Introduction: NumPy & Matplotlib

Jakub Tworzydło

Institute of Theoretical Physics
phone: (022)5532-919, room 5.19
Jakub.Tworzydlo@fuw.edu.pl

1/03/2022 Pasteura, Warsaw

Plan

Plan

1. Plotting with Matplotlib

2. NumPy array, NumPy matrix

Plan

NumPy arrays

Creation of an array is more Python-like than the loop over elements.

```
# Filename: arrays.py
import numpy as np # short name of this module

n = 7
v = np.zeros(n)     # vector is represented as a numpy array

for i in range(n):
    v[i] = i/2.0    # sets elements v_i
print( 'v = ', v )

u = np.arange(-1.,1.,0.2)          # range of values
print( 'u = ', u )

w = np.linspace(-np.pi,np.pi,6) # also some range,
print( 'w = ', w )                 # a given number of points
```

Simplest plot

Plotting is easy, check it with the example below

```
# Filename: plots.py

import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-np.pi, np.pi, 500)
# Numpy math functions operate on whole arrays (!)
y_cos, y_sin = np.cos(x), np.sin(x)

plt.plot(x, y_cos) # plot cosine function
plt.plot(x, y_sin) # plot sine function

plt.show()
```

Enhanced plot

We improve a bit by adding figure size, scale, line styles.

```
# Filename: plots.py
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-np.pi, np.pi, 500)
y_cos, y_sin = np.cos(x), np.sin(x)

# we can set the figure size (in inches)
plt.figure(figsize=(10, 6), dpi=80)
# calculate and set data range
plt.xlim(x.min() * 1.0, x.max() * 1.0)
plt.ylim(y_cos.min() * 1.1, y_cos.max() * 1.1)

plt.plot(x, y_cos, color="b", linewidth=2.5, linestyle="-")
plt.plot(x, y_sin, color="r", linewidth=2.5, linestyle="--"

plt.grid()
plt.show()
```

Plotting with Matplotlib

We can add and manipulate the descriptions!
Add the lines below `plt.figure` command, but before `plt.show` in the previous script. Make sure the code is working.

```
# Figure and axis title
plt.title('Functions: cosine and sine', fontsize=24)
plt.xlabel('x value', fontsize=20)
plt.ylabel('y result', fontsize=20)
# Add legend
plt.legend( ("cos(x)","sin(x)"), loc='upper left',
            fontsize=14 )

# Tell matplotlib to use LaTeX to render text
plt.rc('text', usetex=False)
# Set xticks values and description, modify yticks
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
           ['$-\pi$', '$-\pi/2$', '0', '$\pi/2$', '$\pi$'],
           fontsize=18 )
plt.yticks(ticks=np.arange(-1,1.1,0.5), fontsize=18)
```

Try out options

Experiment yourself by changing the plot and all the options.

Task 1

Using the features introduced above prepare a plot with at least two functions (Bessel?) and including the descriptions. Modify the values given in the example (according to your liking). Save the result.

```
# replace plt.show() with a command to save the plot
plt.savefig("xy_plot.png", format="png")
```

Plan

NumPy matrices

Example below illustrates how to create (element-wise) a vector and a matrix.

```
import numpy as np
n = 5
A = np.zeros((n,n))
x = np.zeros(n)

for i in range(n):
    x[i] = i/2.0            # some vector
    for j in range(n):      # ... and a matrix
        A[i,j] = i + j  + (i+1) % (j+1)

print( x )
print( A )
```

Linear algebra with Numpy

All usual operations are possible. Numpy is highly efficient – relays on LAPACK and BLAS low-level, optimized libraries.

```
b = np.dot(A, x) # matrix vector product: b = A*x

# and we can solve: A*y=b
y = np.linalg.solve(A, b)

# the resulting y is same as x
#      ... up to numerical accuracy
```

Task 2 Write a code, which solves the above linear equation. Calculate the norm of remanent vector `res = x - y` (which is a measure of accuracy) for `n=4000`. Find and use the vector norm function from `NumPy`.

Solving a problem on your own

Task 3

Find a single command in NumPy package, which allows you to find `N = 10000` random numbers from a Gaussian distribution. Produce the data with a given `mean = 15` and a given `simga = 100`. Consult the Matplotlib example to plot the histogram of this data (command hist in matplotlib.pyplot).

A) Calculate mean and standard deviation of data obtained above. Format and print the results on your plot (e.g. in the title). Add description to the plot (e.g. name of the axis), plot the Gaussian distribution, print a legend etc.

B) Calculate an array `x` of `N` random samples uniformly distributed in the interval `[-1,1]`. Sum `P = 10` of such arrays (element by element) and rescale the result (named `xsum`):

```
xnew = mean + sigma * xsum * np.sqrt(3./P)
```

Plot the histogram of new rescaled data. Does it look the same?