In [1]:
```
# FK5024 - TUTORIAL 8
# Python basics
```

In [2]:
```
a = 4
print(a)
```

```
4
```

In [3]:
```
a = a+5
```

In [4]:
```
a;
```

In [5]:
```
print(a)
```

```
9
```

In [6]:
```
print("Hello world!")
```

```
Hello world!
```

In [7]:
```
# FUNCTIONS
# Functions are very important because they avoid repetitions of the code,
# thus minimizing errors and making the code more efficient and more readable

a="Hello"
print (type(a))
```

```
<class 'str'>
```

In [8]:
```
print (type(a))
```

```
<class 'str'>
```

In [9]:
```
# Let's define our own function

def mySum (a,b):
    c = a + b;
    return c;

print (mySum (32,7))
```

```
39
```

In [10]:
```
# CONDITIONS
# Conditions are widely used in any computing language.
# They allow to evaluate different codes when different conditions are met

a = 5
b = 8
if a >= 0 :
    b = b + 20
    print("b =",b)
else :
    print("a is inferior to zero.")
```

```
b = 28
```

In [11]:
```
# SOME COMMON MATHEMATICAL OPERATORS
# <
# >
# >=
# <=
# == (equal)
# != (different)
```

In [12]:
```
# LOOPS
# They allow one to perform operations in sequence until a certain condition
is met

sentence = "Hello world !"
for letter in sentence:
    print (letter)
```

```
H
e
l
l
o

w
o
r
l
d

!
```

In [13]:
```
# You can combine the loops with the conditions

for letter in sentence:
    if letter not in "aeiouy":
        print (letter)
    else:
        print ('.')
```

```
H
.
l
l
.

w
.
r
l
d

!
```

In [14]:
```
# Libraries are files where different related functions are stored,
# and you can use them if you import the library.
```

In [15]:
```
a = 25
#print(sqrt(a)) # this will not work a priori
```

In [16]:
```
# To be able to calculate a square root,
# we need to import the mathematics module

a=25
import math
print(math.sqrt(a))
```

```
5.0
```

In [17]:
```
#help("math") # to see all the functions available
```

In [18]:
```
help("math.sqrt")
```

```
Help on built-in function sqrt in math:

math.sqrt = sqrt(...)
    sqrt(x)

    Return the square root of x.
```

In [19]:
```
# You can also import only the functions you need
# from a module instead of importing it entirely

from math import sqrt
```

In [20]:
```
# You can import * which will spare you the "math." before the functions

from math import *
print (sqrt(a))
```

```
5.0
```

In [21]:
```
# Now we will introduce two libraries: NUMPY and MATPLOTLIB

# NUMPY allows you to use arrays, whereas MATPLOTLIB allows you to plot
```

In [22]:
```
import numpy as np
```

In [23]:
```
# We can now define a vector...

a = np.array([1,4,6,2])
print(a)
```

```
[1 4 6 2]
```

In [24]:
```
# ...and a matrix

A = np.array([[1,1],[4,2],[9,3]])
print(A)
```

```
[[1 1]
 [4 2]
 [9 3]]
```

In [25]:
```
# You can also access the numbers of rows and columns in
# your matrix, with the function "shape" in the object A

A.shape
```

Out[25]:
```
(3, 2)
```

In [26]:
```
# The total number of element is given by the function "size"

A.size
```

Out[26]: 6

In [27]:
```
# Once you have a vector or a matrix, you can easily access
# every single element in them. Note that the first row and
# column are labeled with 0, not with 1

A[0,0]
```

Out[27]: 1

In [28]:
```
A[2,1] # the first number refers to the row,
##       the second number to the column
```

Out[28]: 3

In [29]:
```
print(A[2,:])
```

```
[9 3]
```

In [30]:
```
# You can use loops here too

for row in A:
    print(row)
```

```
[1 1]
[4 2]
[9 3]
```

In [31]:
```
for i in range(len(A)):
    print(A[i])
```

```
[1 1]
[4 2]
[9 3]
```

In [32]:
```
# Fill an array:

B = np.zeros(10)
print(B)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [33]:
```
range(0,10)
```

Out[33]: range(0, 10)

In [34]:
```
# Note that the indentation in python is extremely important

for i in range(10):
    B[i]=i**3
print (B)
```

```
[  0.   1.   8.  27.  64. 125. 216. 343. 512. 729.]
```

In [35]:
```
# Operations on matrices

B = B*2
print(B)
```

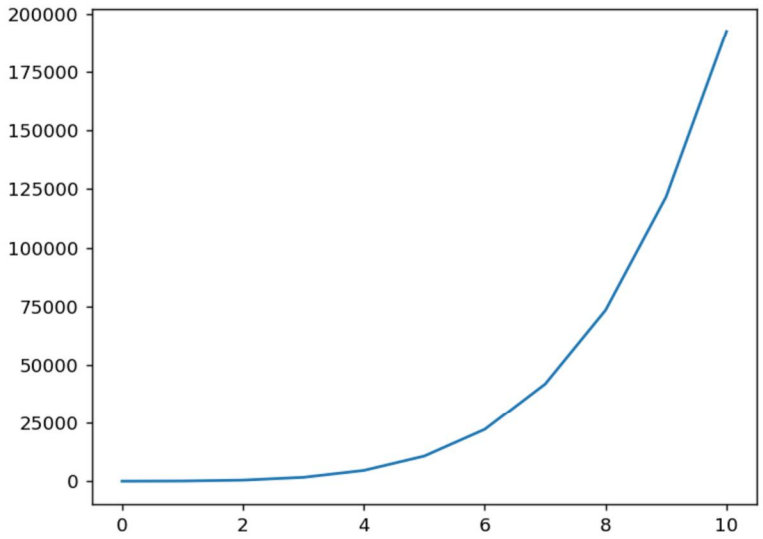
```
[   0.    2.   16.   54.  128.  250.  432.  686. 1024. 1458.]
```

In [36]:
```
# Let's go now to MATPLOTLIB. You use it to make plots

import matplotlib.pyplot as plt
```

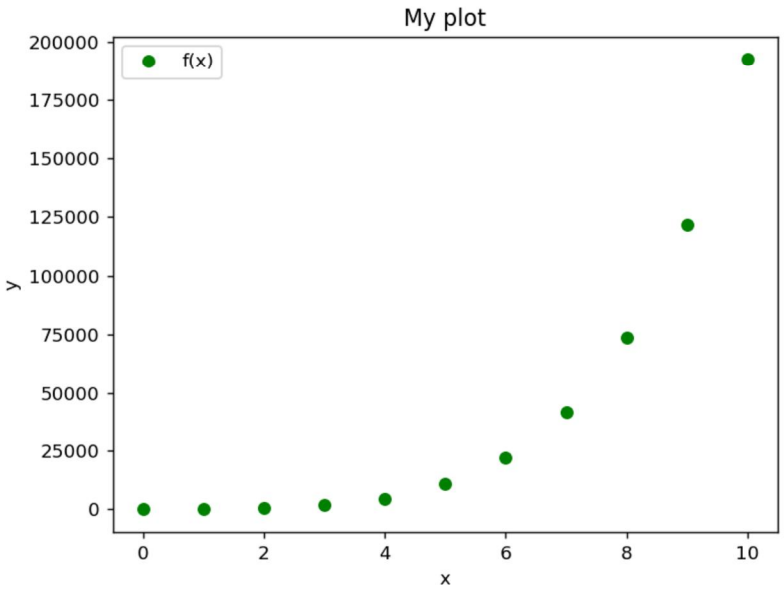
In [37]:
```
x = range(5)
f = np.zeros(len(x))
for i in range(len(x)):
    f[i]=i**2
print('x: ',x)
print('f: ',f)
```

```
x:  range(0, 5)
f:  [ 0.  1.  4.  9. 16.]
```

In [67]:
```
%matplotlib nbagg
# the command above is just to make the plot appear below,
# not in another window
plt.plot(x,f)
plt.show()
```



In [66]:
```
%matplotlib nbagg
plt.plot(x,f,"o",color="green",label="f(x)")
plt.xlabel('x')
plt.ylabel('y')
plt.title('My plot')
plt.legend()
plt.show()
```



In [40]:
```
# And many, many other possibilities!
```

In [41]:
```
# EXERCISES
# (1) Create a null vector of size 10 but where the fifth value is 1.
# (2) Write a function which can compute the factorial of a given number.
# (3) Write a function that will calculate (a + b)^n. Then plot the result fo
r a=1.4, n=5, and b in [0;10].
# (4) Read the data file exercise4_stars.txt and plot the luminosity as a fun
ction of temperature.
# (5) Read the data file FIRAS_CIB_spectrum.dat and plot the monopole spectru
m of the CMB.
#     (5b) Fit the CMB specturm with a fourth-order polynomial.
```

In [42]:
```
# (1)

Z = np.zeros(10)
Z[4] = 1
print(Z)
```

```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```

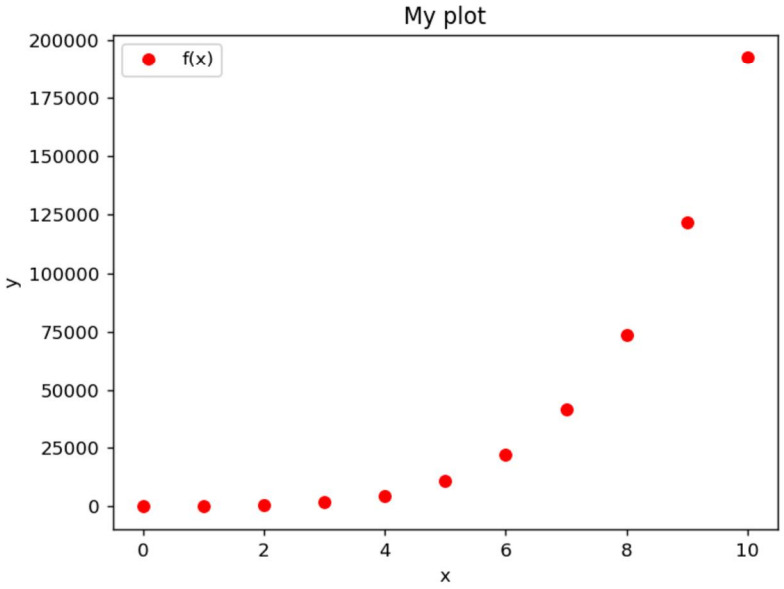
In [43]:
```
# (2) Note the indentation

def factorial(x):
    if x == 0:
        return 1
    return x * factorial(x-1)

print (factorial(4))
```

```
24
```

In [44]:
```
# (3) You have to execute "import numpy as np", if not already done

def newton(a,b,n):
    return (a+b)**n

b = range(11)

F = np.ma.zeros(len(b))

for i in range(len(b)):
    F[i] = newton (1.4,i,5)

print(F)
```

```
[5.37824000e+00 7.96262400e+01 4.54354240e+02 1.64916224e+03
 4.59165024e+03 1.07374182e+04 2.21900662e+04 4.18211942e+04
 7.33904022e+04 1.21665290e+05 1.92541458e+05]
```

In [65]:
```
# import matplotlib.pyplot as plt

%matplotlib nbagg
plt.plot(b,F,"o",color="red",label="f(x)")
plt.xlabel('x')
plt.ylabel('y')
plt.title('My plot')
plt.legend()
plt.show()
```



In [46]:
```
# (4) data from http://www.isthe.com/chongo/tech/astro/HR-temp-mass-table-byhrclass.html

data = np.genfromtxt("exercise4_stars.txt", skip_header = 11)
print(data)
# nan = not a number
```

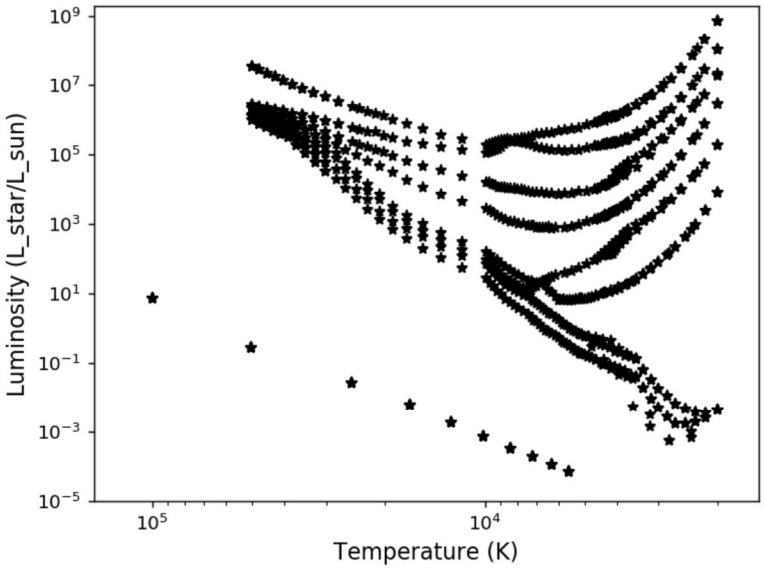
```
[[     nan 1.60e+02 3.41e+07 ... 1.44e+02 1.66e+02 2.55e+02]
 [     nan 1.50e+02 2.59e+06 ... 1.44e+02 1.66e+02 2.55e+02]
 [     nan 1.40e+02 2.15e+06 ... 1.44e+02 1.66e+02 2.55e+02]
 ...
 [     nan 3.00e-01 1.80e-04 ... 2.43e+02 2.43e+02 2.55e+02]
 [     nan 2.00e-01 1.05e-04 ... 2.55e+02 2.47e+02 2.45e+02]
 [     nan 1.00e-01 6.73e-05 ... 2.55e+02 2.39e+02 2.25e+02]]
```

In [47]:
```
temperature = data[:, 4] # data['Temperature']
luminosity = data[:, 2] # data['Luminosity']
```

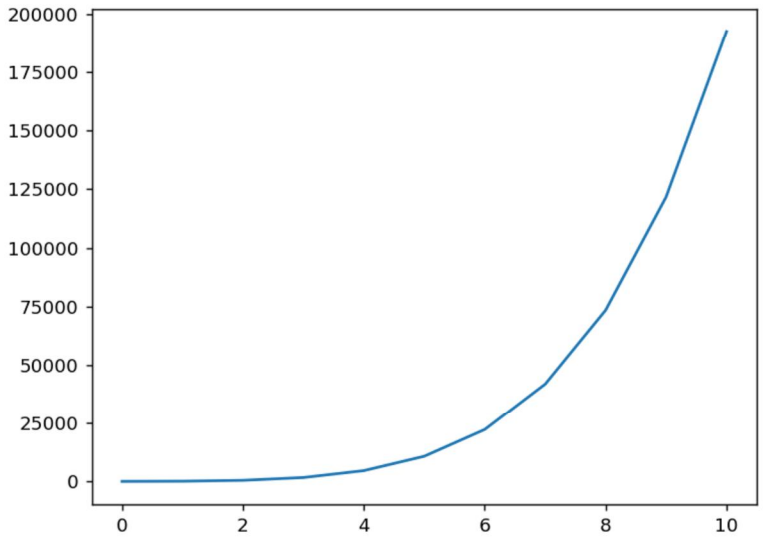
In [48]:
```
print (temperature)
print (luminosity)
```

```
[50000. 50000. 50000. ...  7200.  6300.  5600.]
[3.41e+07 2.59e+06 2.15e+06 ... 1.80e-04 1.05e-04 6.73e-05]
```

In [64]:
```
%matplotlib nbagg
plt.loglog(temperature,luminosity,"*",color="black")
plt.xlabel("Temperature (K)",fontsize=12)
plt.ylabel("Luminosity (L_star/L_sun)",fontsize=12)
plt.ylim(10**-5,2*10**9)
plt.xlim(1.5*10**5,1.5*10**3)
plt.show()
```



In [63]:
```
# Alternative solution to (3)

%matplotlib nbagg
x = range(11)
f=np.ma.zeros(len(x))
def plotting (b,n,x):
    for i in x:
        f[i]=(b+i)**n
    plt.plot(x,f)
    plt.show()
plotting(1.4,5,x)
```



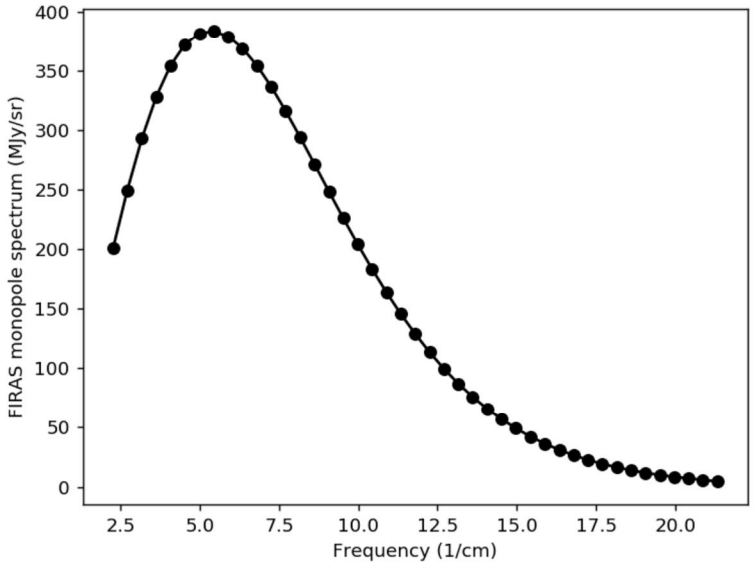
In [51]:
```
# Alternative solution to (2)

def factorial(a):
    x=1
    for i in range(a):
        i+=1 # i=i+1
        x=x*i
    return(x)
print (factorial(a=3))
```

```
6
```

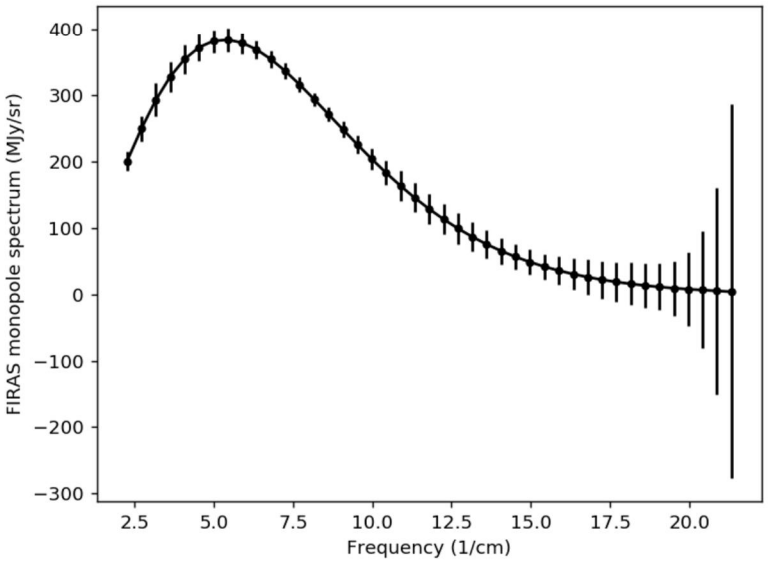
In [52]:
```
# (5)

dataCMB = np.genfromtxt("FIRAS_CIB_spectrum.dat", skip_header = 13)
print(dataCMB)
# nan = not a number
```

```
[[   2.27   200.723    5.      14.       4.   ]
 [   2.72   249.508    9.      19.       3.   ]
 [   3.18   293.024   15.      25.      -1.   ]
 [   3.63   327.77     4.      23.      -1.   ]
 [   4.08   354.081   19.      22.       3.   ]
 [   4.54   372.079  -30.      21.       6.   ]
 [   4.99   381.493  -30.      18.       8.   ]
 [   5.45   383.478  -10.      18.       8.   ]
 [   5.9    378.901   32.      16.      10.   ]
 [   6.35   368.833    4.      14.      10.   ]
 [   6.81   354.063   -2.      13.      12.   ]
 [   7.26   336.278   13.      12.      20.   ]
 [   7.71   316.076  -22.      11.      25.   ]
 [   8.17   293.924    8.      10.      30.   ]
 [   8.62   271.432    8.      11.      36.   ]
 [   9.08   248.239  -21.      12.      41.   ]
 [   9.53   225.94     9.      14.      46.   ]
 [   9.98   204.327   12.      16.      57.   ]
 [  10.44   183.262   11.      18.      65.   ]
 [  10.89   163.83   -29.      22.      73.   ]
 [  11.34   145.75   -46.      22.      93.   ]
 [  11.8    128.835   58.      23.      98.   ]
 [  12.25   113.568    6.      23.     105.   ]
 [  12.71    99.451   -6.      23.     121.   ]
 [  13.16    87.036    6.      22.     135.   ]
 [  13.61    75.876  -17.      21.     147.   ]
 [  14.07    65.766    6.      20.     160.   ]
 [  14.52    57.008   26.      19.     178.   ]
 [  14.97    49.223  -12.      19.     199.   ]
 [  15.43    42.267  -19.      19.     221.   ]
 [  15.88    36.352    8.      21.     227.   ]
 [  16.34    31.062    7.      23.     250.   ]
 [  16.79    26.58    14.      26.     275.   ]
 [  17.24    22.644  -33.      28.     295.   ]
 [  17.7     19.255    6.      30.     312.   ]
 [  18.15    16.391   26.      32.     336.   ]
 [  18.61    13.811  -26.      33.     363.   ]
 [  19.06    11.716   -6.      35.     405.   ]
 [  19.51     9.921    8.      41.     421.   ]
 [  19.97     8.364   26.      55.     435.   ]
 [  20.42     7.087   57.      88.     477.   ]
 [  20.87     5.801 -116.     155.     519.   ]
 [  21.33     4.523 -432.     282.     573.   ]]
```

In [62]:
```
%matplotlib nbagg
plt.plot(dataCMB[:,0],dataCMB[:,1],linestyle='-', marker='o',color="black")
plt.xlabel("Frequency (1/cm)")
plt.ylabel("FIRAS monopole spectrum (MJy/sr)")
plt.show()
```



In [54]:
```
%matplotlib nbagg
plt.errorbar(dataCMB[:,0],dataCMB[:,1],yerr=dataCMB[:,3],linestyle='-',marker
='o',markersize=3.5,color="black")
plt.xlabel("Frequency (1/cm)")
plt.ylabel("FIRAS monopole spectrum (MJy/sr)")
plt.show()
```



In [55]:
```
# We can also fit this data. We need to import the function "curve_fit" from
the library scipy

from scipy.optimize import curve_fit
```

In [56]:
```
# Let's use a fourth-degree polynomial to fit the CMB monopole spectrum data

def func(x,a,b,c,d,e):
    return a + b*x + c*x**2 + d*x**3 + e*x**4
```

In [57]:
```
?curve_fit
```

In [58]:
```
# This is the fit

popt, pcov = curve_fit(func,dataCMB[:,0],dataCMB[:,1],sigma=dataCMB[:,3])
perr = np.sqrt(np.diag(pcov))
```

In [59]:
```
# Print the fit parameters and 1-sigma estimates

print("fit parameter 1-sigma error")
print("———————————————")
for i in range(len(popt)):
    print(str(popt[i])+" +- "+str(perr[i]))
```

```
fit parameter 1-sigma error
———————————————
-298.8009462216686 +- 10.322734097068489
315.77603855985535 +- 5.4614942484519196
-48.226597498715854 +- 0.9376188066579285
2.6811390695100017 +- 0.06357435929798194
-0.05115456082447738 +- 0.001479309373100118
```

In [60]:
```
# Prepare confidence level curves

nstd = 1 # to draw 1-sigma intervals
popt_up = popt + nstd * perr
popt_dw = popt - nstd * perr

#args = [1, 2]
#kwargs = dict("name" = "adri")
#kwargs["name"]

#func(*args, **kwargs)

fit = func(dataCMB[:,0], *popt)
fit_up = func(dataCMB[:,0], *popt_up)
fit_dw = func(dataCMB[:,0], *popt_dw)
```

In [61]:
```
# Plot the data and the fit
fig = plt.subplots(1)
#rcParams["xtick.labelsize"] = 12
#rcParams["ytick.labelsize"] = 12
#rcParams["font.size"]= 12
plt.errorbar(dataCMB[:,0],dataCMB[:,1],yerr=dataCMB[:,3],linestyle="",marker=
'o',markersize=3.5,color="black")

plt.xlabel("Frequency (1/cm)",fontsize=12)
plt.ylabel("FIRAS monopole spectrum (MJy/sr)",fontsize=12)
plt.xlim(0,25)
plt.ylim(-300,420)
plt.title("CMB monopole fit", fontsize=12)
plt.plot(dataCMB[:,0],fit,color="red")
#plt.plot(dataCMB[:,0],fit_up,color="blue")
#plt.plot(dataCMB[:,0],fit_dw,color="blue")
plt.show()
```