},{y,-3,3},PlotRange->All,PlotPoints->{30,30}]
```



Out[13]=
-SurfaceGraphics-

In[14]:=
```
Integrate[fn[x,y],{x,-Infinity,Infinity},{y,-Infinity,Infinity}]
```

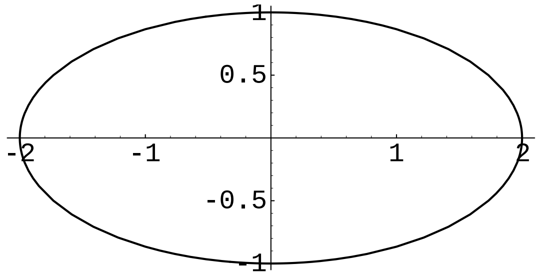
Out[14]=
1

```
(* Si può associare una definizione a qualunque espressione ... *)
```

In[15]:=
```
Ellisse[a_,b_][t_] := {a Cos[t],b Sin[t]}

  (* Ellisse[a,b] è una funzione di una variabile a valori nel piano:
     l'usuale parametrizzazione dell'ellisse di semiassi a e b in forma
     canonica *)
```

In[17]:=
```
ParametricPlot[Ellisse[2,1][t]//Evaluate,{t,0,2Pi},
               AspectRatio->Automatic]
```



Out[17]=
-Graphics-

```
(* Definiamo ora una trasformazione del piano:
   Rotazione[c] = rotazione di un angolo c intorno all'origine *)

(* determiniamo prima la formula della rotazione ... *)
```

```
In[18]:=
  {x,y} /. {x -> r Cos[a],y -> r Sin[a]} (* in coordinate polari *)
Out[18]=
  {r Cos[a], r Sin[a]}
In[19]:=
  % /. a -> a + c (* aggiungiamo c all'angolo *)
Out[19]=
  {r Cos[a + c], r Sin[a + c]}
In[20]:=
  addsincos = {Sin[x_ + y_] -> Sin[x] Cos[y] + Cos[x] Sin[y],
               Cos[x_ + y_] -> Cos[x] Cos[y] - Sin[x] Sin[y]}
Out[20]=
  {Sin[(x_) + (y_)] -> Cos[y] Sin[x] + Cos[x] Sin[y],
    Cos[(x_) + (y_)] -> Cos[x] Cos[y] - Sin[x] Sin[y]}
In[21]:=
  %% /. addsincos  (* applichiamo le formule di addizione *)
Out[21]=
  {r (Cos[a] Cos[c] - Sin[a] Sin[c]), r (Cos[c] Sin[a] + Cos[a] Sin[c])}
In[22]:=
  ExpandAll[%]
Out[22]=
  {r Cos[a] Cos[c] - r Sin[a] Sin[c], r Cos[c] Sin[a] + r Cos[a] Sin[c]}
In[23]:=
  % /. {r Cos[a] -> x, r Sin[a] -> y} (* in coordinate cartesiane *)
Out[23]=
  {x Cos[c] - y Sin[c], y Cos[c] + x Sin[c]}
In[24]:=
  Rotazione[c_][{x_,y_}] = %  (* ... ecco finalmente la definizione *)
Out[24]=
  {x Cos[c] - y Sin[c], y Cos[c] + x Sin[c]}

  (* ora possiamo aggiungere anche l'angolo ad Ellisse *)
In[25]:=
  Ellisse[a_,b_,c_][t_] = Rotazione[c][Ellisse[a,b][t]]
Out[25]=
  {a Cos[c] Cos[t] - b Sin[c] Sin[t], a Cos[t] Sin[c] + b Cos[c] Sin[t]}
```

In[26]:=

```
?? Ellisse
```

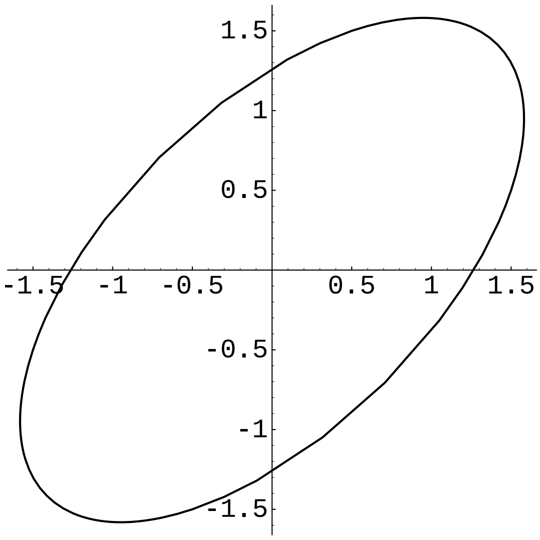
```
Global`Ellisse

Ellisse[a_, b_][t_] := {a*Cos[t], b*Sin[t]}

Ellisse[a_, b_, c_][t_] =
  {a*Cos[c]*Cos[t] - b*Sin[c]*Sin[t],
   a*Cos[t]*Sin[c] + b*Cos[c]*Sin[t]}
```

In[27]:=

```
ParametricPlot[Ellisse[2,1,45 Degree][t]//Evaluate,
  {t,0,2Pi},AspectRatio->Automatic]
```



Out[27]=

```
-Graphics-
```

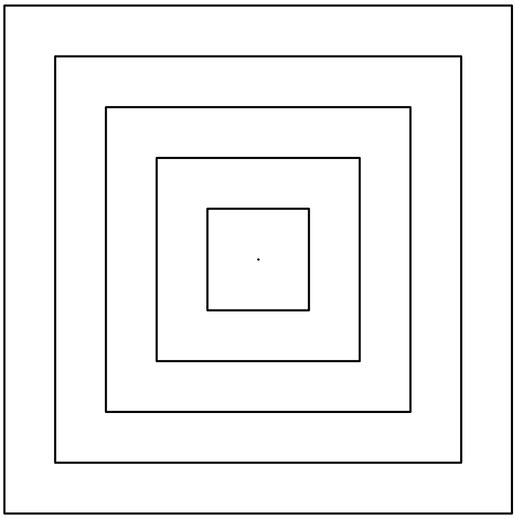
```
(* Una funzione che produce un oggetto grafico ... *)
```

In[28]:=

```
Quadrato[l_] := Graphics[Line[{{l,l},{l,-l},{-l,-l},{-l,l},{l,l}}]]
```

In[29]:=

```
Show[Table[Quadrato[n],{n,0,1,.2}],AspectRatio->Automatic]
```



Out[29]=

-Graphics-

In[30]:=

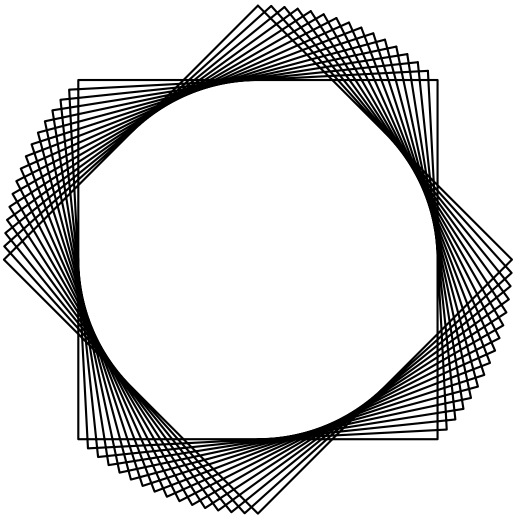
```
(* facciamo operare la funzione Rotazione anche sui poligoni ... *)
```

In[31]:=

```
Rotazione[c_][Graphics[Line[listapunti_]]] :=
  Graphics[Line[Map[Rotazione[c],listapunti]]]
```

In[32]:=

```
Show[Table[Rotazione[a][Quadrato[1]],
           {a,0,45 Degree,3 Degree}],
     AspectRatio->Automatic]
```



Out[32]=

-Graphics-

```
(* Una funzione con un numero imprecisato di argomenti ... *)
```

In[33]:=

```
Media[x__] := Plus[x]/Length[{x}]
```

```
In[34]:=
  Table[Random[Integer,{0,10}],{i,1,8}]
Out[34]=
  {0, 1, 0, 9, 7, 4, 5, 6}
In[35]:=
  Apply[Media,%]
Out[35]=
  4
In[36]:=
  N[%]
Out[36]=
  4.

  (* verifichiamo, nel caso di otto valori arbitrari,
     che la media minimizza la somma degli scarti quadratici ... *)
In[37]:=
  Apply[Media,Table[x[i],{i,1,8}]]
Out[37]=
  x[1] + x[2] + x[3] + x[4] + x[5] + x[6] + x[7] + x[8]
  -----------------------------------------------------
                            8
In[38]:=
  Sum[(x - x[i])^2,{i,1,8}]  (* somma degli scarti quadratici da x *)
Out[38]=
            2              2              2              2              2
  (x - x[1])  + (x - x[2])  + (x - x[3])  + (x - x[4])  + (x - x[5])  +
              2              2              2
    (x - x[6])  + (x - x[7])  + (x - x[8])
In[39]:=
  D[%,x]
Out[39]=
  2 (x - x[1]) + 2 (x - x[2]) + 2 (x - x[3]) + 2 (x - x[4]) +
    2 (x - x[5]) + 2 (x - x[6]) + 2 (x - x[7]) + 2 (x - x[8])
In[40]:=
  Solve[% == 0,x] (* vediamo dove si annulla la derivata prima ... *)
Out[40]=
         x[1] + x[2] + x[3] + x[4] + x[5] + x[6] + x[7] + x[8]
  {{x -> -----------------------------------------------------}}
                                   8
In[41]:=
  D[%%,x] /. %[[1]] (* vediamo se la derivata seconda è positiva ... *)
Out[41]=
  16

  (* tre definizioni diverse per la funzione valore assoluto *)
In[42]:=
  Abs1[x_] := If[x >= 0,x,-x]
```

In[43]:=

```
Abs1[-4]
```

Out[43]=

```
4
```

In[44]:=

```
Abs1[-Sqrt[2]] (* il confronto non viene eseguito se compaiono
                  elementi simbolici ... *)
```

Out[44]=

```
If[-Sqrt[2] >= 0, -Sqrt[2], --Sqrt[2]]
```

In[45]:=

```
Abs2[x_] := If[N[x >= 0],x,-x] (* questa definizione va meglio ... *)
```

In[46]:=

```
Abs2[-Sqrt[2]]
```

Out[46]=

```
Sqrt[2]
```

In[47]:=

```
Abs2[a] (* il problema resta se l'argomento non ha un valore
           numero, in questo caso il valore assoluto dovrebbe
           restare indicato ... *)
```

Out[47]=

```
If[N[a >= 0], a, -a]
```

In[48]:=

```
Abs3[x_?NumberQ] := If[N[x >= 0],x,-x]

  (* questa definizione sarà applicata solo ad argomenti numerici ... *)
```

In[50]:=

```
Abs3[a]
```

Out[50]=

```
Abs3[a]
```

```
(* tre definizioni equivalenti della funzione fattoriale ... *)
```

In[51]:=

```
Fatt1[n_Integer /; n >= 0] := Product[i,{i,1,n}]  (* stile matematico *)
```

In[52]:=

```
Fatt2[n_Integer /; n >= 0] :=                    (* stile procedurale *)
   Block[{i = 1,p = 1},
      While[i <= n,
         p = p i;
         i = i + 1];
      p]
```

In[53]:=

```
Fatt3[0] := 1                           (* stile funzionale ricorsivo *)
Fatt3[n_Integer /; n > 0] := n Fatt3[n - 1]
```

```
In[55]:=
  {Fatt1[5],Fatt2[5],Fatt3[5]}
Out[55]=
  {120, 120, 120}

  (* una funzione con argomento e valore funzionale ... *)
In[56]:=
  FunzioneDerivata[f_] := Function[t,Evaluate[D[f[t],t]]]
In[57]:=
  FunzioneDerivata[Sin]
Out[57]=
  Function[t$, Cos[t$]]
In[58]:=
  FunzioneDerivata[Sin][x]
Out[58]=
  Cos[x]

  (* due funzioni che applicano i principi di equivalenza alle equazioni *)
In[59]:=
  IPrincEq[m1_ == m2_,esp_] := m1 + esp == m2 + esp
In[60]:=
  IIPrincEq[m1_ == m2_,esp_] := m1 esp == m2 esp

  General::spell1:
     Possible spelling error: new symbol name "IIPrincEq"
       is similar to existing symbol "IPrincEq".

  (* esempio (banale) di utilizzo ... *)
In[61]:=
  a x + b == 0
Out[61]=
  b + a x == 0
In[62]:=
  IPrincEq[%,-b]
Out[62]=
  a x == -b
In[63]:=
  IIPrincEq[%,1/a]
Out[63]=
           b
  x == -(-)
           a

  (* si potrebbe aggiungere qualcosa ... *)
In[64]:=
  IIPrincEq[m1_ == m2_ ,esp_] := m1 esp == m2 esp &&
    Numerator[esp] != 0 && Denominator[esp] != 0
```

In[65]:=

```
IIPrincEq[%%%,1/a]
```

Out[65]=

```
          b
x == -(-) && a != 0
          a
```