https://www.halvorsen.blog

Python Programming

Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

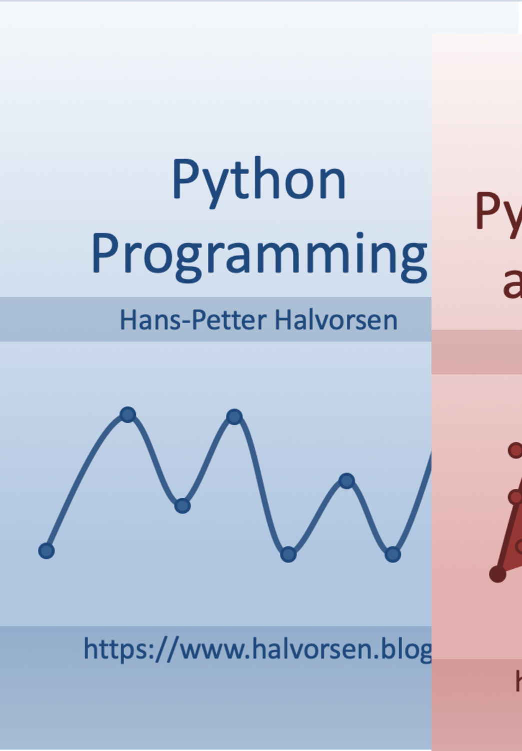
Python Programming

Hans-Petter Halvorsen

https://www.halvorsen.blog

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

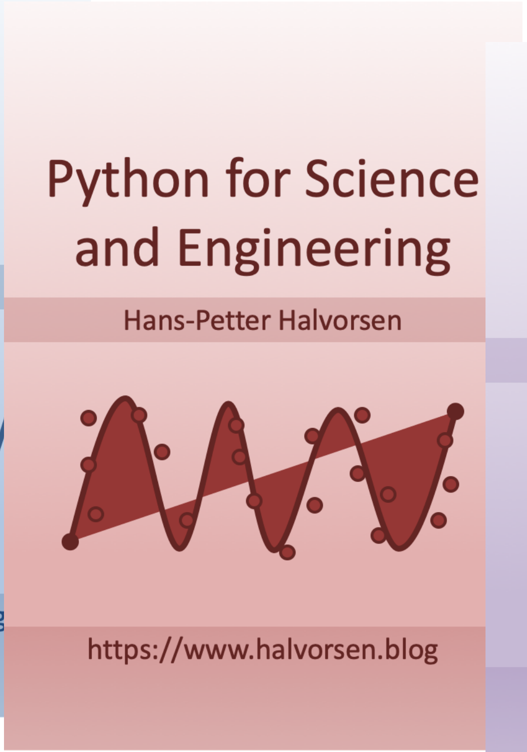
Additional Python Resources



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

Python Software Development

Do you want to learn Software Development?

OK Cancel

https://www.halvorsen.blog

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

Contents

- Variables
- If-Else-Elif (Conditions)
- Arrays
- For Loops
- While Loops
- Create Functions

- If you are familiar with one or more other programming language, these features should be familiar and known to you.
- All programming languages have these features built-in
- But the syntax is slightly different from one language to another

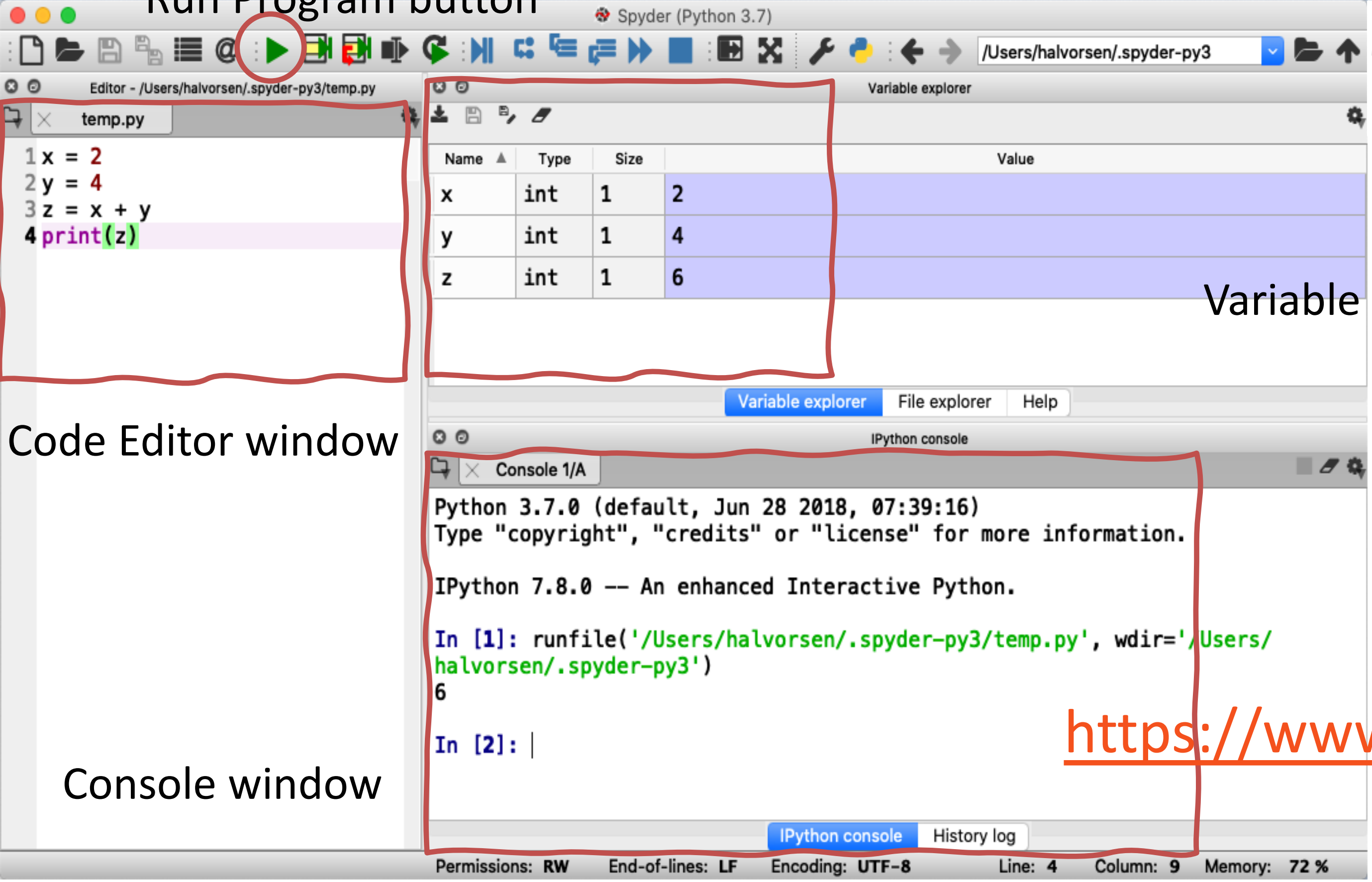
Python Editors

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

SPYDER
The Scientific Python Development Environment

ANACONDA®

Spyder (Anaconda distribution)



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

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

```
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]: |
```

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

Basic Python Program

- We use the basic IDLE editor or another Python Editor like Spyder (included with Anaconda distribution) or Visual Studio Code, etc.

```
print("Hello World!")
```

Variables in Python

Creating variables:

```
> x = 3
> x
3
```

We can use variables in a calculation like this:

```
> x = 3
> y = 3*x
> print(y)
```

We can implement the formula $y(x) = ax + b$ like this:

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

```
> a = 2
> b = 4

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

A variable can have a short name (like x and y) or a more descriptive name (sum, amount, etc).
You don need to define the variables before you use them (like you need to to in, e.g., C/C++/C).

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)
```

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

Conditions

We have the following Conditions we can use in Python:

```
a == b # Equals
a != b # Not Equals
a < b # Less than
a <= b # Less than or equal to
a > b # Greater than
a >= b # Greater than or equal to
```

If – Else - Elif (Conditions)

- In Python you use one of the following or a combination of those:
- If
- If Else
- Elif (known as “Else If“ in most other programming languages)

If

Note also the colon (:)

```
a = 5
b = 8

if a > b:
    print("a is greater than b")

if b > a:
    print("b is greater than a")

if a == b:
    print("a is equal to b")
```

Note! Python uses indentation (spaces)

Other Programming Languages uses curly brackets {} or Begin .. End

Try to change the values for the variables a and b

If - Else

If you have 2 conditions that you need to check, you can use If – Else:

```
a = 5
b = 8

if a > b:
    print("a is greater than b")
else:
    print("b is greater than a or a and b are equal")
```

Elif

If you have more than 2 different conditions you need to check, you typically use Elif:

```
a = 5
b = 8

if a > b:
    print("a is greater than b")
elif b > a:
    print("b is greater than a")
elif a == b:
    print("a is equal to b")
```

Note! Python uses "elif" not "elseif" like many other programming languages do

If need, you can also add an Else at the end for handling “all other conditions”

Arrays

An array is a special variable, which can hold more than one value at a time

Example:

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

Python does not have built-in support for Arrays, but Python Lists can be used instead.

Length of an Array (List):

```
N = len(data)
```

Get a specific element (Indexing):

```
x = data[2]
```

Change a specific element:

```
data[2] = 7.3
```

Add a new value to the end of the Array (List):

```
data.append(11.4)
```

For more advanced use of Arrays in Python you will have to import a library, like the **NumPy** library.

Using Arrays in Functions

Using Arrays in Functions

Note! statistics is a sub library in the Python Standard Library

Example:

```
from statistics import *

data = [1.6, 3.4, 5.5, 9.4]

m = mean(data)
sd = stdev(data)
datamin = min(data)
datamax = max(data)
```

Arrays of Strings

You can also create an Array (List) of Strings:

```
cars = ["Ford", ”Toyota", ”Tesla"]
```

Some useful Functions for manipulating the Array (List):

```
x = cars[1]

x = len(cars)

cars.append(”Porche")

cars.remove(”Tesla")

cars.sort()
```

For Loops

A For loop is used for iterating over a sequence. I guess all your programs will use one or more For loops. So if you have not used For loops before, make sure to learn it now.

Example:

```
cars = ["Ford", "Toyota", "Tesla"]

for car in cars:
  print(car)
```

Array (List) of Strings

Note! Python uses indentation (spaces) →

Other Programming Languages uses curly brackets {} or Begin .. End

Example:

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

for x in data:
    print (x)
```

Array (List) of Numbers

For Loops

The **range()** function is handy to use in For Loops:

```
N = 10

for x in range(N):
  print(x)
```

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

You can also use the range() function like this:

```
start = 4
stop= 12 #but not including

for x in range(start, stop):
  print(x)
```

Or like this:

```
start = 4
stop = 12 #but not including
step = 2

for x in range(start, stop, step):
  print(x)
```

For Loops - Example

Example: Find the Sum and Average/Mean for some given Data:

```
data = [1, 5, 6, 3, 12, 3]

sum = 0

for x in data:
    sum = sum + x
print(sum)

N = len(data)
mean = sum/N
print(mean)
```

Result:

```
30
5.0
```

While Loops

Example: We want to find for what value of x the function has its minimum value

$$y(x) = 2x^2 + 20x - 22$$

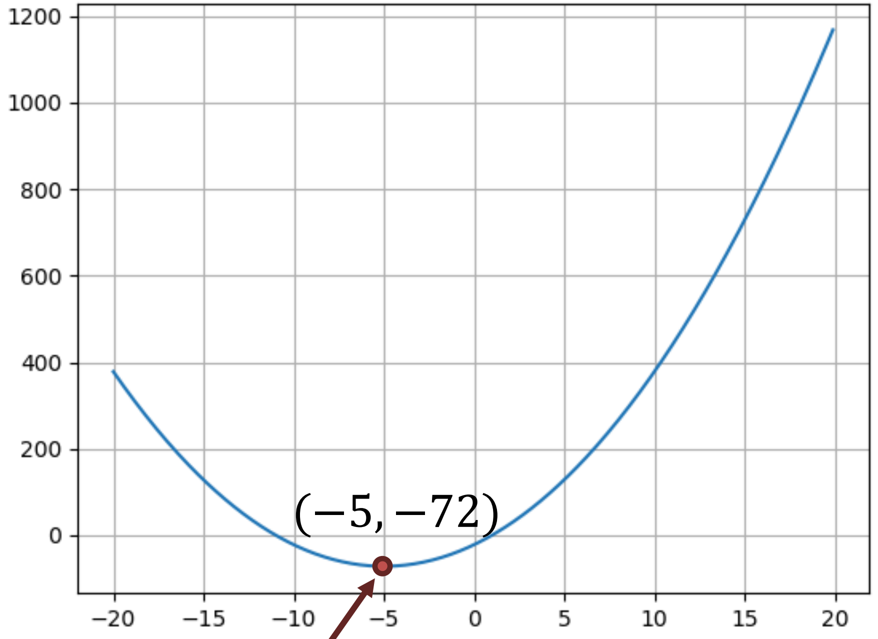
We can of course find the derivative of the function and find where the derivative is equal to zero:

$$\frac{dy}{dx} = 4x + 20 = 0$$

This gives:

$$x_{min} = -5$$

$$y(-5) = 50 - 100 - 22 = -72$$



The minimum of the function

While Loops

Python Solution:

Example: We want to find for what value of x the function has its minimum value

$$y(x) = 2x^2 + 20x - 22$$

We use Python to iterate through all values of $y(x)$ using a While Loop. Inside the While Loop we compare $y(i)$ and $y(i+1)$. If $y(i+1)$ is larger than $y(i)$ we have found the minimum.

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

xstart = -20
xstop = 20
increment = 0.1
x = np.arange(xstart,xstop,increment)
y = 2 * x*x + 20 * x - 22

plt.plot(x,y)
plt.grid()

i = 0

while y[i] > y[i+1]:
    i = i+1

print(x[i])
print(y[i])
```

The Python results becomes the same as the analytical solution:

$(-5, -72) \longleftarrow$

Create Functions

- So far, we have used many of the built-in functions in Python, like print(), plot(), len(), etc.
- There are many built-in functions in Python
- We can also use functions which are part of many of the additional Python Libraries like NumPy, Matplotlib, etc.
- Still, very often we need to make our own functions from scratch

Function Definition

Note that you need to use a colon ":" at the end of line where you define the function.

Note! Python uses indentation (spaces)

Other Programming Languages uses curly brackets {} or Begin .. End

```
def FunctionName:
    <statement-1>
    .
    .
    <statement-N>
    return ...
```

The return value should be stated here

Create Functions

Create the Function:

```
def add(x,y):
    z = x + y
    return z
```

Using the Function within the same script:

```
def add(x,y):
    z = x + y
    return z

# Using the Function:
x = 2
y = 5

z = add(x,y)

print(z)
```

Create Functions in a Separate File

- Although you can mix functions and code in one file, it is much better to create the functions in separate .py files
- In that way you can easily reuse the function in different Python scripts

1

We start by creating a separate Python File, e.g., “**myfunctions.py**“ for the function:

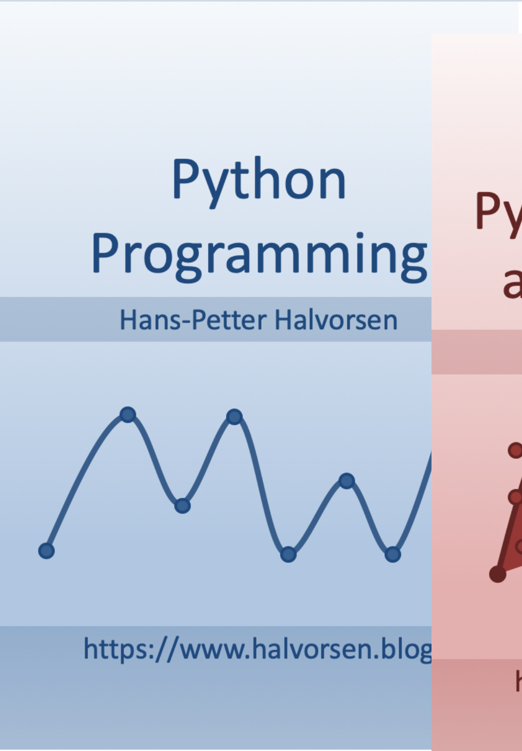
myfunctions.py:

```
def average(x,y):

    return (x + y)/2
```

2 Next, we create a new Python File (e.g., “**testaverage.py**“) where we use the function we created:

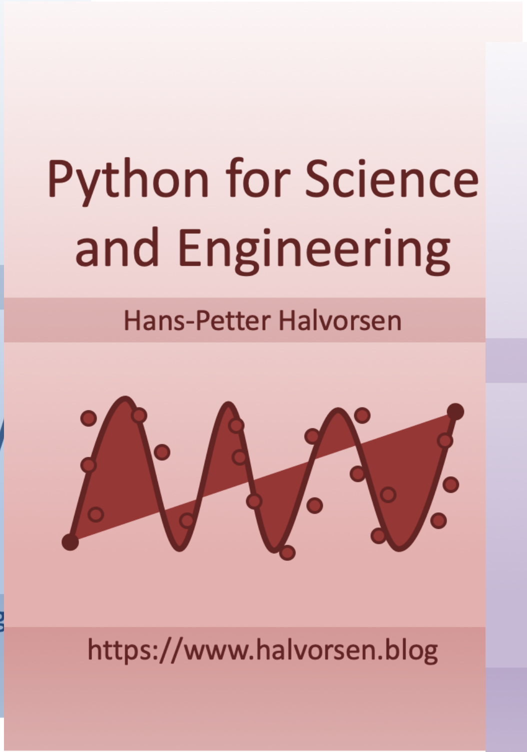
```
from myfunctions import average

a = 2
b = 3

c = average(a,b)

print(c)
```

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

Python Software Development

Do you want to learn Software Development?

OK Cancel

https://www.halvorsen.blog

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