https://www.halvorsen.blog

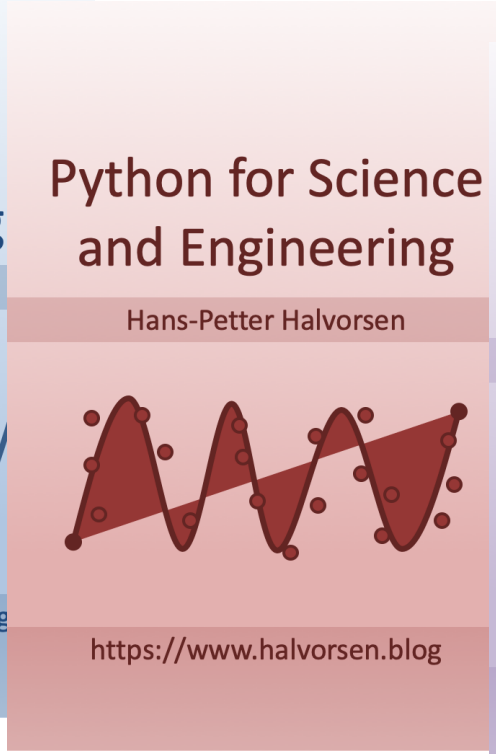
Plotting with Python

Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

https://www.halvorsen.blog/documents/programming/python/

Additional Python Resources



https://www.halvorsen.blog/documents/programming/python/

Contents

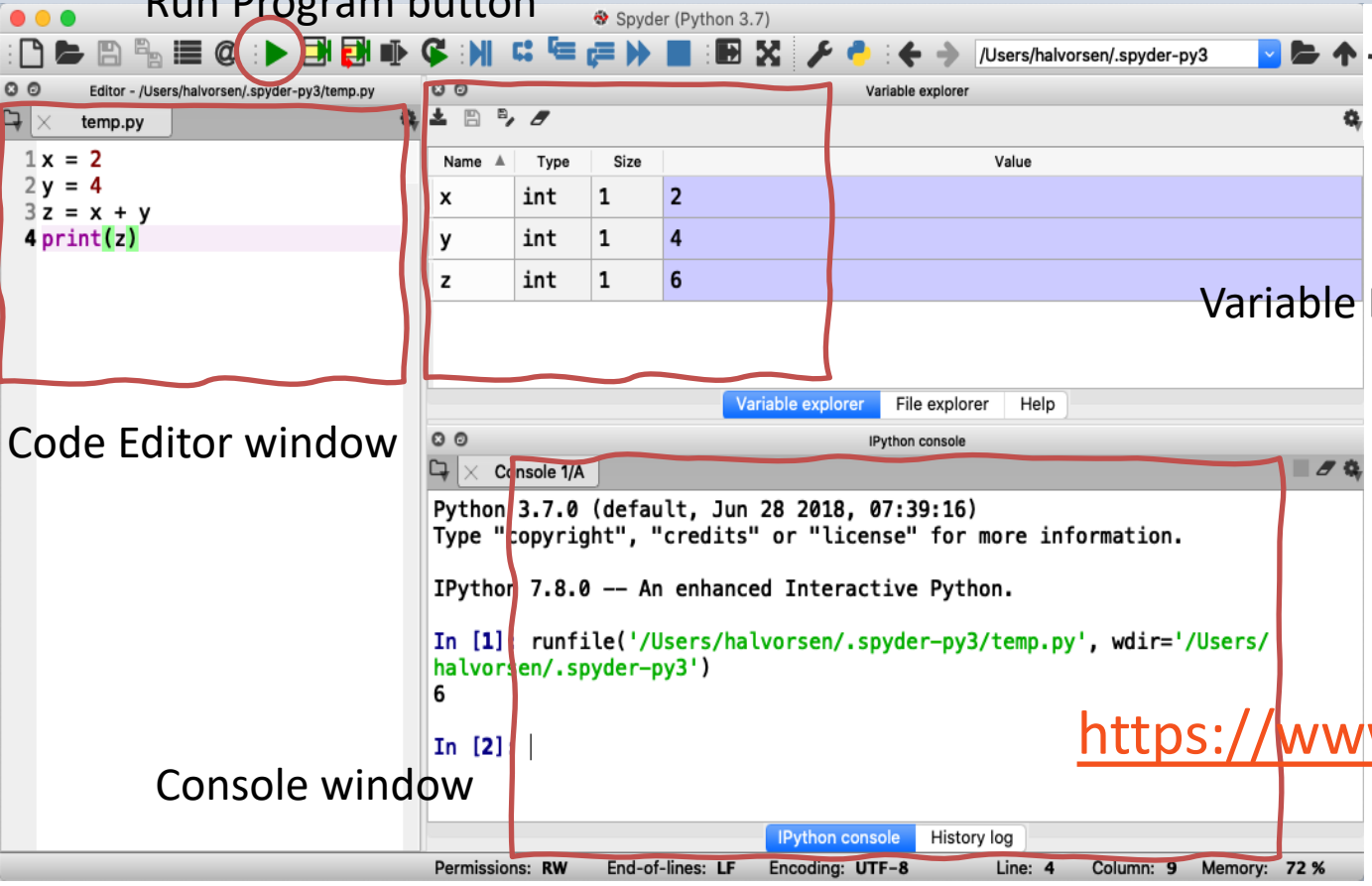
- Python Editors
- Python Libraries/Packages
- Plotting with Matplotlib

Python Editors

- Python IDLE
- **Spyder** (Anaconda distribution)
- PyCharm
- **Visual Studio Code**
- Visual Studio
- Jupyter Notebook
- …

Spyder (Anaconda distribution)

Run Program button



SPYDER
The Scientific Python Development Environment

Variable Explorer window

Code Editor window

Console window

https://www.anaconda.com

Python Packages/Libraries

- Rather than having all its functionality built into its core, Python was designed to be highly extensible.
- This approach has advantages and disadvantages.
- A disadvantage is that you need to install these packages separately and then later import these modules in your code.
- Some important packages are:
  - **NumPy** - NumPy is the fundamental package for scientific computing with Python
  - **Matplotlib** – With this library you can easily make plots in Python

Installing Packages/Libraries

- If you have installed Python using the **Anaconda distribution**, all the most used Python Packages/Libraries are included (NumPy, Matplotlib, +++)
- Else, you typically use **PIP** to install Python packages

PIP

- PIP is a Package Manager for Python packages/modules
- With PIP you can download and install Python packages/modules from the Python Package Index (PyPI)
- What is a Package? A package contains all the files you need for a module. Modules are Python code libraries you can include in your project.
- The Python Package Index (PyPI) is a repository of Python packages
- Typically you just enter “**pip install <packagename>**“
- PIP uses the Python Package Index, PyPI as a source, which stores almost 200.000 Python projects

https://pypi.org

Using libraries

You need to use the **import** keyword on top of you Python script:

```
import packagename as alias

.. Your Python code
```

Example: Using numpy:

```
import numpy as np

x = 3

y = np.sin(x)

print(y)
```

NumPy

pip install numpy

- The only prerequisite for NumPy is Python itself.
- If you don’t have Python yet and want the simplest way to get started, you can use the **Anaconda Distribution** - it includes Python, NumPy, and other commonly used packages for scientific computing and data science.
- Or use “pip install numpy“

https://numpy.org

Matplotlib

```
import matplotlib.pyplot as plt
```

- Typically you need to create some plots or charts. In order to make plots or charts in Python you will need an external library. The most used library is Matplotlib
- Matplotlib is a Python 2D plotting library
- Here you find an overview of the Matplotlib library: https://matplotlib.org
- Matplotlib is included with Anaconda Distribution
- If you are familiar with MATLAB and basic plotting in MATLAB, using the Matplotlib is very similar.
- The main difference from MATLAB is that you need to import the library, either the whole library or one or more functions.

Matplotlib

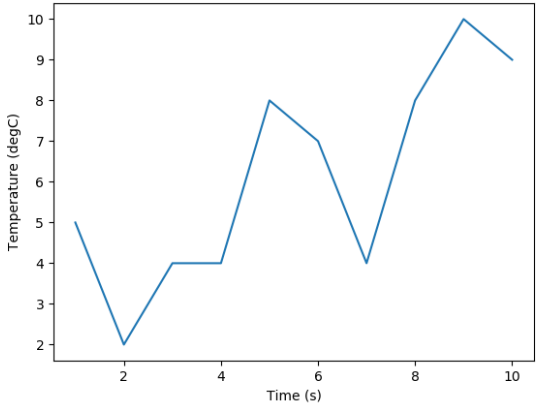
Here are some plotting functions that you will use a lot:

- plot()
- title()
- xlabel()
- ylabel()
- axis()
- grid()
- subplot()
- legend()
- show()

Matplotlib

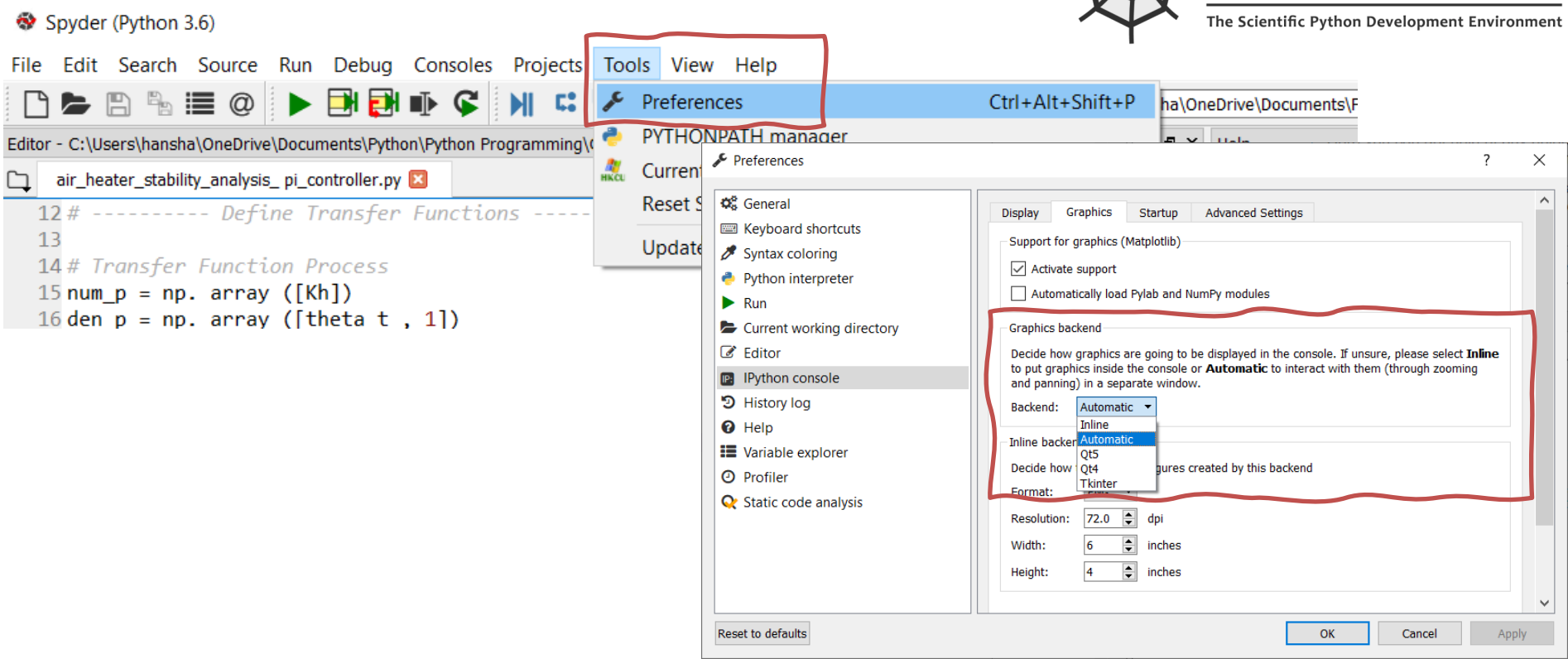
In this example we have two arrays with data. We want to plot x vs. y. We can assume x is a time series and y is the corresponding temperature in degrees Celsius.

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [5, 2,4, 4, 8, 7, 4, 8, 10, 9]

plt.plot(x,y)
plt.xlabel('Time (s)')
plt.ylabel('Temperature (degC)')
plt.show()
```



Matplotlib in Spyder

Typically you want to show figures and plots in separate windows

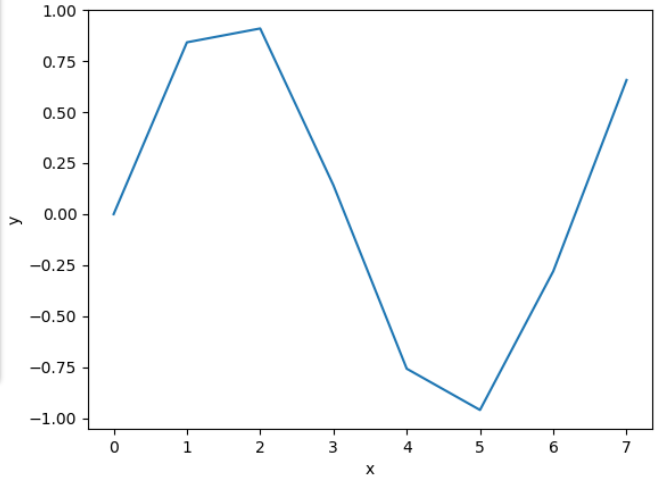


Matplotlib

Example: Plotting a Sine Curve

```
import numpy as np
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5, 6, 7]

y = np.sin(x)

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Note! The curve is not smooth due to few data points



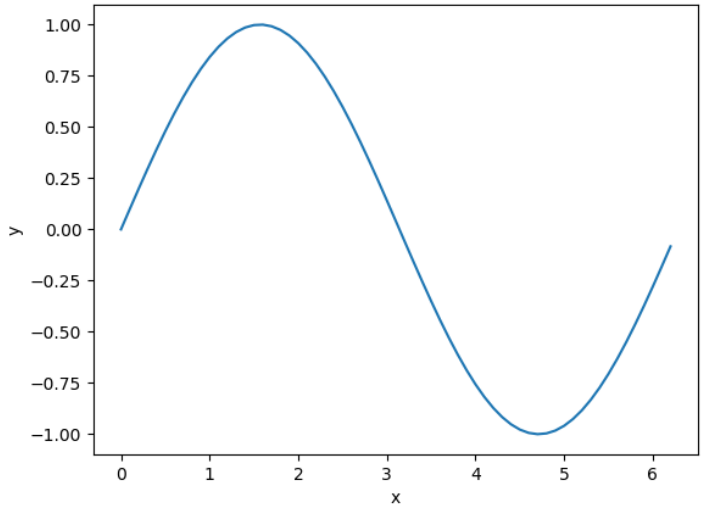
If you want grids you can use the **grid()** function

Matplotlib

Improved Solution: Plotting a Sine Curve

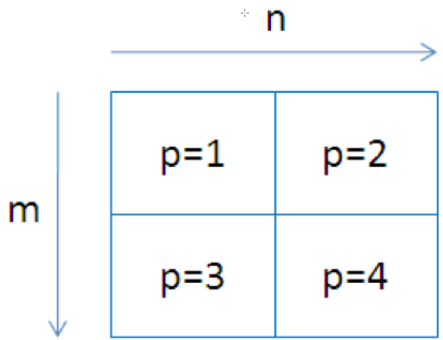
```
import matplotlib.pyplot as plt
import numpy as np

xstart = 0
xstop = 2*np.pi
increment = 0.1

x = np.arange(xstart,xstop,increment)
y = np.sin(x)

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Better!



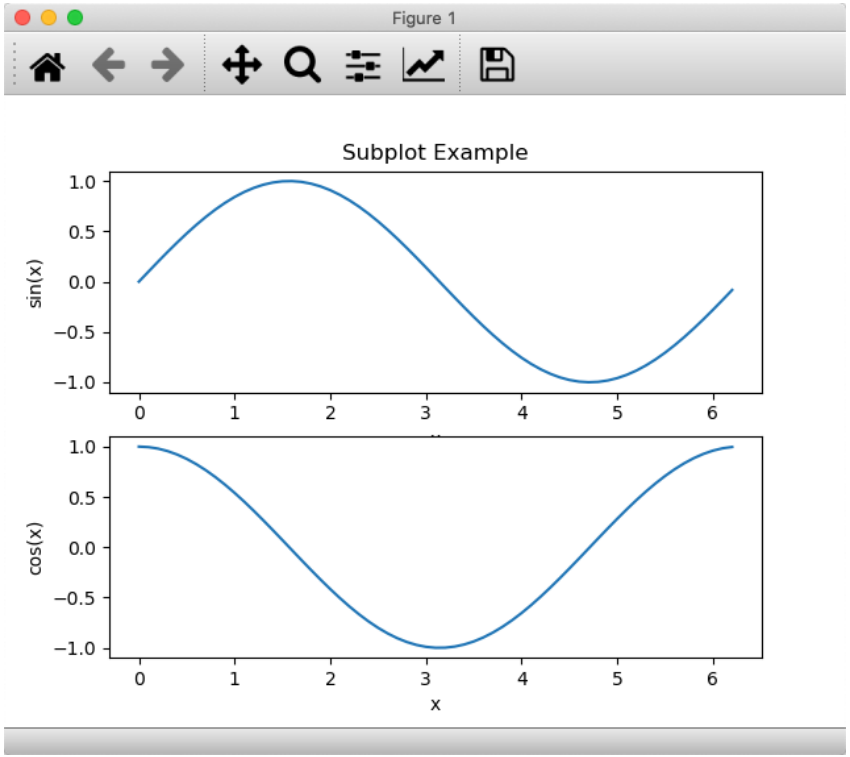
Sub-Plots

- The subplot command enables you to display multiple plots in the same window (Figure)
- Typing "subplot(m,n,p)" partitions the Figure window into an m-by-n matrix of small subplots




```
subplot(2,2,1)
```

Sub-Plots

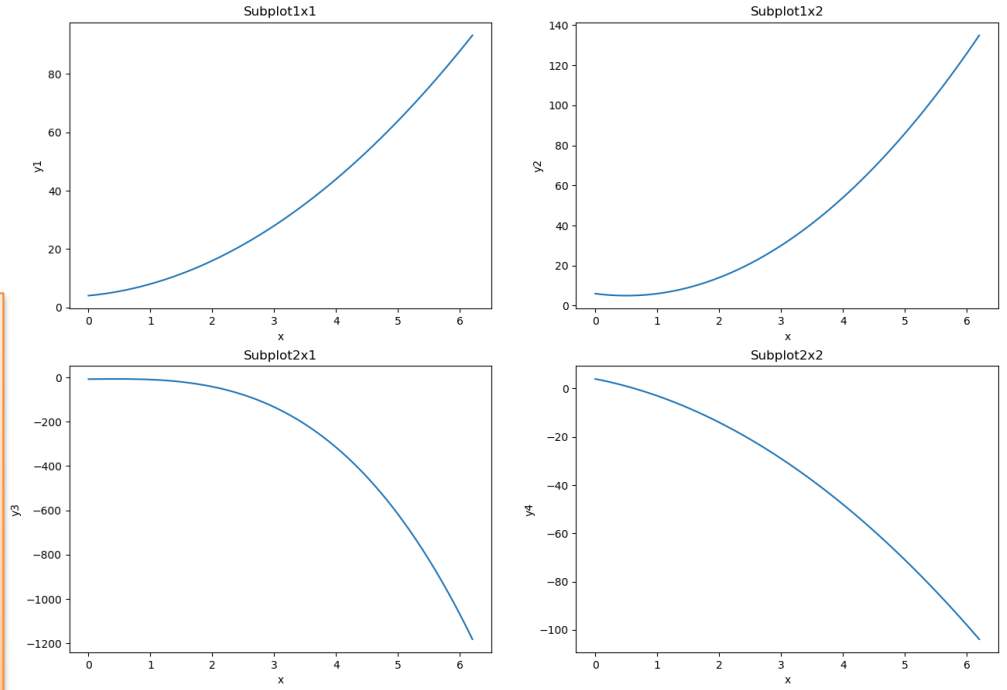



```
import matplotlib.pyplot as plt
import numpy as np

xstart = 0
xstop = 2*np.pi
increment = 0.1

x = np.arange(xstart,xstop,increment)

y = np.sin(x)

plt.subplot(2,1,1)
plt.plot(x, y)
plt.title("Subplot Example")
plt.xlabel('x')
plt.ylabel('sin(x)')

z = np.cos(x)

plt.subplot(2,1,2)
plt.plot(x, z)
plt.xlabel('x')
plt.ylabel('cos(x)')
plt.show()
```

Sub-Plots

```
import matplotlib.pyplot as plt
import numpy as np

xstart = 0
xstop = 2*np.pi
increment = 0.1

x = np.arange(xstart,xstop,increment)

y1 = 2*x**2 + 2*x + 4
plt.subplot(2,2,1)
plt.plot(x, y1)
plt.title("Subplot1x1")
plt.xlabel('x')
plt.ylabel('y1')

y2 = 4*x**2 - 4*x + 6
plt.subplot(2,2,2)
plt.plot(x, y2)
plt.title("Subplot1x2")
plt.xlabel('x')
plt.ylabel('y2')
```

```
y3 = -5*x**3 + 3*x - 8
plt.subplot(2,2,3)
plt.plot(x, y3)
plt.title("Subplot2x1")
plt.xlabel('x')
plt.ylabel('y3')

y4 = -2*x**2 - 5*x + 4
plt.subplot(2,2,4)
plt.plot(x, y4)
plt.title("Subplot2x2")
plt.xlabel('x')
plt.ylabel('y4')
plt.show()
```



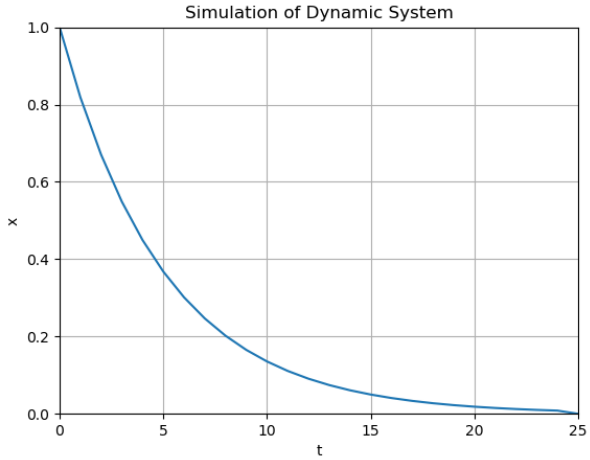
Simulation and Plotting

Given the system (differential equation): $\dot{x} = ax$

The solution is given by: $x(t) = e^{at}x_0$

Where $a = -\frac{1}{T}$ T is the time constant, $T = 5$

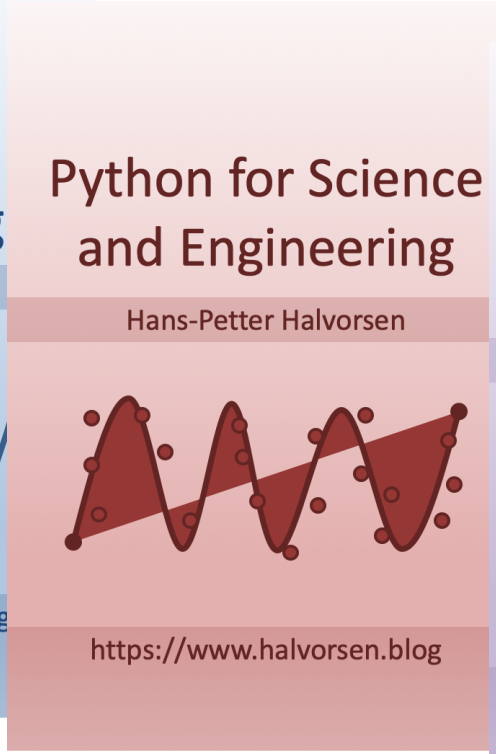
Initial condition: $x(0) = x_o = 1$ $0 \leq t \leq 25$



We should add Grid, and proper Title and Axis Labels to the plot

```
import math as mt
import numpy as np
import matplotlib.pyplot as plt

# Model Parameters
T = 5
a = -1/T

# Simulation Parameters
x0 = 1
t = 0
tstart = 0
tstop = 25
increment = 1
x = []
x = np.zeros(tstop+1)
t = np.arange(tstart,tstop+1,increment)

# Define the Function
for k in range(tstop):
    x[k] = mt.exp(a*t[k]) * x0

# Plot the Simulation Results
plt.plot(t,x)
plt.title('Simulation of Dynamic System')
plt.xlabel('t')
plt.ylabel('x')
plt.grid()
plt.axis([0, 25, 0, 1])
plt.show()
```

Additional Python Resources



https://www.halvorsen.blog/documents/programming/python/

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog