

<https://www.halvorsen.blog>



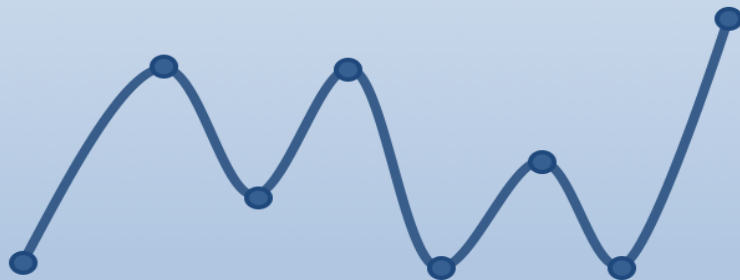
# File Handling in Python

Hans-Petter Halvorsen

# Free Textbook with lots of Practical Examples

## Python Programming

Hans-Petter Halvorsen



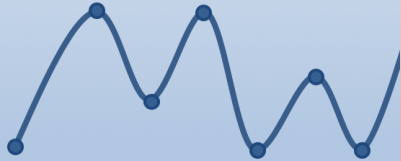
<https://www.halvorsen.blog>

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

# Additional Python Resources

## Python Programming

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Science and Engineering

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Control Engineering

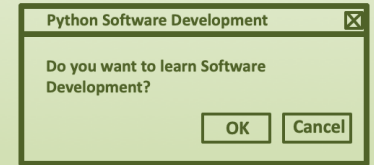
Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Software Development

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

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

# Contents

- File Handling in Python
- Write Data to File
- Read Data from File
- Logging Data to File
- Open File in Excel
- Open Data in Python and Plot the Data

# Python Editors

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



SPYDER

The Scientific Python Development Environment



ANACONDA®



# Spyder (Anaconda distribution)

Run Program button

The screenshot displays the Spyder Python IDE interface. The top toolbar contains a green play button (Run Program button) circled in red. The main window is divided into three panes: the Code Editor window on the left, the Variable Explorer window in the top right, and the IPython Console window at the bottom. The Code Editor window shows a Python script named 'temp.py' with the following code:

```
1 x = 2
2 y = 4
3 z = x + y
4 print(z)
```

The Variable Explorer window displays a table of variables:

Name	Type	Size	Value
x	int	1	2
y	int	1	4
z	int	1	6

The IPython Console window shows the execution of the script:

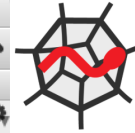
```
Python 3.7.0 (default, Jun 28 2018, 07:39:16)
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/halvorsen/.spyder-py3/temp.py', wdir= /Users/
halvorsen/.spyder-py3')
6

In [2]: |
```

At the bottom of the interface, the status bar shows: Permissions: RW, End-of-lines: LF, Encoding: UTF-8, Line: 4, Column: 9, Memory: 72 %.



SPYDER

The Scientific Python Development Environment

<https://www.anaconda.com>

Code Editor window

Variable Explorer window

Console window

# File Handling in Python

- Python has several functions for creating, reading, updating, and deleting files
- The key function for working with files in Python is the `open ( )` function.
- The `open ( )` function takes two parameters; Filename, and Mode.

# File Handling in Python

There are four different methods (modes) for opening a file:

- **"x"** - **Create** - Creates the specified file, returns an error if the file exists
- **"w"** - **Write** - Opens a file for writing, creates the file if it does not exist
- **"r"** - **Read** - Default value. Opens a file for reading, error if the file does not exist
- **"a"** - **Append** - Opens a file for appending, creates the file if it does not exist

In addition you can specify if the file should be handled as binary or text mode

- **"t"** - **Text** - Default value. Text mode
- **"b"** - **Binary** - Binary mode (e.g. images)



<https://www.halvorsen.blog>



# Write Data to File

Hans-Petter Halvorsen

# Write Data to File

To create a **New** file in Python, use the `open()` method, with one of the following parameters:

- `"x"` - **Create** - Creates the specified file, returns an error if the file exists
- `"w"` - **Write** - Opens a file for writing, creates the file if it does not exist
- `"a"` - **Append** - Opens a file for appending, creates the file if it does not exist


To write to an **Existing** file, you must add a parameter to the `open()` method:

- `"w"` - **Write** - Opens a file for writing, creates the file if it does not exist
- `"a"` - **Append** - Opens a file for appending, creates the file if it does not exist

# Write Data to File

File Name

x = Create



```
f = open("myfile.txt", "x")
```

```
data = "Hello World"
```

```
f.write(data)
```

```
f.close()
```

<https://www.halvorsen.blog>



# Read Data from File

Hans-Petter Halvorsen

# Read Data from File


To read to an existing file, you must add the following parameter to the `open()` function:

- `"r"` - **Read** - Default value. Opens a file for reading, error if the file does not exist

# Read Data from File

File Name

r = Read



```
f = open("myfile.txt", "r")
```

```
data = f.read()
```

```
print(data)
```

```
f.close()
```

<https://www.halvorsen.blog>



# Logging Data to File

Hans-Petter Halvorsen

# Logging Data to File

Typically you want to write multiple data to the, e.g., assume you read some temperature data at regular intervals and then you want to save the temperature values to a File.

Logging Data to File:

```
data = [1.6, 3.4, 5.5, 9.4]

f = open("logdata.txt", "x")

for value in data:
    record = str(value)
    f.write(record)
    f.write("\n")

f.close()
```

Read Logged Data from File:

```
f = open("logdata.txt", "r")

for record in f:
    record = record.replace("\n", "")
    print(record)

f.close()
```

Add New Line (or "Enter")





<https://www.halvorsen.blog>



# Practical Examples

Hans-Petter Halvorsen

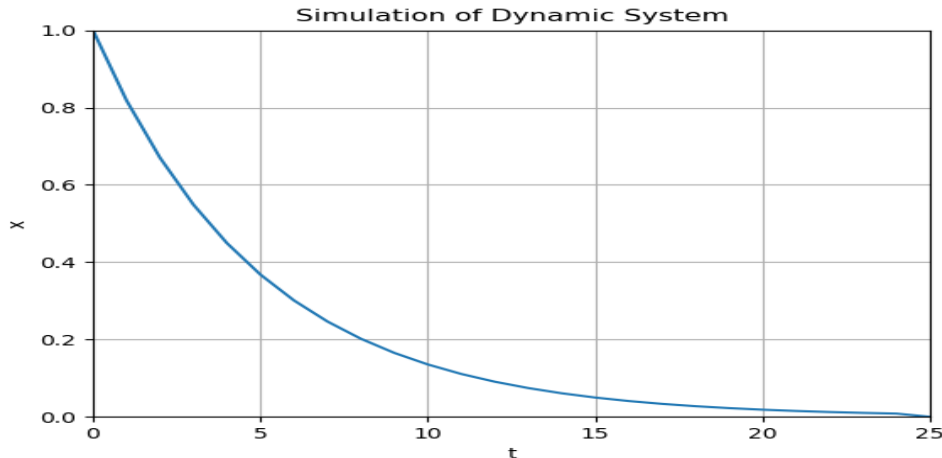
# Simulation and Plotting

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:  $x(0) = x_0 = 1$   $0 \leq t \leq 25$



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

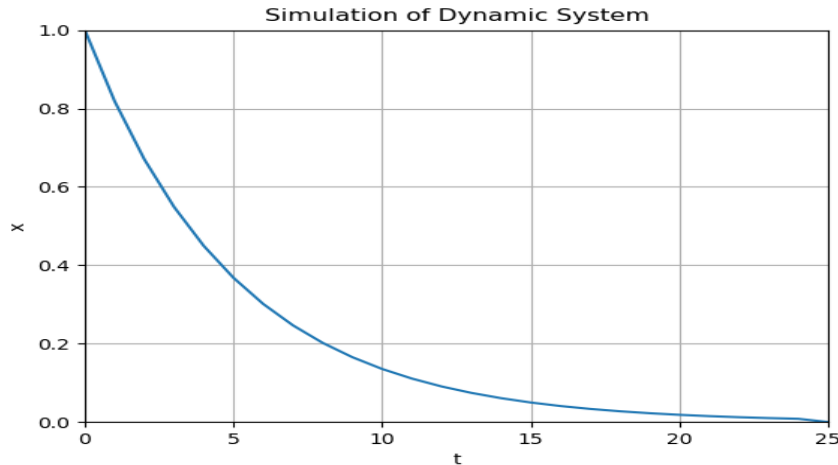
# Simulation Parameters
x0 = 1
t = 0
tstart = 0
tstop = 25
increment = 1
x = []
x = np.zeros(tstop+1)
t = np.arange(tstart,tstop+1,increment)

# 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()
```

# Logging Simulation Data

simdata.txt	
0	1.0
1	0.82
2	0.67
3	0.55
4	0.45
5	0.37
6	0.3
7	0.25
8	0.2
9	0.17
10	0.14
11	0.11
12	0.09
13	0.07
14	0.06
15	0.05
16	0.04
17	0.03
18	0.03
19	0.02
20	0.02
21	0.01
22	0.01
23	0.01
24	0.01



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

```
# Open File
f = open("simdata.txt", "w")
```

```
def writedata(t, x):
    time = str(t)
    value = str(round(x, 2))
    f.write(time + "\t" + value)
    f.write("\n")
```

```
# 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)
```

```
for k in range(tstop):
    x[k] = mt.exp(a*t[k]) * x0
    writedata(t[k], x[k])
```

```
f.close()
```

```
# 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()
```

# Code Details

A separate Function for dealing with the Writing to the File has been made. That is of course not necessary, but the code becomes easier to read and maintain.

I guess you should always think about the code structure, because when your program grows it may become "messy". Also consider reuse where possible, i.e., create Functions in separate files, etc.

```
def writedata(t, x):  
    time = str(t)  
    value = str(round(x, 2))  
    f.write(time + "\t" + value)  
    f.write("\n")
```

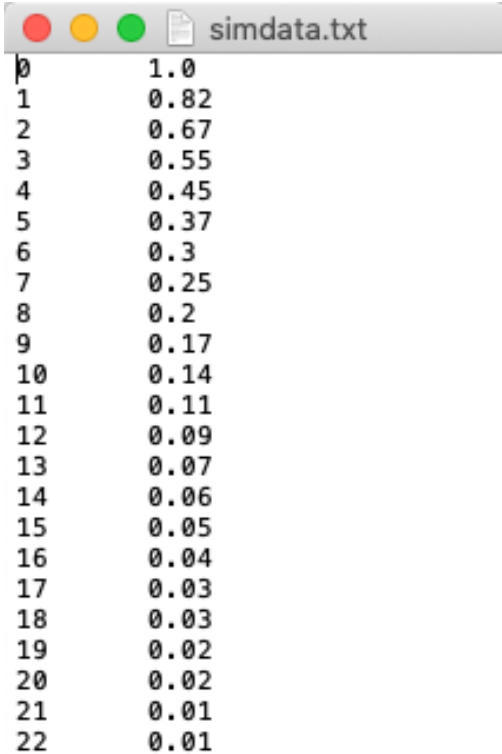
Convert to String values

Show values with 2 Decimals

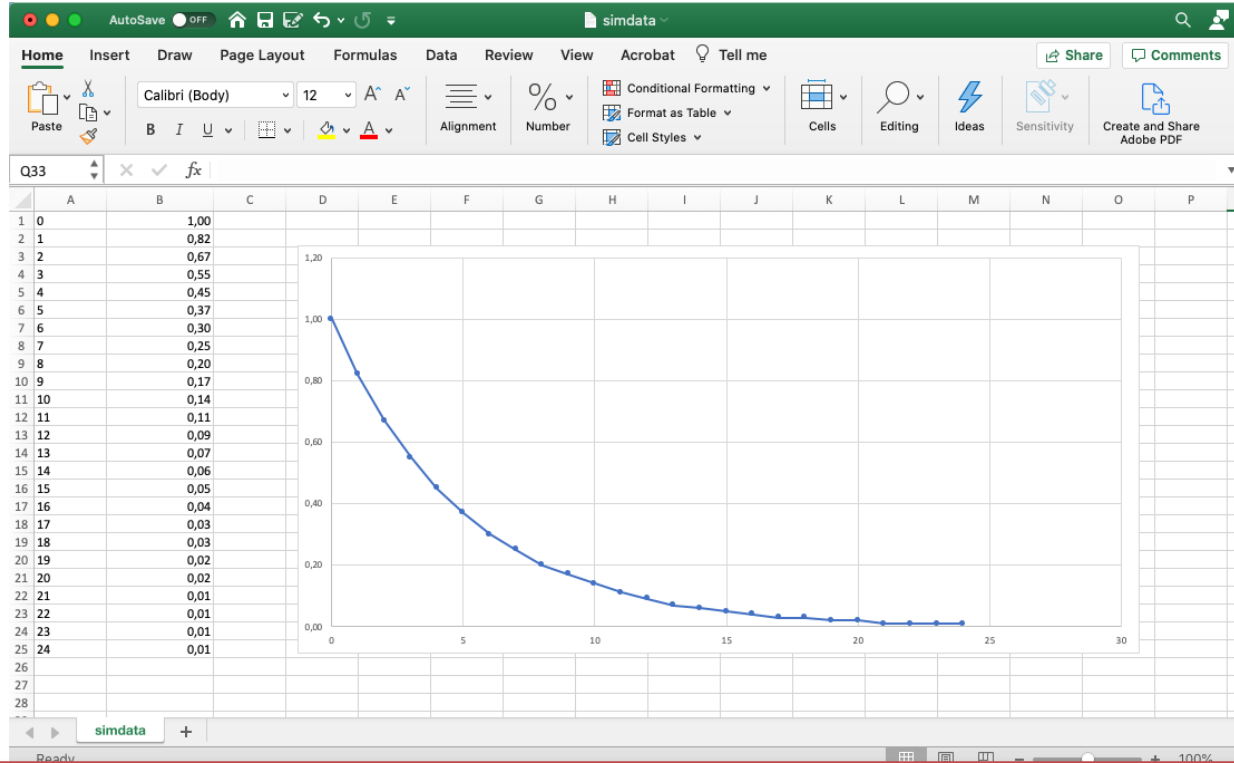
Add New Line (or "Enter")

Add **Tabulator**, so the t and x will be in 2 separate columns in the file

# Open Data in Excel



0	1.0
1	0.82
2	0.67
3	0.55
4	0.45
5	0.37
6	0.3
7	0.25
8	0.2
9	0.17
10	0.14
11	0.11
12	0.09
13	0.07
14	0.06
15	0.05
16	0.04
17	0.03
18	0.03
19	0.02
20	0.02
21	0.01
22	0.01



Depending on your Settings in Excel, you may need to convert the Decimal Point from “.” to “,”

# Open Data in Excel

simdata.txt

```
0 1.0
1 0.82
2 0.67
3 0.55
4 0.45
5 0.37
6 0.3
7 0.25
8 0.2
9 0.17
10 0.14
11 0.11
12 0.09
13 0.07
14 0.06
15 0.05
16 0.04
17 0.03
18 0.03
19 0.02
20 0.02
21 0.01
22 0.01
23 0.01
24 0.01
```

Text Import Wizard - Step 1 of 3

The Text Wizard has determined that your data is **Delimited**.  
If this is correct, choose Next, or choose the data type that best describes your data.

Delimited – Characters such as commas or tabs separate each field.  
 Fixed width – Fields are aligned in columns with spaces between each field.

Start import at row:  File origin:

Preview of selected data:

Preview of file /Users/halvorsen/Downloads/python/simdata.txt.

```
1 1.0
2 0.82
3 0.67
4 0.55
5 0.45
6 0.37
7 0.3
8 0.25
9 0.2
```

Cancel < Back Next > Finish

Text Import Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains.

Delimiters

Tab  Treat consecutive delimiters as one  
 Semicolon  
 Comma  
 Space  
 Other:

Text qualifier:

Preview of selected data:

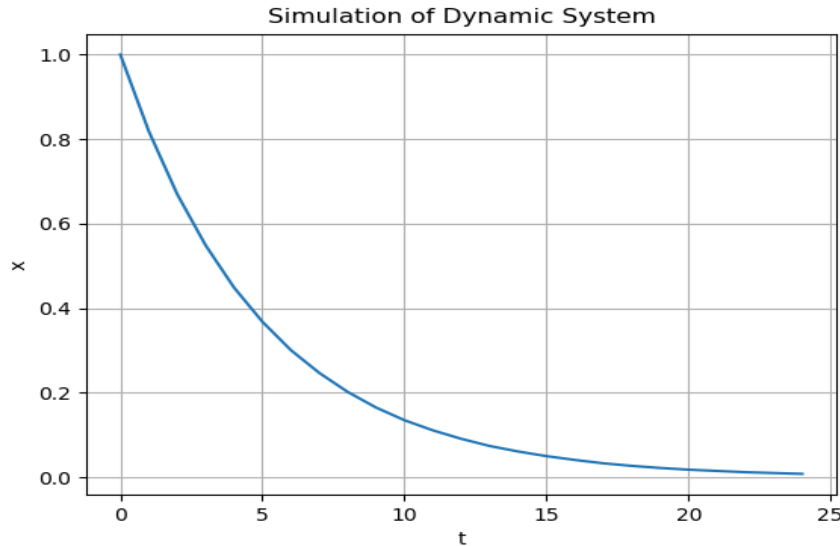
```
0 1.0
1 0.82
2 0.67
3 0.55
4 0.45
5 0.37
6 0.3
7 0.25
8 0.2
```

Cancel < Back Next > Finish

# Read Simulation Data

simdata.txt

```
0 1.0
1 0.82
2 0.67
3 0.55
4 0.45
5 0.37
6 0.3
7 0.25
8 0.2
9 0.17
10 0.14
11 0.11
12 0.09
13 0.07
14 0.06
15 0.05
16 0.04
17 0.03
18 0.03
19 0.02
20 0.02
21 0.01
22 0.01
23 0.01
24 0.01
```



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

```
f = open("simdata.txt", "r")
```

```
t = []
```

```
x = []
```

```
for record in f:
```

```
    record = record.replace("\n", "")
```

```
    time, value = record.split("\t")
```

```
    t.append(int(time))
```

```
    x.append(float(value))
```

```
f.close()
```

```
# Plot the File Data
```

```
plt.plot(t,x)
```

```
plt.title('Simulation of Dynamic  
System')
```

```
plt.xlabel('t')
```

```
plt.ylabel('x')
```

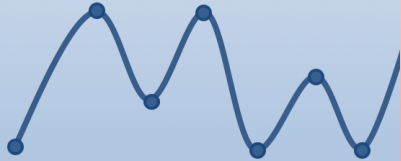
```
plt.grid()
```

```
plt.show()
```

# Additional Python Resources

## Python Programming

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Science and Engineering

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Control Engineering

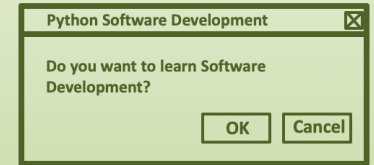
Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Software Development

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

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



# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

Web: <https://www.halvorsen.blog>

