# MATPLOTLIB

## 1.importing library

- matplotlib is preinstalled with pyzo

  import matplotlib.pyplot as plt

- possible dependencies:
  - **numpy** - extension for arrays, matrices,...
  - **libpng** - loading and saving png files
  - **dateutil** - datetime handling
  - **tk/pyqt** - GUI frameworks
- possible dependecies - cartography

- tutorials
  - http://matplotlib.org/users/beginner.html

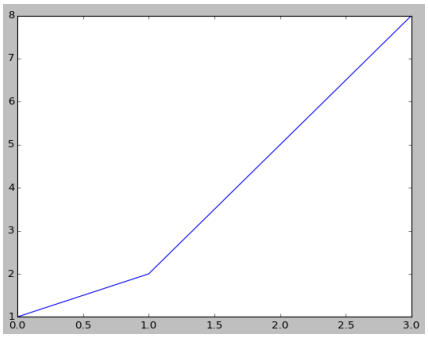
## 2. pyplot basics- linecharts / scatterplot example and basic commands

basic example - linechart

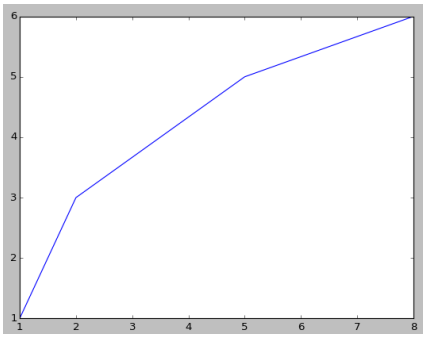
```
import matplotlib.pyplot as plt

plt.plot([1,2,3,7])
plt.ylabel('some numbers')
plt.show()
```

pyplot.**plot()** - plots lines/markers to the axes

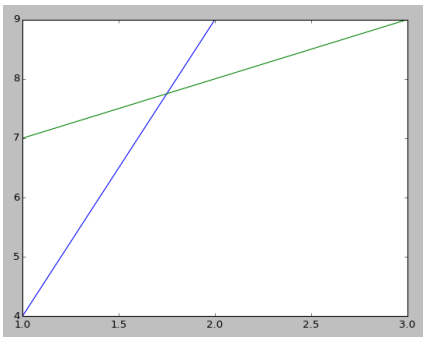
plt.plot([1, 2, 5, 8])



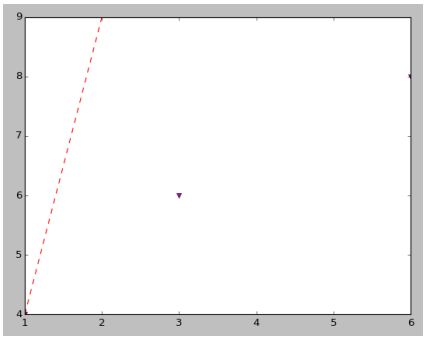

plt.plot([1, 2, 5, 8], [1,3,5,6])



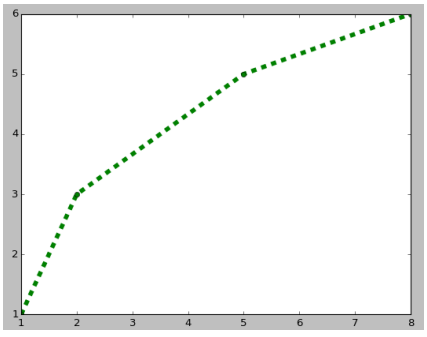

plt.plot([1, 2], [4,9], [1,3], [7,9])




plt.plot([1, 2], [4,9], ‘r--’, [1,3, 6], [4, 6, 8], ‘mv’) # red dashed lines, magenta triangles




plt.plot([1, 2, 5, 8], [1,3,5,6], linestyle=’dashed’, color=’green’, marker=’o’, linewidth=’5’)

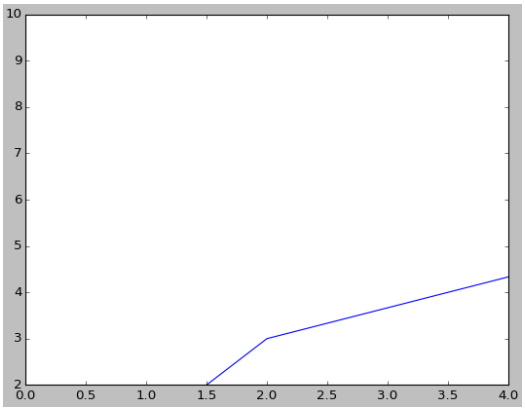


| character | description |
|---|---|
| '-' | solid line style |
| '--' | dashed line style |
| '-.' | dash-dot line style |
| ':' | dotted line style |
| '.' | point marker |
| ',' | pixel marker |
| 'o' | circle marker |
| 'v' | triangle_down marker |
| '^' | triangle_up marker |
| '<' | triangle_left marker |
| '>' | triangle_right marker |
| '1' | tri_down marker |
| '2' | tri_up marker |
| '3' | tri_left marker |
| '4' | tri_right marker |
| 's' | square marker |
| 'p' | pentagon marker |
| '*' | star marker |
| 'h' | hexagon1 marker |
| 'H' | hexagon2 marker |
| '+' | plus marker |
| 'x' | x marker |
| 'D' | diamond marker |
| 'd' | thin_diamond marker |
| '\|' | vline marker |
| '_' | hline marker |

| character | color |
|---|---|
| 'b' | blue |
| 'g' | green |
| 'r' | red |
| 'c' | cyan |
| 'm' | magenta |
| 'y' | yellow |
| 'k' | black |
| 'w' | white |

pyplot.**axis** - setting the extent of the axis

```
plt.plot([1, 2, 5, 8], [1,3,5,6])
plt.axis([0, 4, 4, 50])
```



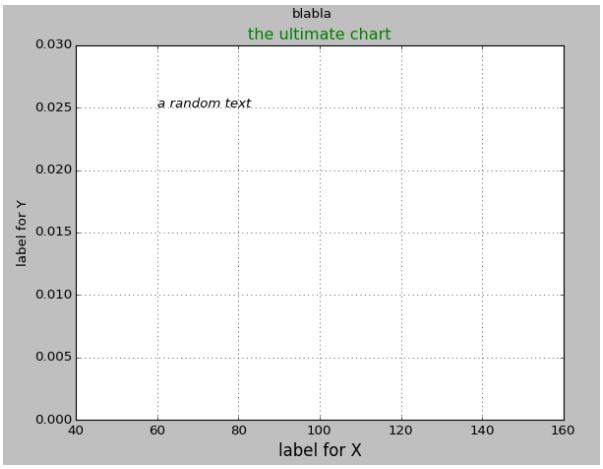

pyplot.**show** - displays a figure

pyplot.**setp** - controlling properties

```
lines = plt.plot([1, 2, 5, 8], [1,3,5,6])
plt.setp(lines, color='r')
```

pyplot.**xlabel**/pyplot.**ylabel** - name of the axis
pyplot.**title -** sets title
pyplot.**suptitle** - add a title to the Figure
pyplot.**figtext** - add text at an arbitrary location to the Figure
pyplot.**text** - put title on selected position
pyplot.**grid** - switch grid on and off

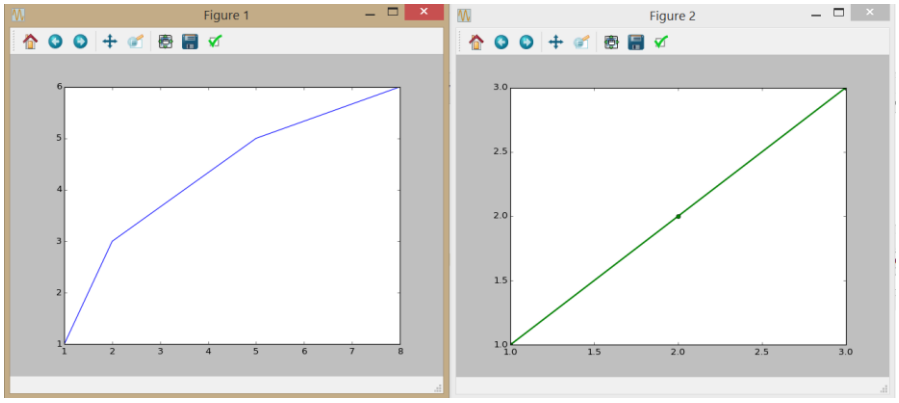
```
import matplotlib.pyplot as plt

plt.xlabel('label for X', fontsize=15)
plt.ylabel('label for Y')
plt.title('the ultimate chart', color='green')
plt.suptitle('blabla')
plt.text(60, .025, r'a random text', style='italic')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```



| Property | Value Type |
|---|---|
| alpha | float |
| backgroundcolor | any matplotlib color |
| bbox | rectangle prop dict plus key 'pad' which is a pad in points |
| clip_box | a matplotlib.transform.Bbox instance |
| clip_on | [True \| False] |
| clip_path | a Path instance and a Transform instance, a Patch |
| color | any matplotlib color |
| family | [ 'serif' \| 'sans-serif' \| 'cursive' \| 'fantasy' \| 'monospace' ] |
| fontproperties | a matplotlib.font_manager.FontProperties instance |
| horizontalalignment or ha | [ 'center' \| 'right' \| 'left' ] |
| label | any string |
| linespacing | float |
| multialignment | ['left' \| 'right' \| 'center' ] |
| name or fontname | string e.g., ['Sans' \| 'Courier' \| 'Helvetica' ...] |
| picker | [None\|float\|boolean\|callable] |
| position | (x,y) |
| rotation | [ angle in degrees 'vertical' \| 'horizontal' |
| size or fontsize | [ size in points \| relative size, e.g., 'smaller', 'x-large' ] |
| style or fontstyle | [ 'normal' \| 'italic' \| 'oblique'] |
| text | string or anything printable with '%s' conversion |
| transform | a matplotlib.transform transformation instance |
| variant | [ 'normal' \| 'small-caps' ] |
| verticalalignment or va | [ 'center' \| 'top' \| 'bottom' \| 'baseline' ] |
| visible | [True \| False] |
| weight or fontweight | [ 'normal' \| 'bold' \| 'heavy' \| 'light' \| 'ultrabold' \| 'ultralight'] |
| x | float |
| y | float |
| zorder | any number |

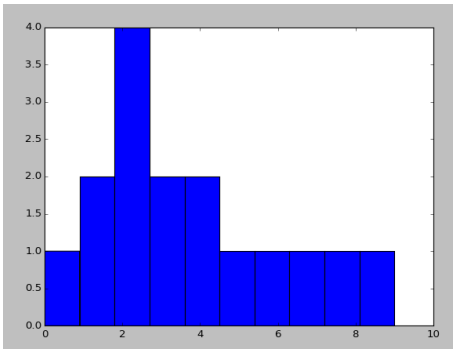
## 3. handling more figures

```
import matplotlib.pyplot as plt

plt.figure(1)
plt.plot([1, 2, 5, 8], [1,3,5,6])

plt.figure(2)
plt.plot([1,2,3], [1,2,3], 'go-', label='line 1', linewidth=2)

plt.show()
```

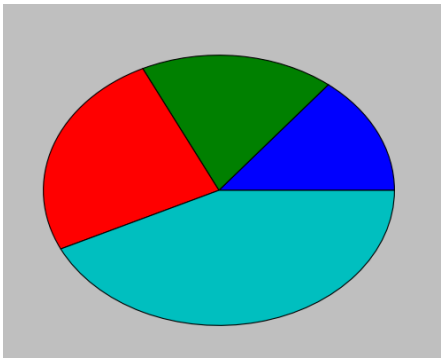


## 4. other types of plots

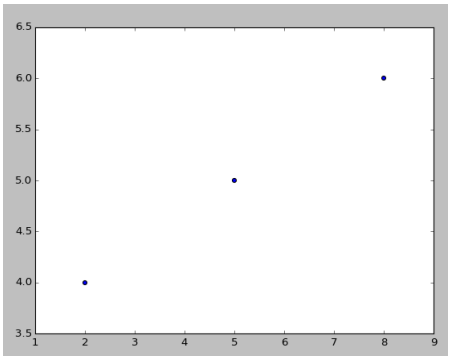
plt.hist([0,2,1,2,6,7,4,8,9,5,4,1,3,2,3,2], bins=10)



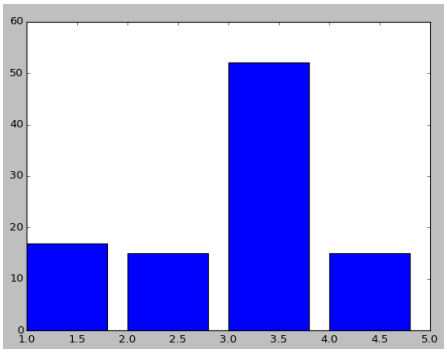

plt.pie([4,5,7,12])



plt.scatter([2,5,8],[4,5,6])

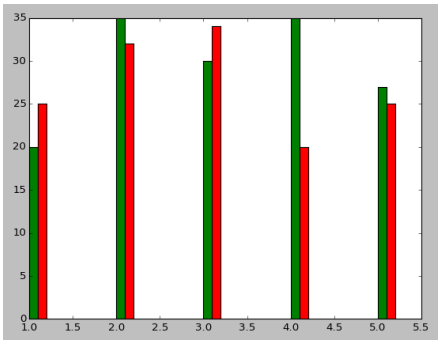



plt.bar([1, 2, 3, 4],[17, 15, 52, 15])




plt.bar([1,2,3,4,5], [20, 35, 30, 35, 27], 0.1, color='g')

plt.bar([1.1,2.1,3.1,4.1,5.1], [25, 32, 34, 20, 25], 0.1, color='r')



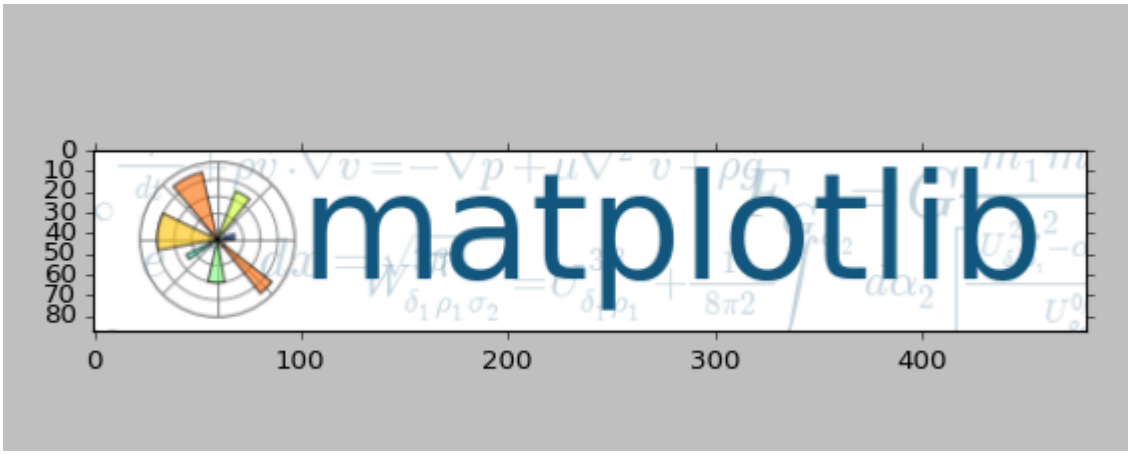
…

online collection of documented examples
http://matplotlib.org/api/pyplot_api.html

## 5. Loading external images:

- using matplotlib.image

```
# request image from url and load it with matplotlib
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import urllib.request

with urllib.request.urlopen("http://matplotlib.sourceforge.net/_static/logo2.png") as response:
        img = plt.imread(response)
        plt.imshow(img)
```




tutorial http://matplotlib.org/users/image_tutorial.html

## 6. Example- choropleth map

loading external geojson:

```
import matplotlib.pyplot as plt
import json

with open ('C:/pathto/countries.geojson', 'r', encoding='utf8') as countriesJson:
	countries = json.load(countriesJson)
```

putting shapes on matplotlib canvas:

```
for country in countries['features']:
	for shape in country['geometry']['coordinates']:
		for coords in shape:

			x = [i for i,j in coords]
			y = [j for i,j in coords]

			plt.plot(x, y)

plt.axis('scaled')
plt.grid(True)
plt.show()
```

defining colormap:

- http://matplotlib.org/examples/color/colormaps_reference.html

```
from matplotlib.pyplot import cm

colors = plt.get_cmap('copper', 6)
```

getting number from attribute:

```
income = int(country['properties']['income_grp'][0])
```

coloring polygons:

- http://stackoverflow.com/questions/10550477/how-do-i-set-color-to-rectangle-in-matplotlib

```
plt.fill(x,y, facecolor=colors(income), edgecolor='black', linewidth=0.4)
```

event handling

- https://jakevdp.github.io/blog/2012/12/06/minesweeper-in-matplotlib/
- http://matplotlib.org/users/event_handling.html

- setting ax object

```
fig = plt.figure()
ax = fig.add_subplot(111)
```

- global list of polygons with attributes.

```
polygons = []

# http://matplotlib.org/api/patches_api.html
polygon = plt.Polygon(coords)
polygons.append({'shape': polygon, 'props': country['properties'], 'income': income})
ax.add_patch(polygon)
polygon.set_facecolor(colors(income))
```

- setting event listener

```
fig.canvas.mpl_connect('motion_notify_event', on_move)
```

- define on_move function

```
def on_move(event):
	for pol in polygons:
		income = int(pol['props']['income_grp'][0])
		pol['shape'].set_facecolor(colors(income))
	for pol in polygons:
		if pol['shape'].contains_point((event.x, event.y)):
			pol['shape'].set_facecolor('black')
	fig.canvas.draw()
```

labels

- http://matplotlib.org/api/text_api.html

- creating text object

```
text = ax.text(0,0, '', color='r')
```

- redefining on_move function

```
def on_move(event):
	text.set_x(event.x - 300)
	text.set_y(event.y - 250)
	text.set_text('')

	for pol in polygons:
		income = int(pol['props']['income_grp'][0])
		pol['shape'].set_facecolor(colors(income))
	for pol in polygons:
		if pol['shape'].contains_point((event.x, event.y)):
			pol['shape'].set_facecolor('black')
			text.set_text(pol['props']['name'])
	fig.canvas.draw()
```

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import cm
import json

path = 'C:/Users/mu/Desktop/phd/vyuka/Programovani/matplot geojson/'

colors = plt.get_cmap('copper', 6)
polygons = []

def on_move(event):
	text.set_x(event.x - 300)
	text.set_y(event.y - 250)
	text.set_text('')

	for pol in polygons:
		income = int(pol['props']['income_grp'][0])
		pol['shape'].set_facecolor(colors(income))

	for pol in polygons:
		if pol['shape'].contains_point((event.x, event.y)):
			pol['shape'].set_facecolor('black')
			text.set_text(pol['props']['name'])
	fig.canvas.draw()

fig = plt.figure()
ax = fig.add_subplot(111)
text = ax.text(0,0, '', color='r')

with open (path + 'countries.geojson', 'r', encoding='utf8') as countriesJson:
	countries = json.load(countriesJson)

	for country in countries['features']:
		income = int(country['properties']['income_grp'][0])
		for shape in country['geometry']['coordinates']:
			for coords in shape:

			x = [i for i,j in coords]
			y = [j for i,j in coords]
			polygon = plt.Polygon(coords)
			polygons.append({'shape': polygon, 'props': country['properties'], 'income':
			income})
			ax.add_patch(polygon)
			polygon.set_facecolor(colors(income))
```

```
fig.canvas.mpl_connect('motion_notify_event', on_move)
plt.axis('scaled')
plt.grid(True)
plt.show()
```

## 7. Extension - basemap

pip install basemap --allow-external basemap --allow-unverified basemap
(SET GEOS_DIR= “C:\OSGeo4W64\lib”)

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
# setup Lambert Conformal basemap.
# set resolution=None to skip processing of boundary datasets.
m = Basemap(width=12000000,height=9000000,projection='lcc',
            resolution=None,lat_1=45.,lat_2=55,lat_0=50,lon_0=-107.)
m.shadedrelief()
plt.show()
```