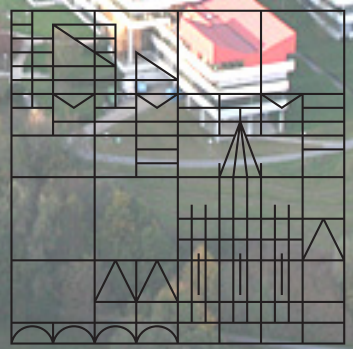
Python Plotting

Matplotlib, etc.

S. Gerlach

Mar. 24, 2022

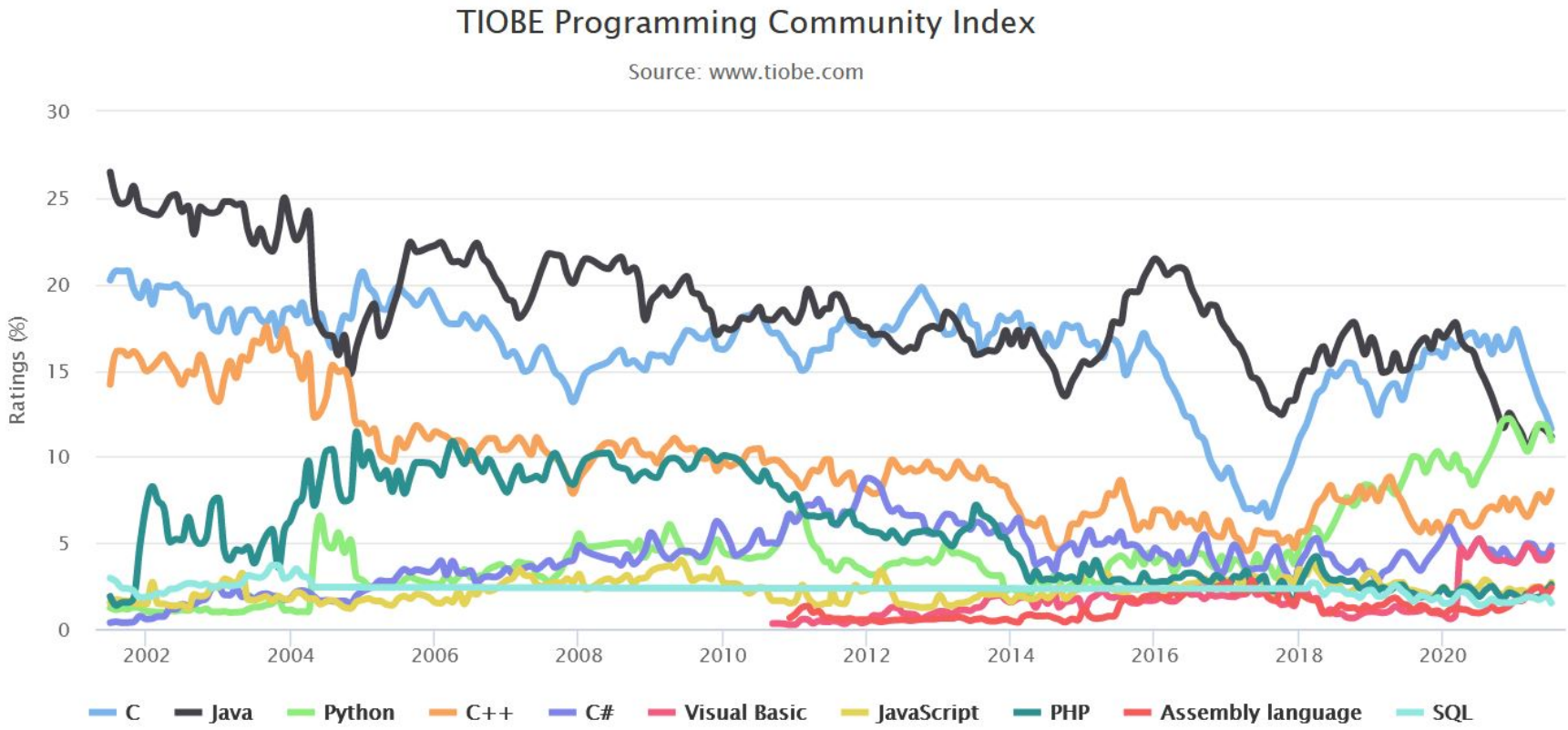
Universität
Konstanz



Programming languages - Usage (TIOBE)

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Python	2	5	6	8	25	25	-	-
Java	3	1	1	1	2	16	-	-
C++	4	3	3	3	3	2	2	6
C#	5	4	5	7	12	-	-	-
Visual Basic	6	13	-	-	-	-	-	-
JavaScript	7	7	10	9	9	20	-	-
PHP	8	6	4	4	10	-	-	-
Assembly language	9	11	-	-	-	-	-	-
SQL	10	-	-	-	37	-	-	-
Ada	29	28	17	16	18	7	5	2
Lisp	33	27	13	13	17	8	4	3
Pascal	268	75	15	17	15	5	3	7
(Visual) Basic	-	-	7	5	4	3	8	5

Programming languages - Usage (TIOBE)



Programming language Python

- Wide spread (well known, many applications)
- **Free Software**
- **Interactive Scripting**
- **Performance** by using optimized libraries or binding to C/C++
- Easy & flexible (few language features, many libraries)

current: Python 2.7.18 (End-of-Life 2020), Python 3.7.12 - 3.10.2

`https://en.wikibooks.org/wiki/Python_Programming`
`http://www.scipy-lectures.org/_downloads/ScipyLectures-simple.pdf`

Usage - interaktive

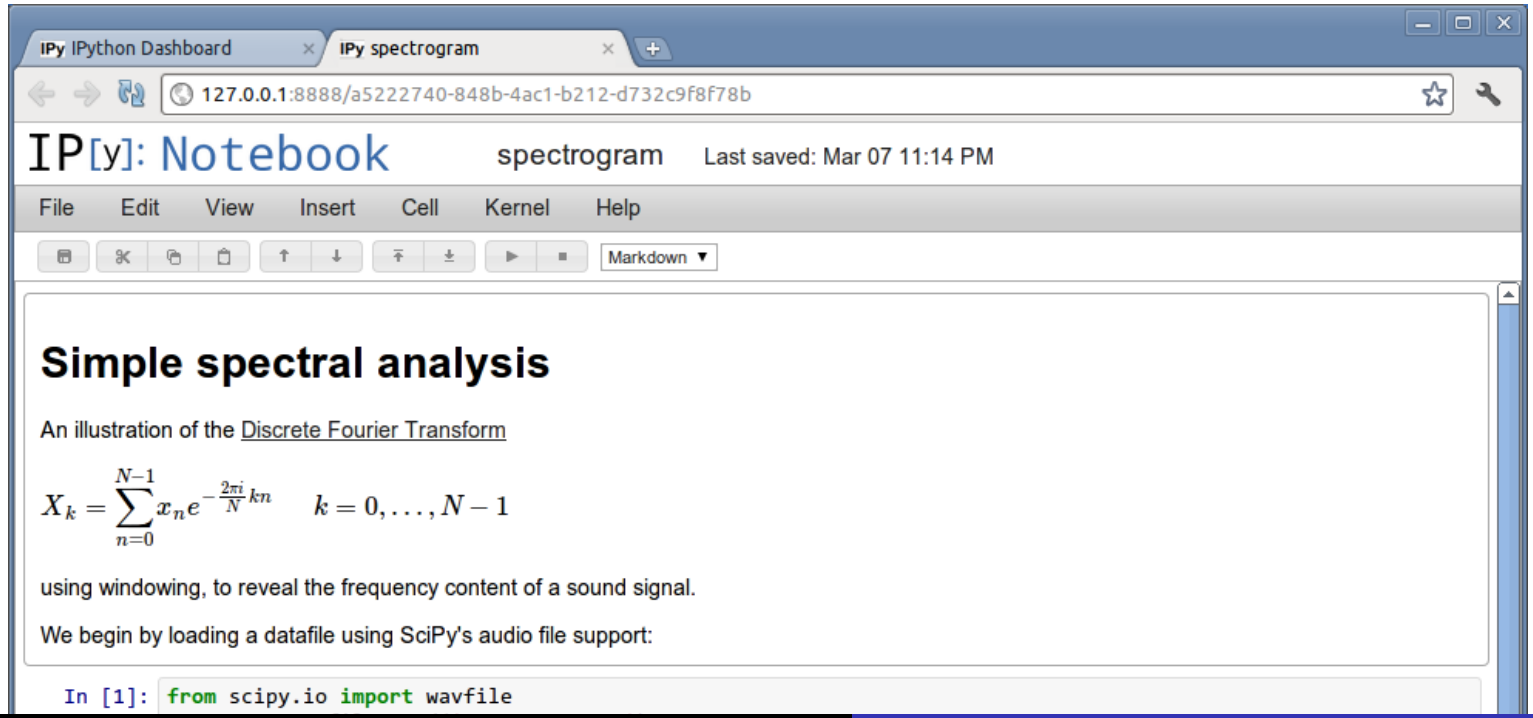
1. **Terminal**: `python`, `python3`

   ```
   >>> 1+1
   ```

2. **IPython** - opt. Terminal: `ipython3`, `ipython3 qtconsole`

   ```
   In[1]: %hist
   In[2]: hist?
   In[2]: help(abs)
   ```

3. **Im Browser** - IPython-Notebook: `ipython3 notebook`

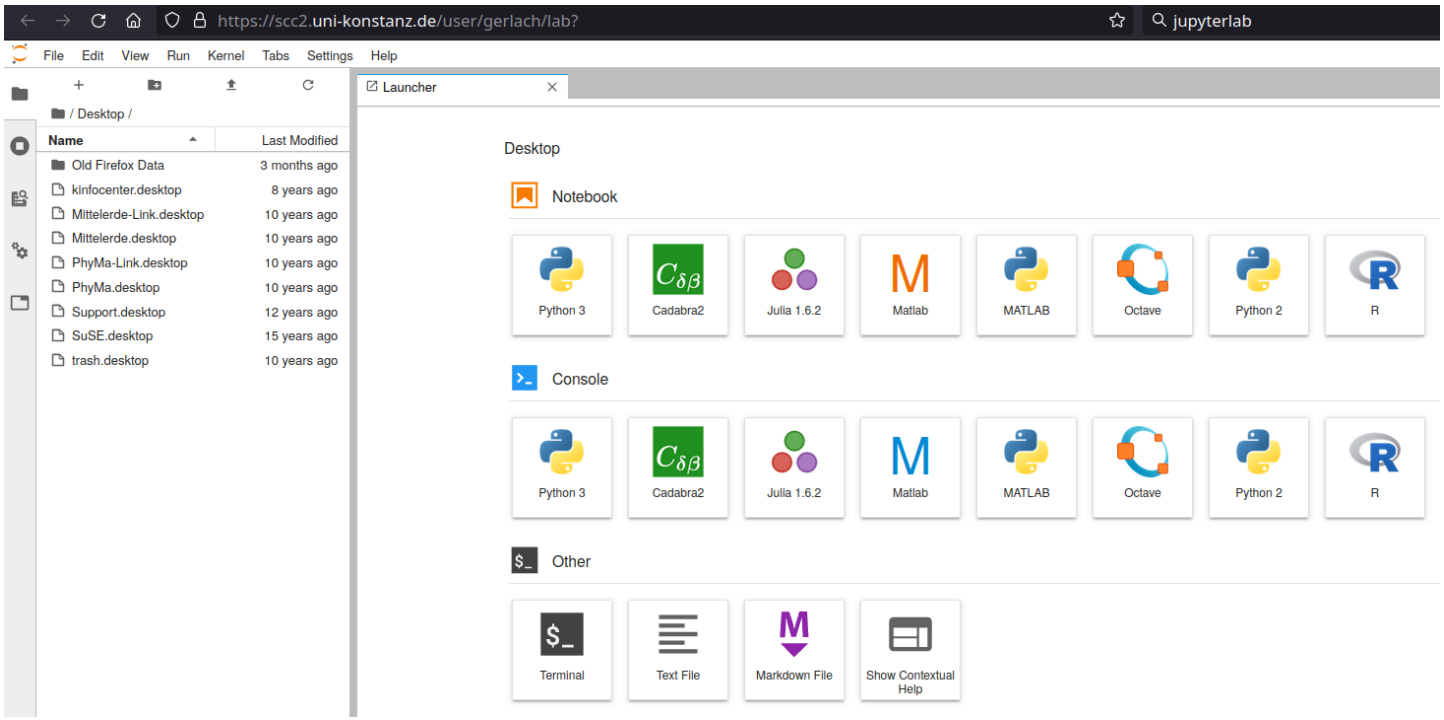


Usage - interaktiv

4 **IDE**: Spyder, PyCharm, Eric, ...

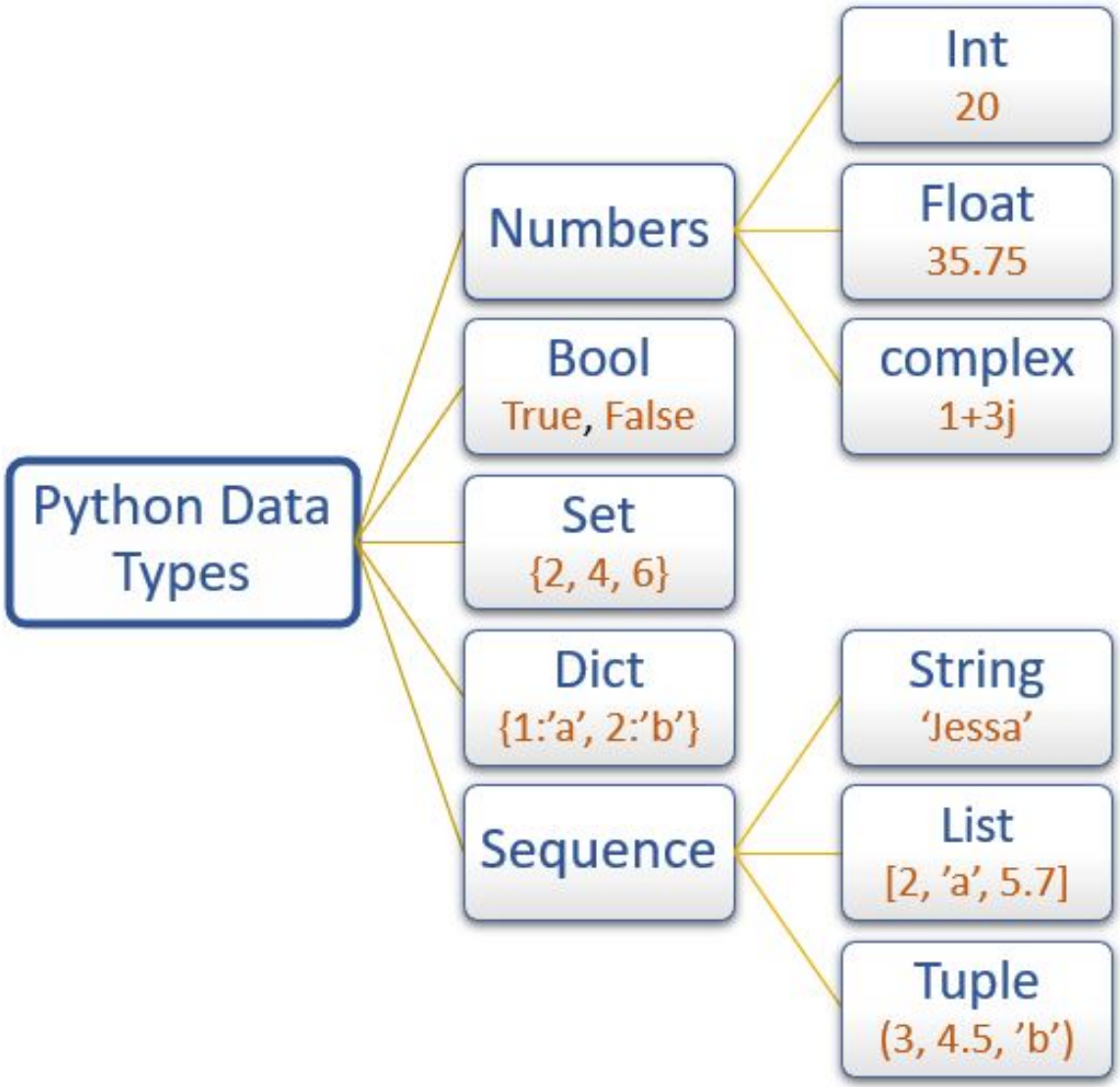
5 **Online (REPL)**: `https://www.python.org/shell/`, `https://repl.it/languages/python3`

6 **Jupyterlab/Jupyterhub**: `jupyter lab`, Online (SCCKN)



7 From C/C++ (CYTHON, pybind11, ...)

Datatypes in Python



Python Basics 1

Python 3 Beginner's Reference Cheat Sheet

Alvaro Sebastian
http://www.sixthresearcher.com

Main data types

boolean = *True* / *False*
integer = 10
float = 10.01
string = "123abc"
list = [value1, value2, ...]
dictionary = { key1:value1, key2:value2, ...}

Numeric operators

+	addition
-	subtraction
*	multiplication
/	division
**	exponent
%	modulus
//	floor division

Comparison operators

==	equal
!=	different
>	higher
<	lower
>=	higher or equal
<=	lower or equal

Boolean operators

and	logical AND
or	logical OR
not	logical NOT

Special characters

#	coment
\n	new line
\<char>	scape char

String operations

string[i]	retrieves character at position i
string[-1]	retrieves last character
string[i:j]	retrieves characters in range i to j

List operations

list = []	defines an empty list
list[i] = x	stores x with index i
list[i]	retrieves the item with index I
list[-1]	retrieves last item
list[i:j]	retrieves items in the range i to j
del list[i]	removes the item with index i

Dictionary operations

dict = {}	defines an empty dictionary
dict[k] = x	stores x associated to key k
dict[k]	retrieves the item with key k
del dict[k]	removes the item with key k

String methods

string.upper()	converts to uppercase
string.lower()	converts to lowercase
string.count(x)	counts how many times x appears
string.find(x)	position of the x first occurrence
string.replace(x,y)	replaces x for y
string.strip(x)	returns a list of values delimited by x
string.join(L)	returns a string with L values joined by string
string.format(x)	returns a string that includes formatted x

List methods

list.append(x)	adds x to the end of the list
list.extend(L)	appends L to the end of the list
list.insert(i,x)	inserts x at i position
list.remove(x)	removes the first list item whose value is x
list.pop(i)	removes the item at position i and returns its value
list.clear()	removes all items from the list
list.index(x)	returns a list of values delimited by x
list.count(x)	returns a string with list values joined by S
list.sort()	sorts list items
list.reverse()	reverses list elements
list.copy()	returns a copy of the list

Dictionary methods

dict.keys()	returns a list of keys
dict.values()	returns a list of values
dict.items()	returns a list of pairs (key,value)
dict.get(k)	returns the value associtated to the key k
dict.pop()	removes the item associated to the key and returns its value
dict.update(D)	adds keys-values (D) to dictionary
dict.clear()	removes all keys-values from the dictionary
dict.copy()	returns a copy of the dictionary

Legend: x,y stand for any kind of data values, **s** for a string, **n** for a number, **L** for a list where **i,j** are list indexes, **D** stands for a dictionary and **k** is a dictionary key.

Python 3 Beginner's Reference Cheat Sheet

Alvaro Sebastian
http://www.sixthresearcher.com

Built-in functions

print(x, sep='y')	prints x objects separated by y
input(s)	prints s and waits for an input that will be returned
len(x)	returns the length of x (s, L or D)
min(L)	returns the minimum value in L
max(L)	returns the maximum value in L
sum(L)	returns the sum of the values in L
range(n1,n2,n)	returns a sequence of numbers from n1 to n2 in steps of n
abs(n)	returns the absolute value of n
round(n1,n)	returns the n1 number rounded to n digits
type(x)	returns the type of x (string, float, list, dict ...)
str(x)	converts x to string
list(x)	converts x to a list
int(x)	converts x to a integer number
float(x)	converts x to a float number
help(s)	prints help about x
map(function, L)	Applies function to values in L

Conditional statements

```
if <condition> :
    <code>
else if <condition> :
    <code>
...
else:
    <code>

if <value> in <list>:
```

Data validation

```
try:
    <code>
except <error>:
    <code>
else:
    <code>
```

Working with files and folders

```
import os
os.getcwd()
os.makedirs(<path>)
os.chdir(<path>)
os.listdir(<path>)
```

Loops

```
while <condition>:
    <code>

for <variable> in <list>:
    <code>

for <variable> in
range(start,stop,step):
    <code>

for key, value in
dict.items():
    <code>
```

Loop control statements

break	finishes loop execution
continue	jumps to next iteration
pass	does nothing

Running external programs

```
import os
os.system(<command>)
```

Functions

```
def function(<params>):
    <code>
    return <data>
```

Modules

```
import module
module.function()

from module import *
function()
```

Reading and writing files

```
f = open(<path>,'r')
f.read(<size>)
f.readline(<size>)
f.close()

f = open(<path>,'r')
for line in f:
    <code>
f.close()

f = open(<path>,'w')
f.write(<str>)
f.close()
```

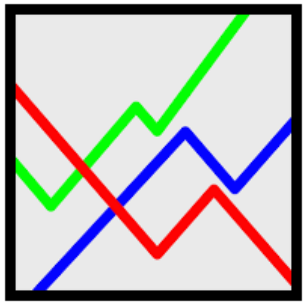
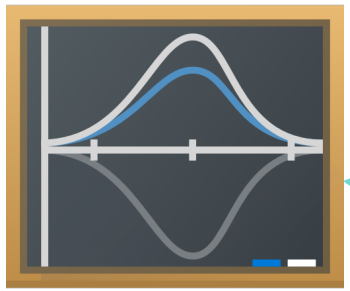
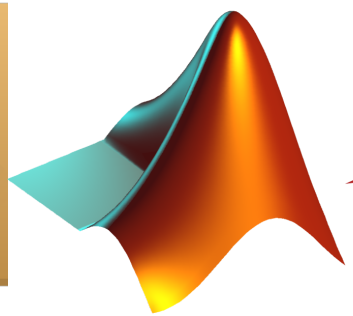
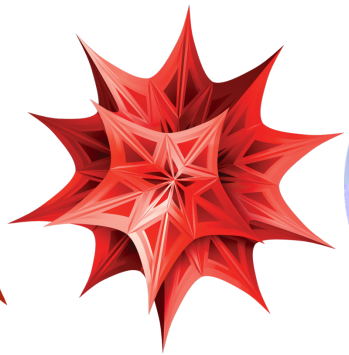
Legend: x,y stand for any kind of data values, **s** for a string, **n** for a number, **L** for a list where **i,j** are list indexes, **D** stands for a dictionary and **k** is a dictionary key.

Summary - What is Python

- Open-Source, all platforms, portable
- High-level (further away from machine language, and closer to natural languages)
- Interactive (Terminal, IPython, Notebook)
- Interpreted (Scripting)
- General purpose
- Multiparadigm (Procedural, Functional, OO, ..)
- Minimalistic constructs
- Structured code (indentation)
- Dynamically typed
- Built-in advanced types (string, dictionary, list, ..)
- Powerful standard library and many extensions
- Widely used, excellent support/documentation

Plotting in Science

- **Dedicated Apps**: Interactive data analysis and visualization
  gnuplot, Grace, LabPlot, Origin, Igor Pro, ...
- **Frameworks**:
  Matlab, Octave, Mathematica, ...
- Specialized for **3D/Rendering**:
  Povray, Paraview, VMD, ...
- In **C/C++**: GUI (Qt, ...) + Plotting libaries
- In **Python**: Matplotlib (Plotly, Seaborn, Pandas, ...)

matplotlib

Matplotlib - Plotting

- Python Graphics library, since 2003, BSD license
- **Backends**: Qt5, Gtk, Tk, wxWidgets, ...

```
1 import matplotlib as mp
2 mp.use("Qt5Agg")
```

- **Export** to PDF, EPS, PNG, SVG, ...
- `pyplot` module: "MATLAB like" interface
- `pylab` (procedural) vs. `pyplot` (OO) interface

```
1 %import pylab as pl
2 import matplotlib.pyplot as pl
```

- Several **toolkits**: seaborn, mplot3d

Matplotlib - Usage

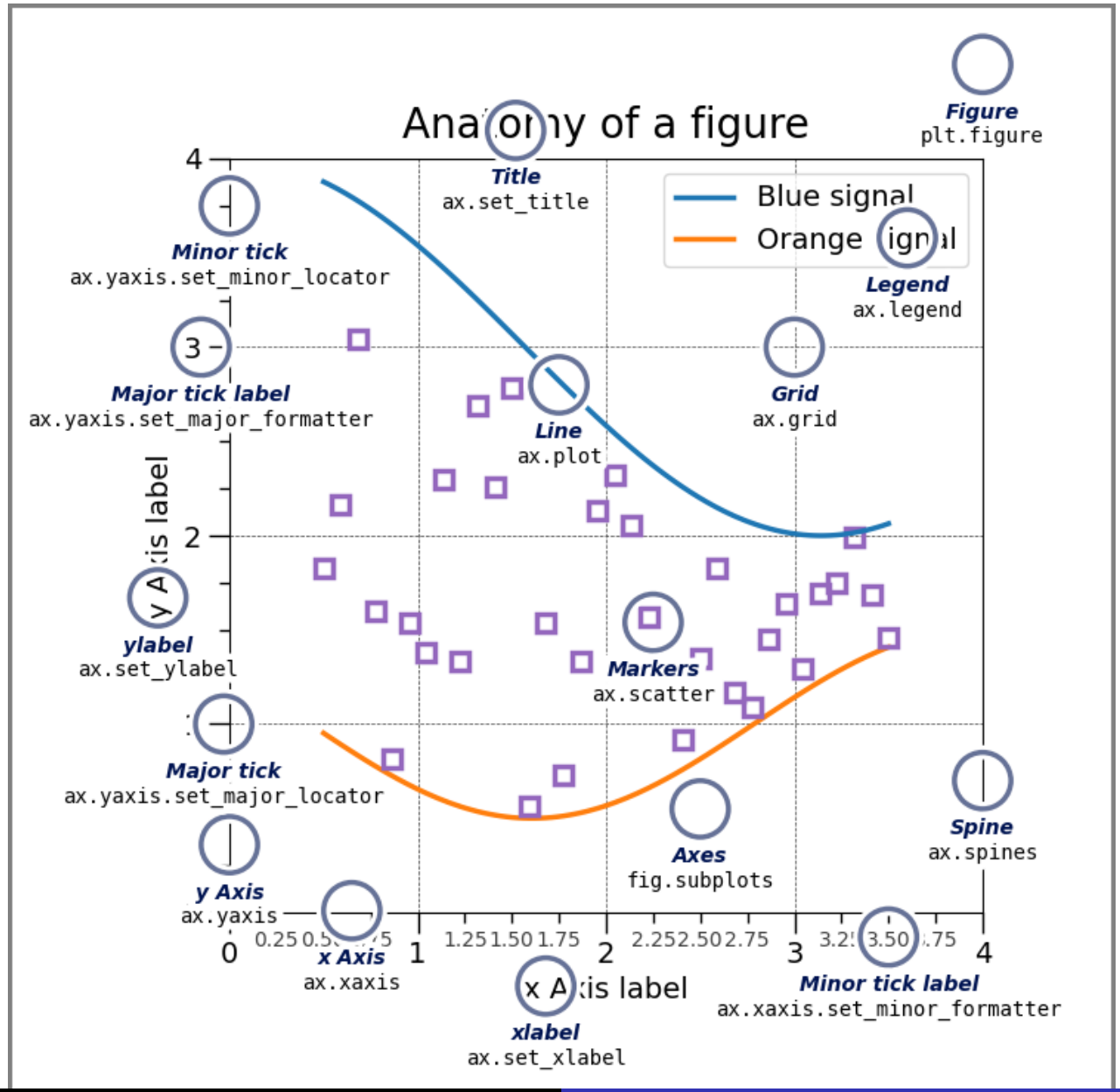
In Script:

```
1  import matplotlib.pyplot as pl
2  from numpy import linspace
3
4  fig, ax = pl.subplots()
5
6  x = linspace(0, 7, 100)
7  #x = arange(0, 7, 0.1)
8  ax.plot(x, x**2)
9  pl.show()
10 #pl.tight_layout()
11 #pl.savefig('plot.png', format='png', dpi=300)
```

In Notebook:

```
$ ipython3 notebook
In[1]: from numpy import arange
In[2]: import matplotlib.pyplot as pl
In[3]: x=arange(0,10)
In[4]: pl.plot(x**2);
```

Matplotlib - Plot



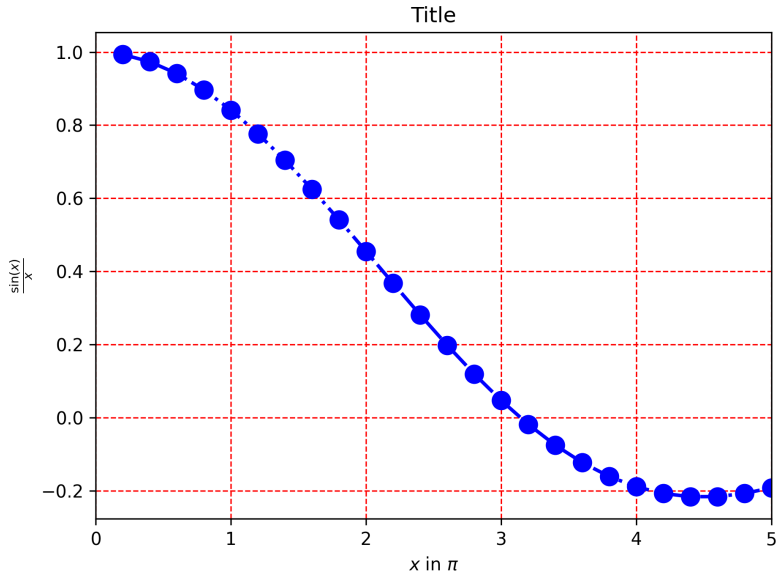
Matplotlib - Styling

Label:

```
1 ax.set_title("Title")
2 ax.set_xlabel("x in $\pi$")
3 ax.set_ylabel("$\\frac{\sin(\pi x)}{x}$")
```

Styles:

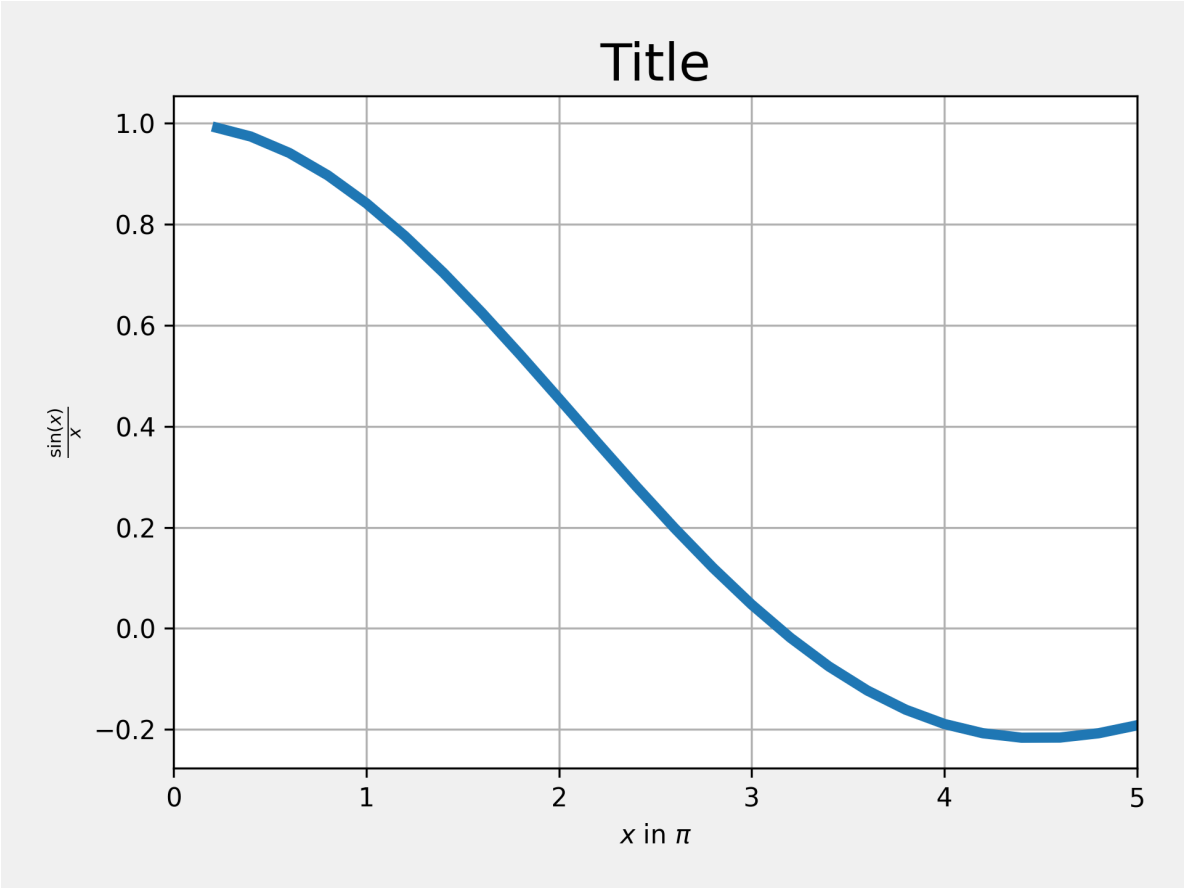
```
1 x = arange(0, 7, 0.5)
2 ax.set_xlim(0, 5)
3 ax.grid(color='r', linestyle='--')
4 ax.plot(x, sin(x)/x, 'bo-.', linewidth = 2, markersize = 10)
```



Matplotlib - Styling

Pre-defined styles:

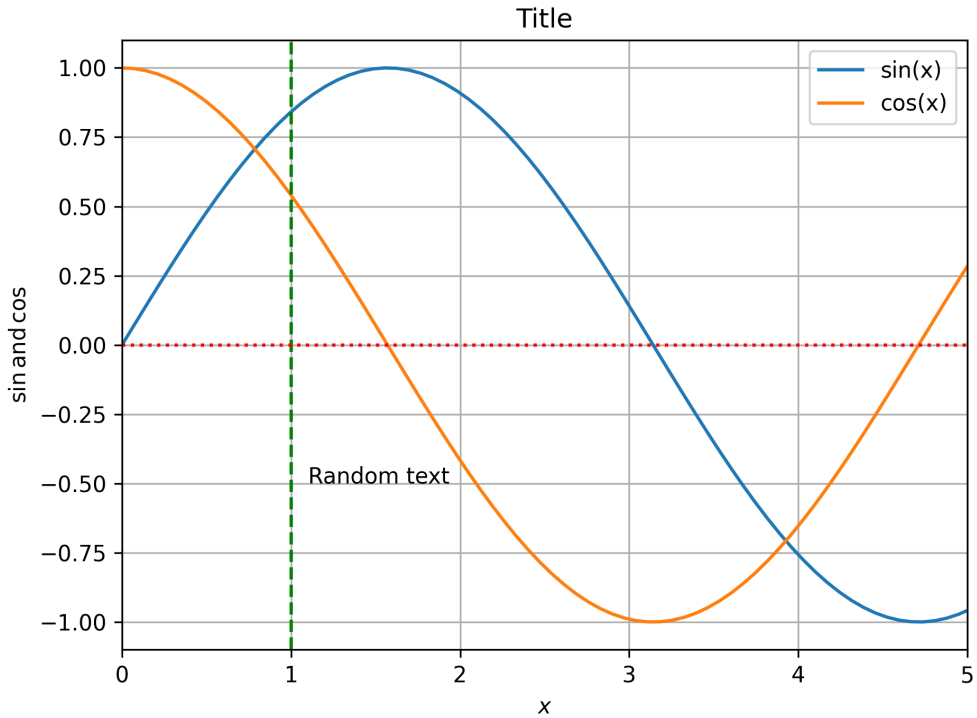
```
1 pl.style.use('fivethirtyeight')
```



see: `https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html`

Matplotlib - Legend and more

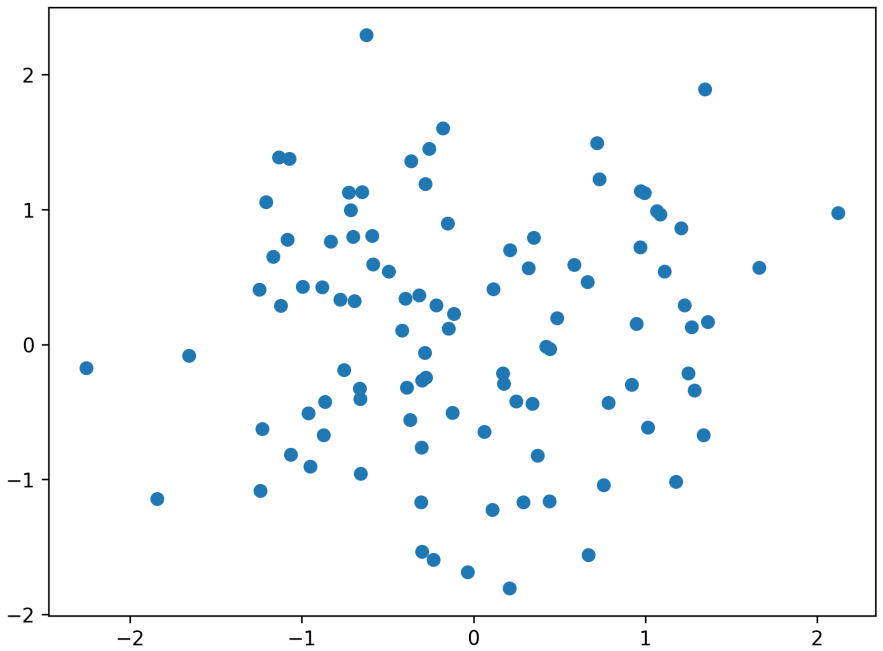
```
1  ax.plot(x, y1, label = "sin(x)")
2  ax.plot(x, y2, label = "cos(x)")
3  ax.legend(loc = 'best')
4
5  pl.text(1.5,-.5,'Random text')
6  pl.axvline(x=1, color ='g', linestyle='--')
7  pl.axhline(y=0, color ='r', linestyle=':')
```



Matplotlib - Plot types

Plot types:

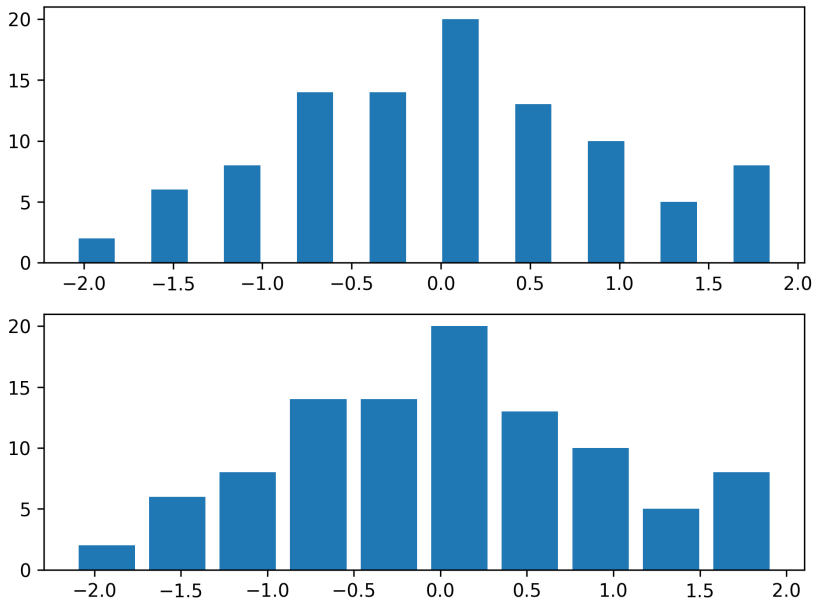
```
1 x = random.normal(mu, sigma, 100)
2 y = random.normal(mu, sigma, 100)
3 ax.scatter(x, y)
4 #pl.bar(...)
5 #ax.hist(x, 10, rwidth=0.9)
6 #pl.pie(...)
```



see https://matplotlib.org/stable/plot_types/index.html

Matplotlib - Subplots

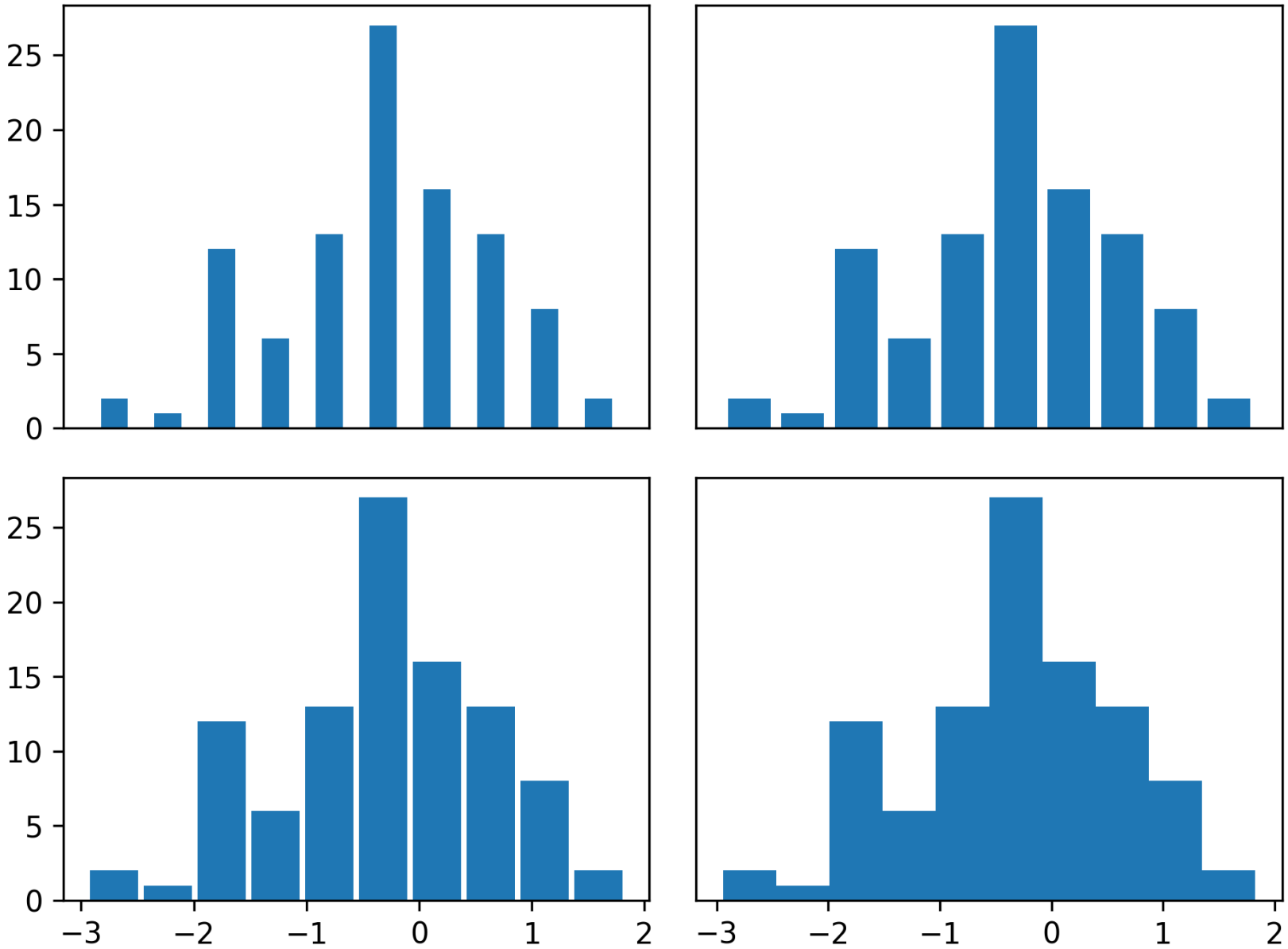
```
1  fig = pl.figure()
2  ax1 = fig.add_subplot(211)
3  ax1.hist(x, 10, rwidth=0.5)
4
5  ax2 = fig.add_subplot(212)
6  ax2.hist(x, 10, rwidth=0.8)
```



Matplotlib - Subplots

```
1  fig = pl.figure()
2  ax1 = fig.add_subplot(221)
3  ax1.hist(x, 10, rwidth=0.5)
4  ax1.get_xaxis().set_visible(False)
5
6  ax2 = fig.add_subplot(222)
7  ax2.hist(x, 10, rwidth=0.8)
8  ax2.get_xaxis().set_visible(False)
9  ax2.get_yaxis().set_visible(False)
10 #ax2.grid(True)
11
12 ax3 = fig.add_subplot(223, sharex=ax1)
13 ax3.hist(x, 10, rwidth=0.9)
14
15 ax4 = fig.add_subplot(224, sharex=ax2)
16 ax4.hist(x, 10)
17 ax4.get_yaxis().set_visible(False)
```

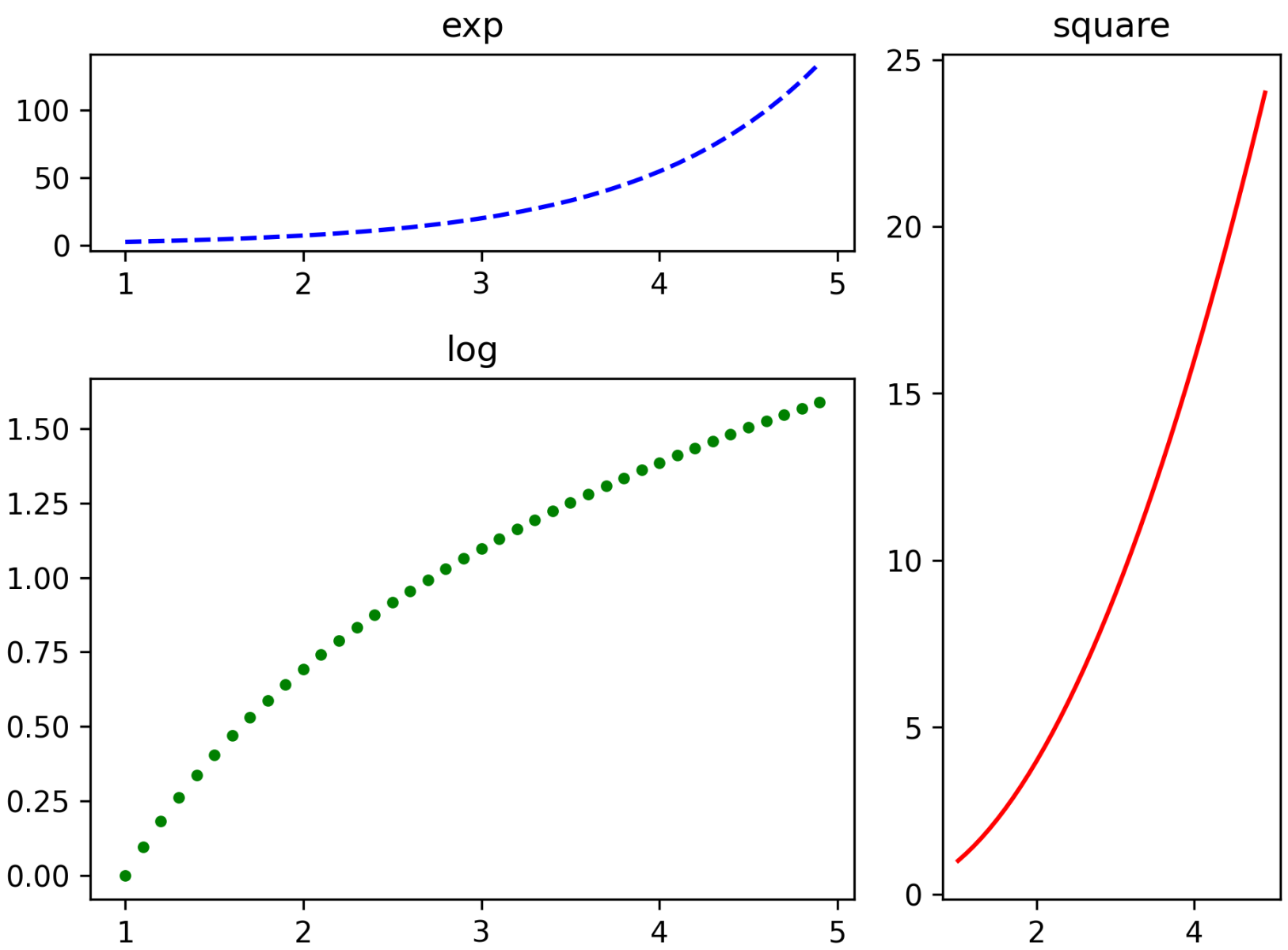
Matplotlib - Subplots



Matplotlib - Subplots Grid

```
1  a1 = pl.subplot2grid((3,3), (0,0), colspan = 2)
2  a2 = pl.subplot2grid((3,3), (0,2), rowspan = 3)
3  a3 = pl.subplot2grid((3,3), (1,0), rowspan = 2, colspan = 2)
4
5  x = arange(1, 5, .1)
6  a1.plot(x, exp(x),'b--')
7  a1.set_title('exp')
8  a2.plot(x, x*x,'r')
9  a2.set_title('square')
10 a3.plot(x, log(x),'g.')
11 a3.set_title('log')
```

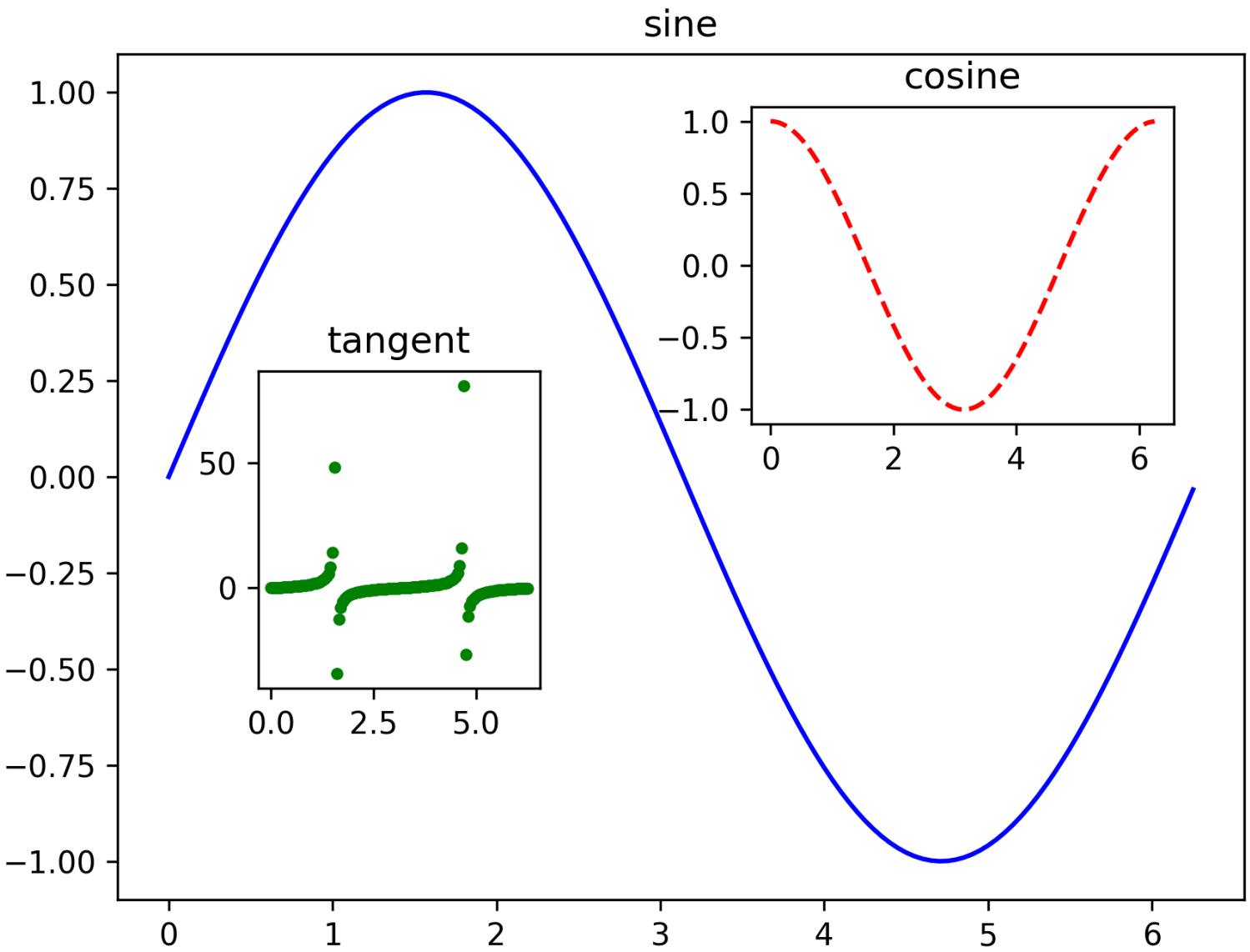
Matplotlib - Subplots Grid



Matplotlib - Inset

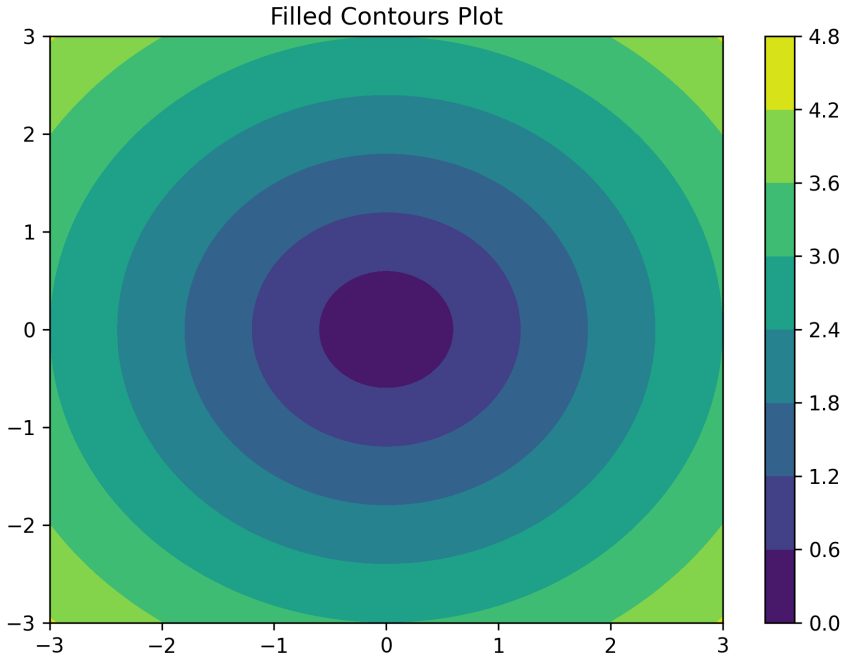
```
1  fig = pl.figure()
2  axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
3  axes2 = fig.add_axes([0.55, 0.55, 0.3, 0.3]) # inset axes
4  axes3 = fig.add_axes([0.2, 0.3, 0.2, 0.3]) # inset axes
5
6  x = arange(0, 2*pi, 0.05)
7  axes1.plot(x, sin(x), 'b')
8  axes2.plot(x, cos(x), 'r--')
9  axes3.plot(x, tan(x), 'g.')
10 axes1.set_title('sine')
11 axes2.set_title("cosine")
12 axes3.set_title("tangent")
```

Matplotlib - Inset



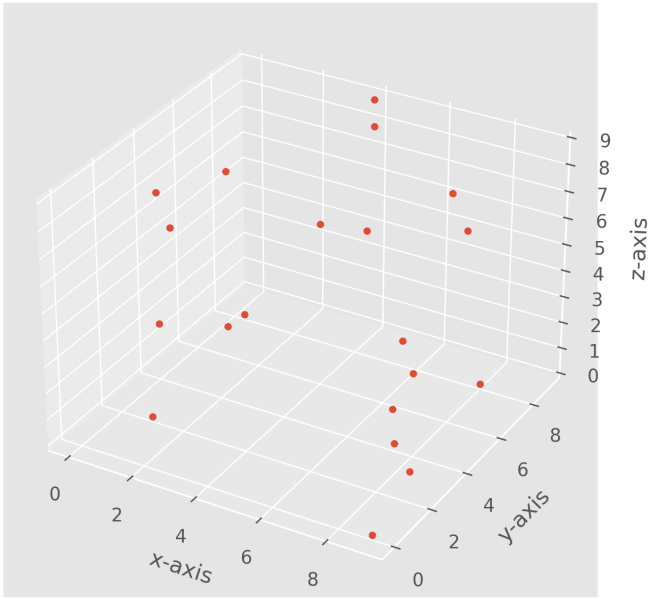
Matplotlib - Contour Plots

```
1  xlist = linspace(-3.0, 3.0, 100)
2  ylist = linspace(-3.0, 3.0, 100)
3  X, Y = meshgrid(xlist, ylist)
4
5  Z = sqrt(X**2 + Y**2)
6
7  fig,ax = pl.subplots(1,1)
8  cp = ax.contourf(X, Y, Z)
9  fig.colorbar(cp)
```



Matplotlib - 3D Plots

```
1  fig = pl.figure()
2  ax = fig.add_subplot(111, projection='3d')
3
4  x = random.randint(0, 10, size = 20)
5  y = random.randint(0, 10, size = 20)
6  z = random.randint(0, 10, size = 20)
7
8  pl.plot(x,y,z, '.')
9
10 ax.set_zlabel('z-axis')
```



Matplotlib - Pictures

```
1  S=pl.imread("ising.png")
2
3  print(S.shape)
4  print(S.dtype)
5  print(S.max())
6  print(S.min())
7
8  pl.axis('off')
9  pl.imshow(S, cmap='gray')
10 #pl.matshow(S, 0)
11
12 pl.show()
```

`imread()`: PNG images are returned as float arrays (0-1). All other formats are returned as int arrays, with a bit depth determined by the file's contents.

better: **Python Imaging Library**

```
1  from PIL import Image
2  # 'P'-pixel, 'L'-greyscale, 'RGB', 'CMYK'
3  S=np.array(Image.open('ising.png').convert('L'))
```

see `https://matplotlib.org/stable/gallery/index`

Matplotlib - Animations

```
1  from matplotlib.animation import FuncAnimation
2
3  fig, ax = pl.subplots()
4  xdata, ydata = [], []
5  ln, = pl.plot([], [], 'ro')
6
7  ax.set_xlim(0, XMAX)
8
9  def update(frame):
10     xdata.append(frame)
11     ydata.append(sin(frame)/frame)
12     ln.set_data(xdata, ydata)
13     return ln,
14
15 ani = FuncAnimation(fig, update,
16                      frames=linspace(0, XMAX, 128),
17                      blit=True)
```

see also double_pendulum.py

Making your plots accessible

- Always make sure your **text is large** enough to read. Use the fontsize parameter in `xlabel`, `ylabel`, `title`, and `legend`, and `tick_params` with `labelsize` to increase the text size of the numbers on your axes.
- Make your **graph elements easy to see**. Use `s` to increase the size of your scatterplot markers and `linewidth` to increase the sizes of your plot lines.
- Using only color to distinguish between different plot elements is unreadable to colorblind people (or in black-and-white). Use **different** `linestyle` **for lines and marker for symbols**. If unsure, use Coblis or Color Oracle to check how your plots would look like to colorblind.