https://www.halvorsen.blog

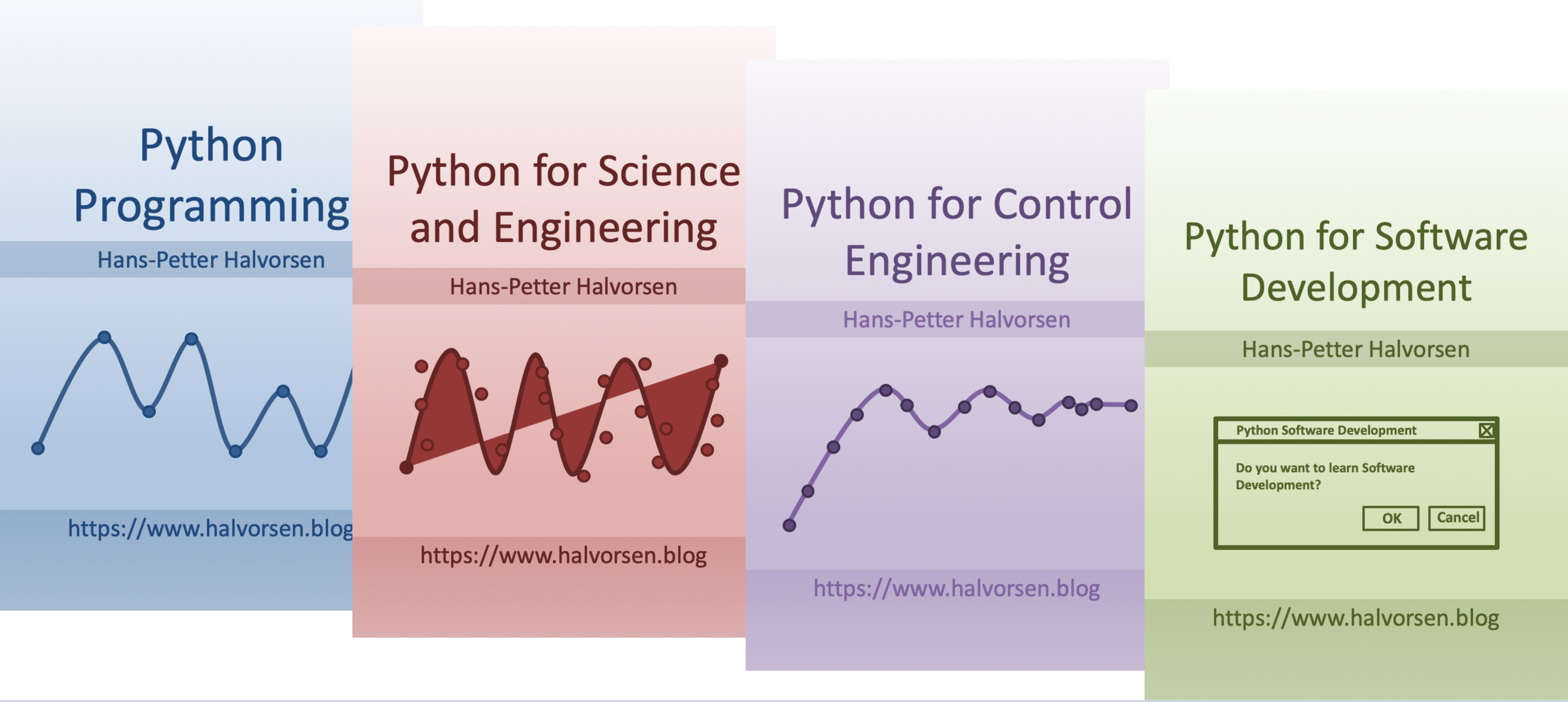
Mathematics in Python

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/

Mathematics in Python

- Python is a powerful tool for mathematical calculations
- Python Standard Library
  - math Module
  - statistics Module
- NumPy Library

Contents

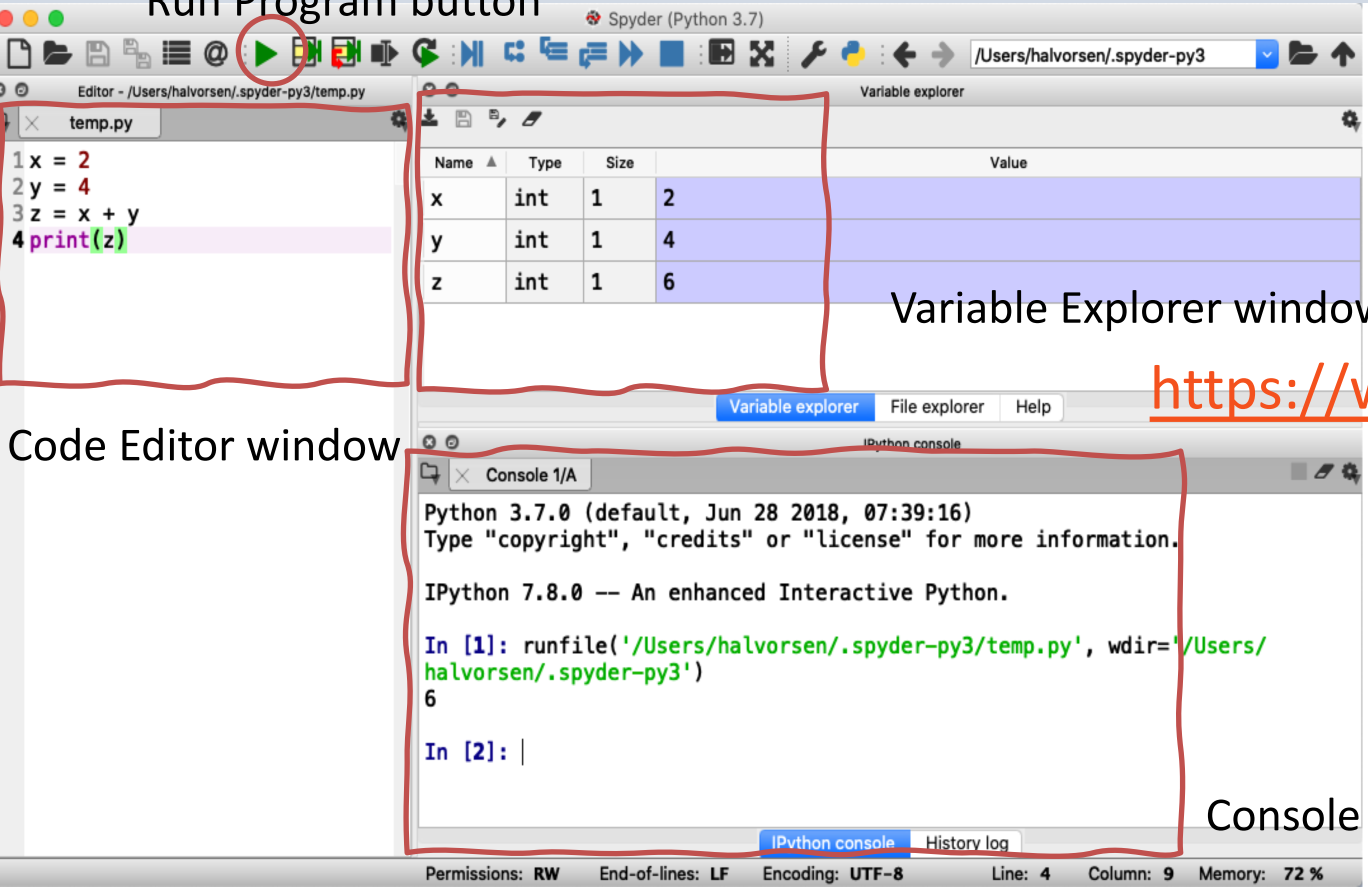
- Python Standard Library and Basic Math Functions
- NumPy Library
- Statistics
- Trigonometric Functions
- Polynomials
- Complex Numbers

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

https://www.anaconda.com

Code Editor window

Console window

Calculations in Python

We can use variables in a calculation like this:

$$y(x) = 2x + 4$$

$y(3) = ?$

$y(5) = ?$

```
> a = 2
> b = 4

> x = 3
> y = a*x + b
> print(y)

> x = 5
> y = a*x + b
> print(y)
```

$$y(x) = ax + b$$

Python Standard Library

- Python allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your programs.
- The Python Standard Library consists of different modules for handling file I/O, basic mathematics, etc.
- You don't need to install the modules in the Python Standard Library separately, but you need to important them when you want to use some of these modules or some of the functions within these modules.

math Module

Python Standard Library

The math module has all the basic math functions you need, such as:

- Trigonometric functions: sin(x), cos(x), etc.
- Logarithmic functions: log(), log10(), etc.
- Statistics: mean(), stdev(), etc.
- Constants like pi, e, inf, nan, etc.

math Module

If we need only the sin() function, we can do like this:

```
from math import sin

x = 3.14
y = sin(x)
```

If we need many functions, we can do like this:

```
from math import *

x = pi
y = sin(x)
print(y)

y = cos(x)
print(y)

…
```

If we need a few functions, we can do like this:

```
from math import sin, cos

x = 3.14
y = sin(x)
print(y)

y = cos(x)
print(y)
```

We can also do like this:

```
import math
x = 3.14
y = math.sin(x)
print(y)
```

Basic Math Functions

Some basic math functions in Python Standard Library:

- `math.exp(x)`
- `math.log(x)`
- `math.log10(x)`
- `math.pow(x,y)`
- `math. sqrt(x)`
- …

https://docs.python.org/3/library/math.html

Some basic Examples:

```
import math as mt

x = 3

y = mt.exp(x)
print(y)

y = mt.log(x)
print(y)

y = mt.log10(x)
print(y)

n = 2
y = mt.pow(x,n)
print(y)
```

Mathematical Expressions

Let's create the following mathematical expression in Python:

$$f(x,y) = 3x^2 + \sqrt{x^2 + y^2} + e^{ln(x)}$$

$f(2,2) = ?$

Python Code:

```
import math as mt

x = 2
y = 2

f = 3*mt.pow(x,2) + mt.sqrt(mt.pow(x,2) + mt.pow(y,2)) + mt.exp(mt.log(x))

print(f)
```

The answer becomes $f(2,2) = 16.83$

Mathematical Expressions

Let's create a **function** that calculates the following mathematical expression:

$$f(x,y) = 3x^2 + \sqrt{x^2 + y^2} + e^{ln(x)}$$

Python Code:

```
import math as mt

def func_ex(x,y):
    f = 3*mt.pow(x,2) + mt.sqrt(mt.pow(x,2) + mt.pow(y,2)) + mt.exp(mt.log(x))
    return f

x = 2
y = 2

f = func_ex(x,y)

print(f)
```

NumPy

- The Python Standard Library consists basic Math functions, for fore advanced Math functions, you typically want to use the NumPy Library
- 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

NumPy

Basic NumPy Example:

```
import numpy as np

x = 3

y = np.sin(x)

print(y)
```

In this example we use both the math module in the Python Standard Library and the NumPy library:

```
import math as mt
import numpy as np

x = 3

y = mt.sin(x)
print(y)

y = np.sin(x)
print(y)
```

As you see, NumPy also have also similar functions (e.g., sim(), cos(), etc.) as those who is part of the math library, but they are more powerful

Mathematical Expressions

Let's create the following mathematical expression in Python using **NumPy**:

$$f(x,y) = 3x^2 + \sqrt{x^2 + y^2} + e^{ln(x)}$$

$f(2,2) = ?$

Python Code:

Previously we used math in the Python Standard Library

```
import numpy as np

def func_ex(x,y):
    f = 3*np.power(x,2) + np.sqrt(np.power(x,2) + np.power(y,2)) + np.exp(np.log(x))
    return f

x = 2
y = 2

f = func_ex(x,y)

print(f)
```

The answer becomes $f(2,2) = 16.83$

Mathematical Expressions

$$f(x,y) = 3x^2 + \sqrt{x^2 + y^2} + e^{ln(x)}$$

Let's find the values of $f(x,y)$ for $0 \leq x \leq 10$ and $0 \leq y \leq 10$

In order to do that we can use a **Nested For loop**:

```
import numpy as np

def func_ex(x,y):
    f = 3*np.power(x,2) + np.sqrt(np.power(x,2) +
    np.power(y,2)) + np.exp(np.log(x))
    return f

start = 0
stop = 11
increment = 1

x_data = np.arange(start,stop,increment)
y_data = np.arange(start,stop,increment)

for x in x_data:
    for y in y_data:
        f = func_ex(x,y)
        print(f"f({x},{y})={f}")
```

Statistics

- Mean / Average
- Variance
- Standard Deviation
- Median

The mean is the sum of the data divided by the number of data points. It is commonly called "the average”:

$$\mu = \bar{x} = \frac{x_1 + x_2 + \cdots + x_N}{N} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Variance is a measure of the variation in a data set:

$$var(x) = \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

The standard deviation is a measure of the spread of the values in a dataset or the value of a random variable. It is defined as the square root of the variance:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

$$\sigma^2 = var(x) \Leftrightarrow \sigma = \sqrt{var(x)}$$

Median

Given the following dataset:

```
data = [-1.0, 11, 2.5, 3.25, 5.75]
```

Put them in ascending order:

```
data = [-1.0, 2.5, 3.25, 5.75, 11]
```

The Median is the value in the middle

If even numbers in the dataset:

```
data = [-1.0, 11, 2.5, 3.25]
```

Put them in ascending order:

```
data = [-1.0, 2.5, 3.25, 11]
```

The Median will be:

$$(2.5 + 3.25)/2 = 2.875$$

Statistics

Example:

Statistics using the **statistics** module in **Python Standard Library**:

```
import statistics as st

data = [-1.0, 11, 2.5, 3.25, 5.75]

#Mean or Average
m = st.mean(data)
print(m)

# Standard Deviation
st_dev = st.stdev(data)
print(st_dev)

# Median
med = st.median(data)
print(med)

# Variance
var = st.variance(data)
print(var)
```

IMPORTANT: Do not name your file "statistics.py" since the import will be confused and throw the errors of the library not existing and the mean function not existing.

Trigonometric Functions

- Python offers lots of Trigonometric functions, e.g., sin, cos, tan, etc.
- Note! Most of the trigonometric functions require that the angle is expressed in radians.
- We can use **Math** module in the **Python Standard Library**
- Or we can use the **NumPy** library

Trigonometric Functions

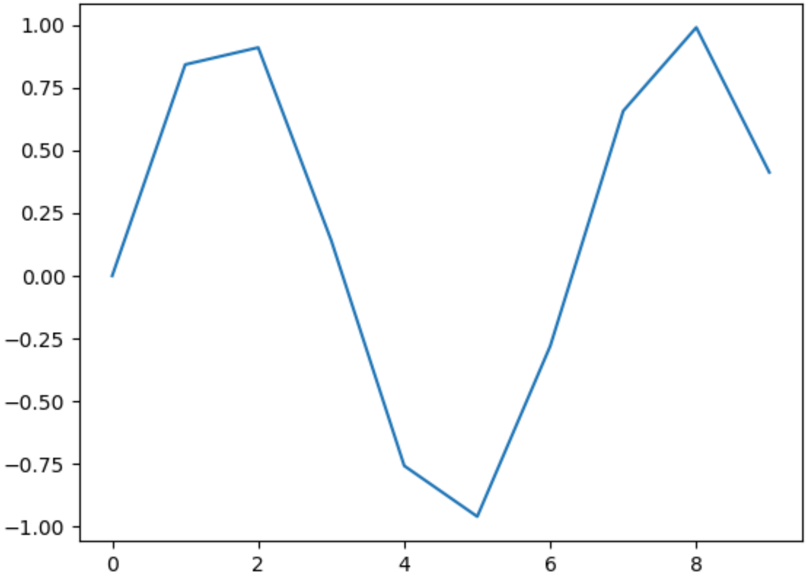
Trigonometric functions in the **Math module** in the **Python Standard Library**:

```
import math as mt

x = 2*mt.pi

y = mt.sin(x)
print(y)

y = mt.cos(x)
print(y)

y = mt.tan(x)
print(y)
```

Trigonometric Functions

Plotting Example using a For Loop and the **matplotlib** library:

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

xdata = []
ydata = []

for x in range(0, 10):
    xdata.append(x)
    y = mt.sin(x)
    ydata.append(y)

plt.plot(xdata, ydata)
plt.show()
```

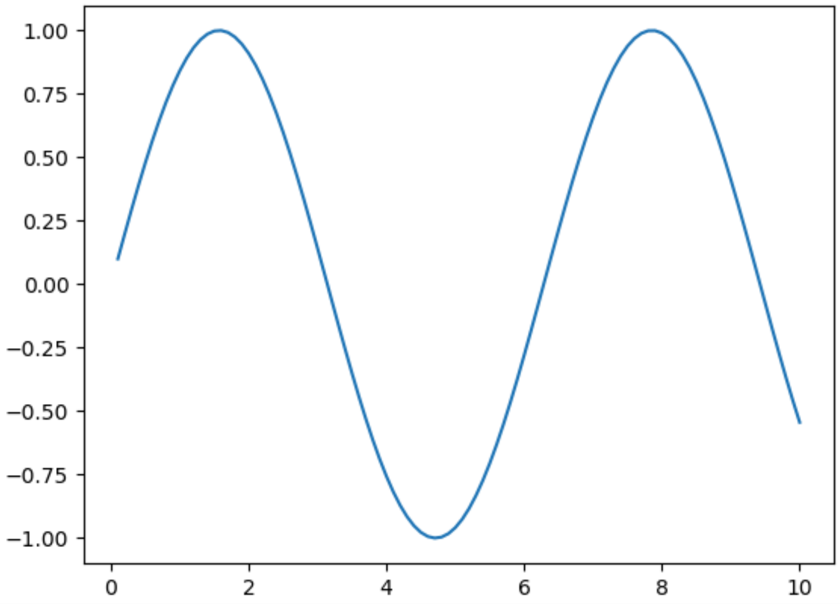


Trigonometric Functions

Improved Plotting Example:

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

x = 0
N = 100
xdata = []
ydata = []

for i in range(0, N):
    x = x + 0.1
    xdata.append(x)
    y = mt.sin(x)
    ydata.append(y)

plt.plot(xdata, ydata)
plt.show()
```

“Smoother“ curve:



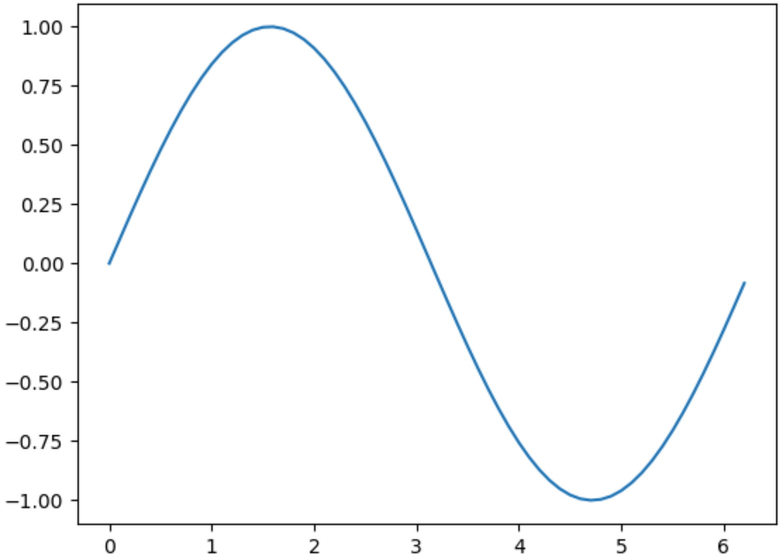
The problem with using the Trigonometric functions in the the Math module from the Python Standard Library is that they don't handle an array as input.

Trigonometric Functions

Using **NumPy**:

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

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

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

y = np.sin(x)

plt.plot(x, y)
plt.show()
```

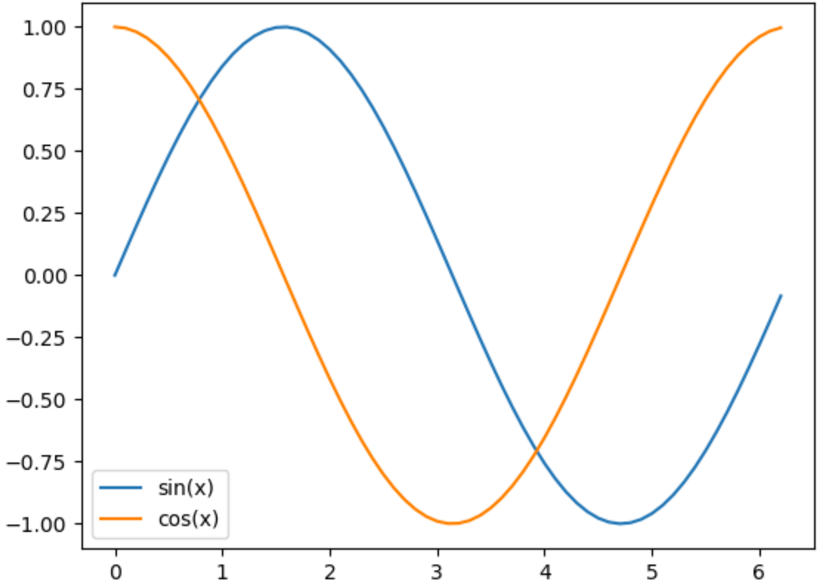
The Trigonometric Functions in the NumPy library can handle **arrays** as input arguments. <u>No For Loop needed!</u>



Trigonometric Functions

You can also plot multiple plots like this:

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

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

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

y1 = np.sin(x)
y2 = np.cos(x)

plt.plot(x, y1, x, y2)
plt.legend(["sin(x)", "cos(x)"])
plt.show()
```

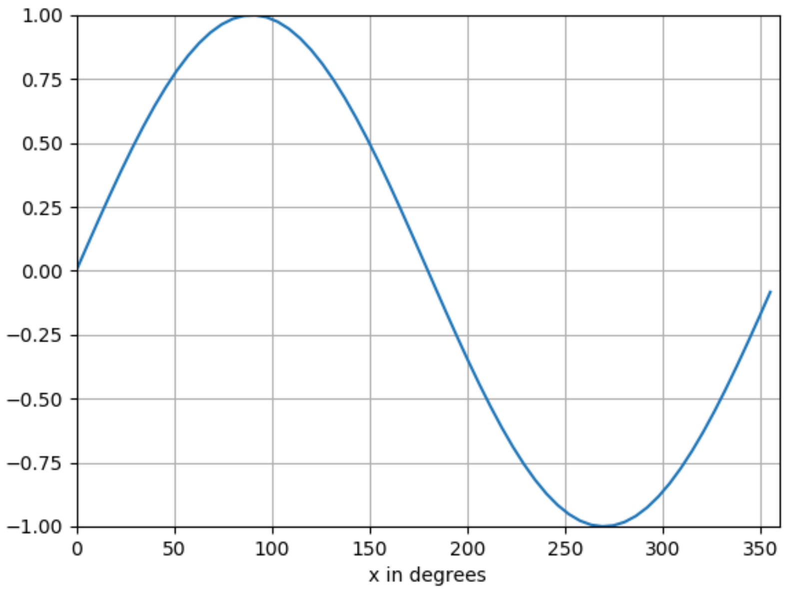


Trigonometric Functions

Converting to degrees (x-axis):

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

def r2d(r):
    d = r * (180/np.pi)
    return d

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

x = np.arange(xstart,xstop,increment)
x_deg = r2d(x)

y = np.sin(x)

plt.plot(x_deg, y)
plt.xlabel("x in degrees")
plt.axis([0, 360, -1, 1])
plt.grid()
```

Here I have created my own Function r2d(r)
You could have used **math.degrees(*x*)**



Polynomials

A polynomial is expressed as:

$$p(x) = p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + \ p_{n+1}$$

where $p_1, p_2, p_3, \ldots$ are the coefficients of the polynomial.

We will use the Polynomial Module in the NumPy Package.

https://numpy.org/doc/stable/reference/routines.polynomials.polynomial.html

Polynomials

Given the following polynomial:

$$p(x) = -5.45x^4 + 3.2x^2 + 8x + 5.6$$

We need to rewrite it like this in Python:

$$p(x) = 5.6 + 8x + 3.2x^2 + 0x^3 - 5.45x^4$$

```
import numpy.polynomial.polynomial as poly

p = [5.6, 8, 3.2, 0, -5.45]

r = poly.polyroots(p)
print(r)
```

$p(x) = 0 \rightarrow x = ?$

Polynomials

Given the following polynomial:

$$p(x) = -2.1x^4 + 2x^3 + 5x + 11$$

We need to rewrite it like this in Python:

$$p(x) = 11 + 5x + 0x^2 + 2x^3 - 2.1x^4$$

```
import numpy.polynomial.polynomial as poly

p = [11, 5, 0, 2, -2.1]
r = poly.polyroots(p)
print(r)

x = 2
px = poly.polyval(x, p)
print(px)
```

$p(x) = 0 \rightarrow x = ?$

$p(2) = ?$

Polynomials

```
import numpy.polynomial.polynomial as poly

p1 = [1, 1, -1]
p2 = [2, 0, 0, 1]

p = poly.polymul(p1, p2)

print(p)

r = poly.polyroots(p)
print(r)

x = 2
px = poly.polyval(x, p)
print(px)
```

Given the following polynomials:

$$p_1(x) = 1 + x - x^2$$

$$p_2(x) = 2 + x^3$$

Let's find the polynomial $p(x) = p_1(x) \cdot p_2(x)$ using Python

And let's find the roots of the polynomial

$$p(x) = 0$$

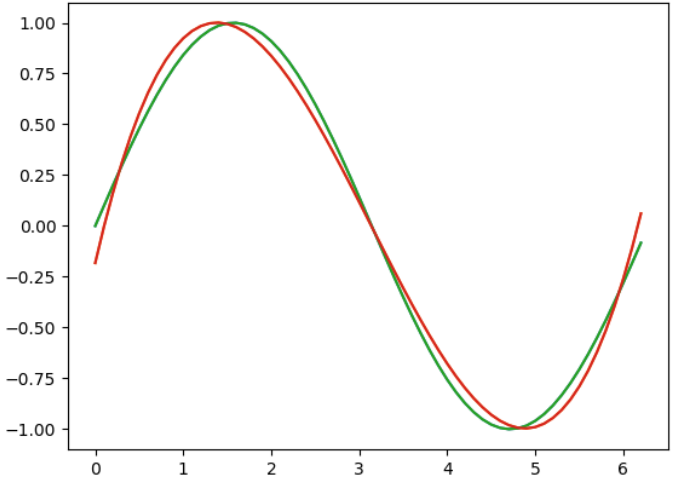
Polynomial Fitting

Find a Polynomial that best fits the following function:

$$y = \sin(x)$$

Try with different order of the polynom

```
N = 3
```



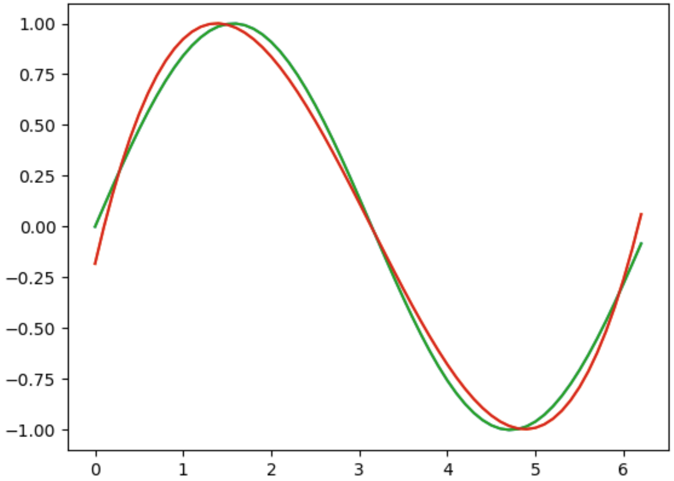

```
import numpy as np
import numpy.polynomial.polynomial as poly
import matplotlib.pyplot as plt

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

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

plt.plot(x, y)

N = 3
p = poly.polyfit(x,y,N)
print(p)

y2 = poly.polyval(x, p)

plt.plot(x, y2)
plt.show()
```

Polynomial Fitting

Find a Polynomial that best fits the following function:

$$y = \sin(x)$$

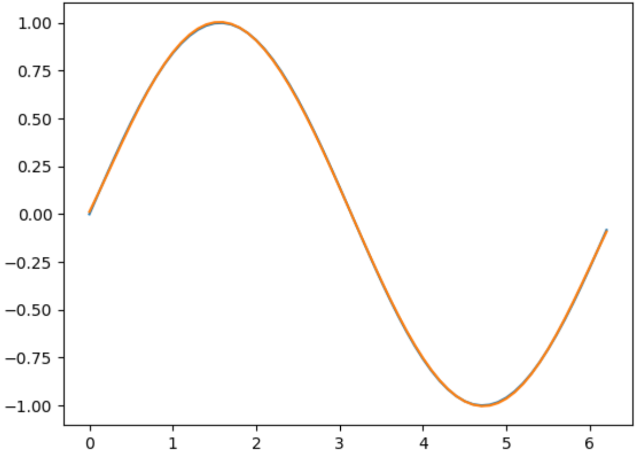
N = 3



[-0.18215486 1.88791463 -0.87536931 0.09309684]

$$p(x) = -0.18 + 1.88x - 0.88x^2 + 0.09x^3$$

N = 5



[0.01223516 0.87014661 0.27985151 -0.39981223 0.08841641 -0.0056342]

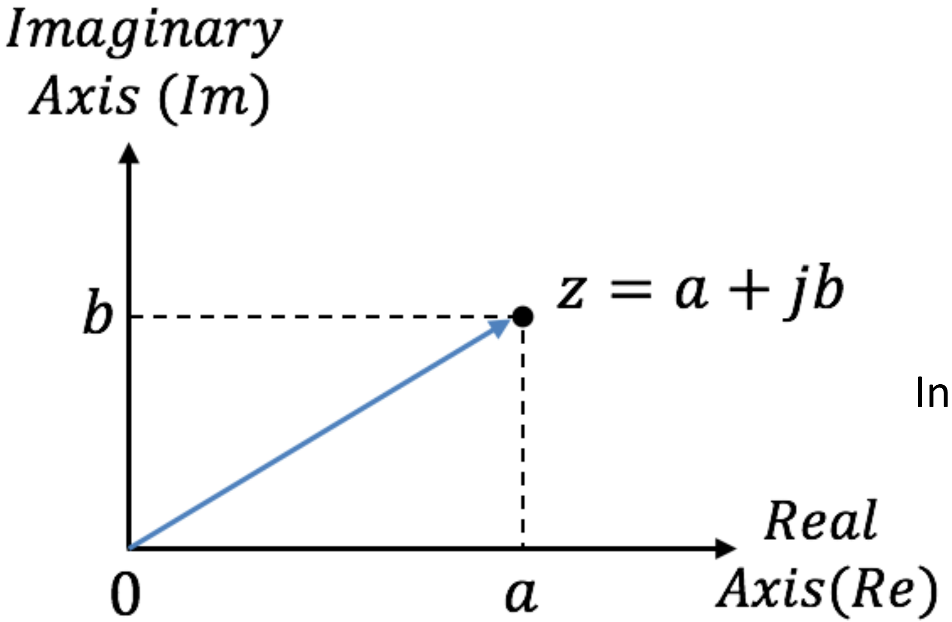
$$p(x) = 0.01 + 0.87x + 0.28x^2 - 0.4x^3 + 0.09x^4 - 0.006x^5$$

Complex Numbers

A complex number is defined like this: $z = a + jb$

Where a is called the real part of z and b is called the imaginary part of z, i.e.:

$$Re(z) = a, Im(z) = b$$



The imaginary j is defined as:

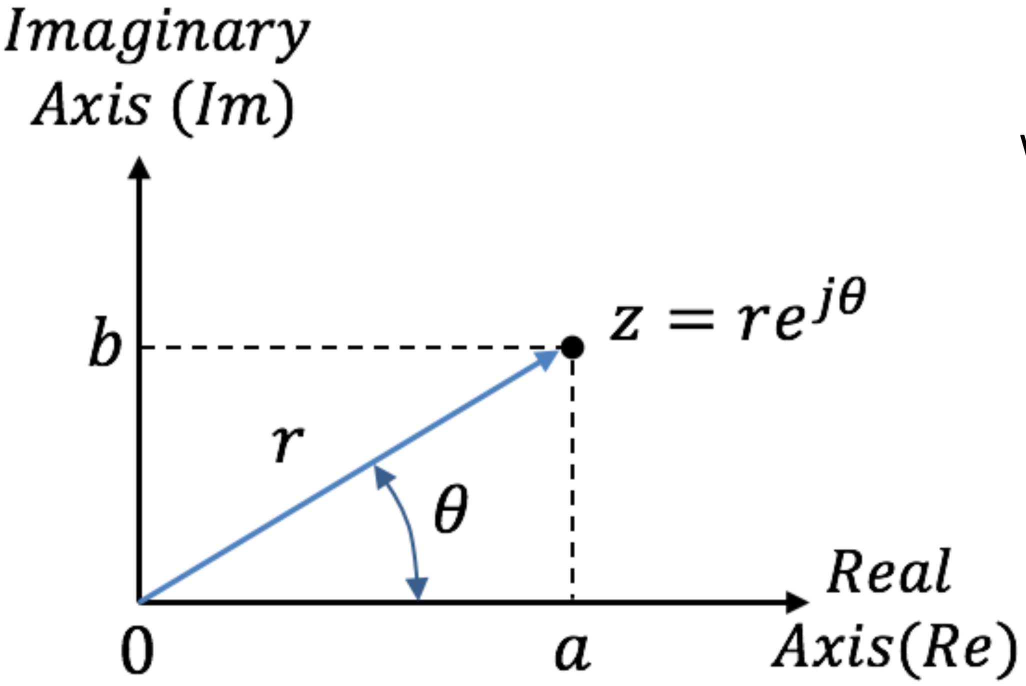
$$j = \sqrt{-1}$$

In Python you define a complex number like this:

```
z = 3 + 2j
```

Complex Numbers – Polar Form

Complex numbers can also be expressed on exponential/polar form:



$$z = re^{j\theta}$$

Where: $r = |z| = \sqrt{a^2 + b^2}$

$$\theta = atan\left(\frac{b}{a}\right)$$

Note that $a = r\cos\theta$ and $b = r\sin\theta$

Complex Numbers

Given the following complex numbers:

$$a = 5 + 3j$$
$$b = 1 - 1j$$

In Python we can define the complex numbers and perform basic operations (+, -, *, /) like this:

```
a = 5 + 3j
b = 1 - 1j

c = a + b
print(c)

d = a - b
print(d)

e = a * b
print(e)

f = a / b
print(f)
```

Complex Numbers

In addition, there exists several Complex Number Functions in Python. We use the **cmath** library:

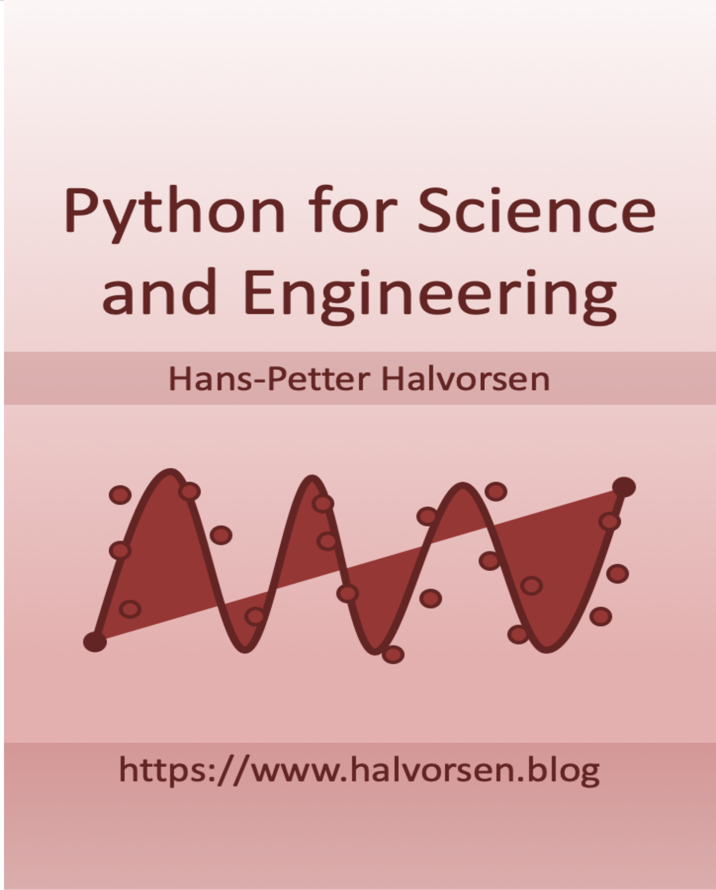
```
cmath.phase(x)
cmath.polar(x)
cmath.rect(r, phi)
cmath.exp(x)
cmath.log10(x)
cmath.sqrt(x)
cmath.acos(x)
cmath.asin(x)
cmath.atan(x)
cmath.cos(x)
cmath.sin(x)
cmath.tan(x)
```

https://docs.python.org/3/library/cmath.html

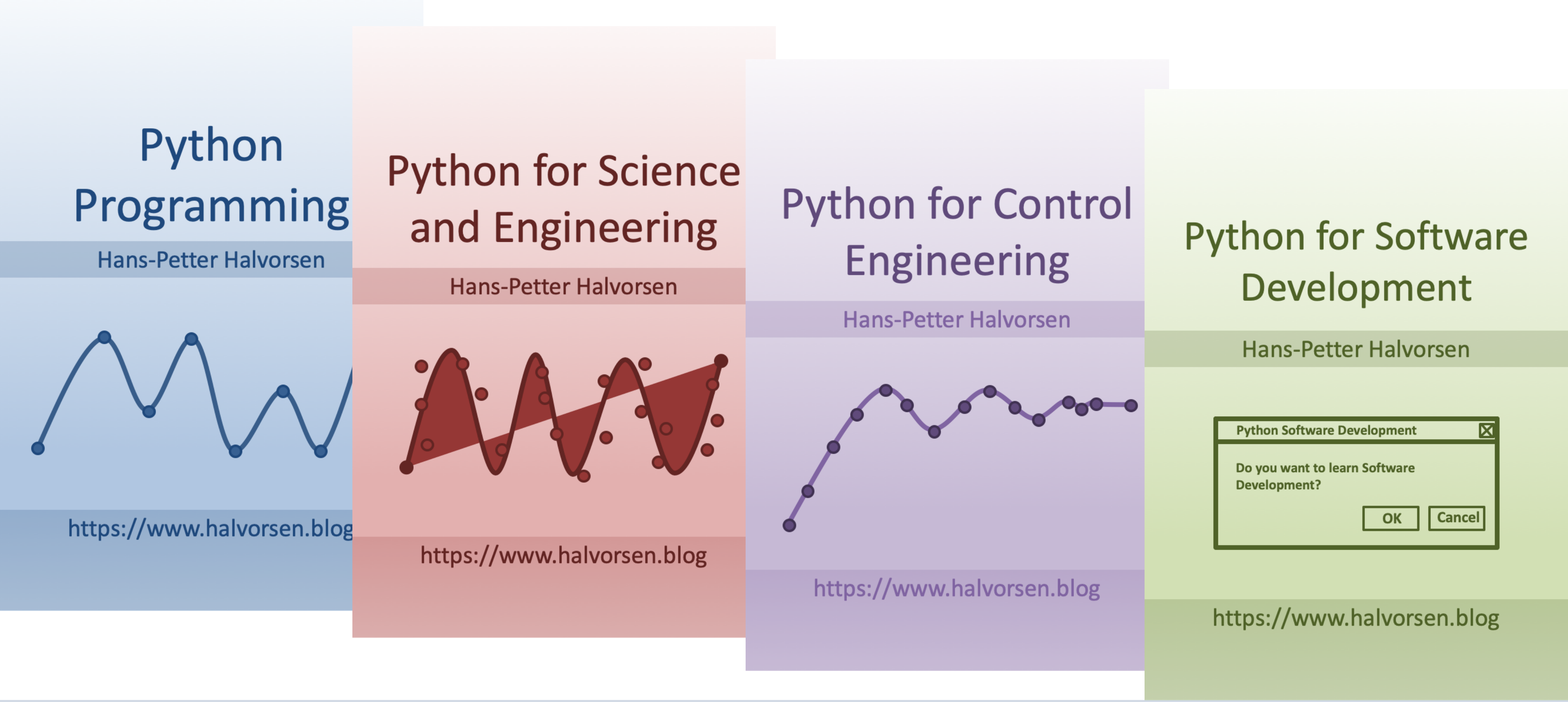
```
import cmath

x = 2
y = -3

# converting x and y into complex number
z = complex(x,y)
print(z.real)
print(z.imag)

print(z.conjugate())

# converting to polar form
w = cmath.polar(z)
print (w)

# converting to to rectangular form
w = cmath.rect(2,3)
print (w)
```

Advanced Mathematics

- Linear Algebra
- Complex Numbers
- Differential Equations
- Interpolation
- Curve Fitting
- Least Square Method
- Numerical Differentiation
- Numerical Integration
- Optimization



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