https://www.halvorsen.blog

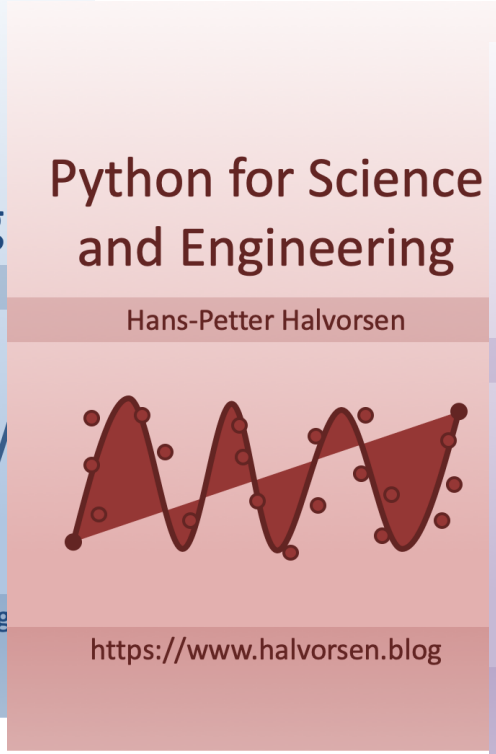
Create Functions with Python

Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

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

Additional Python Resources



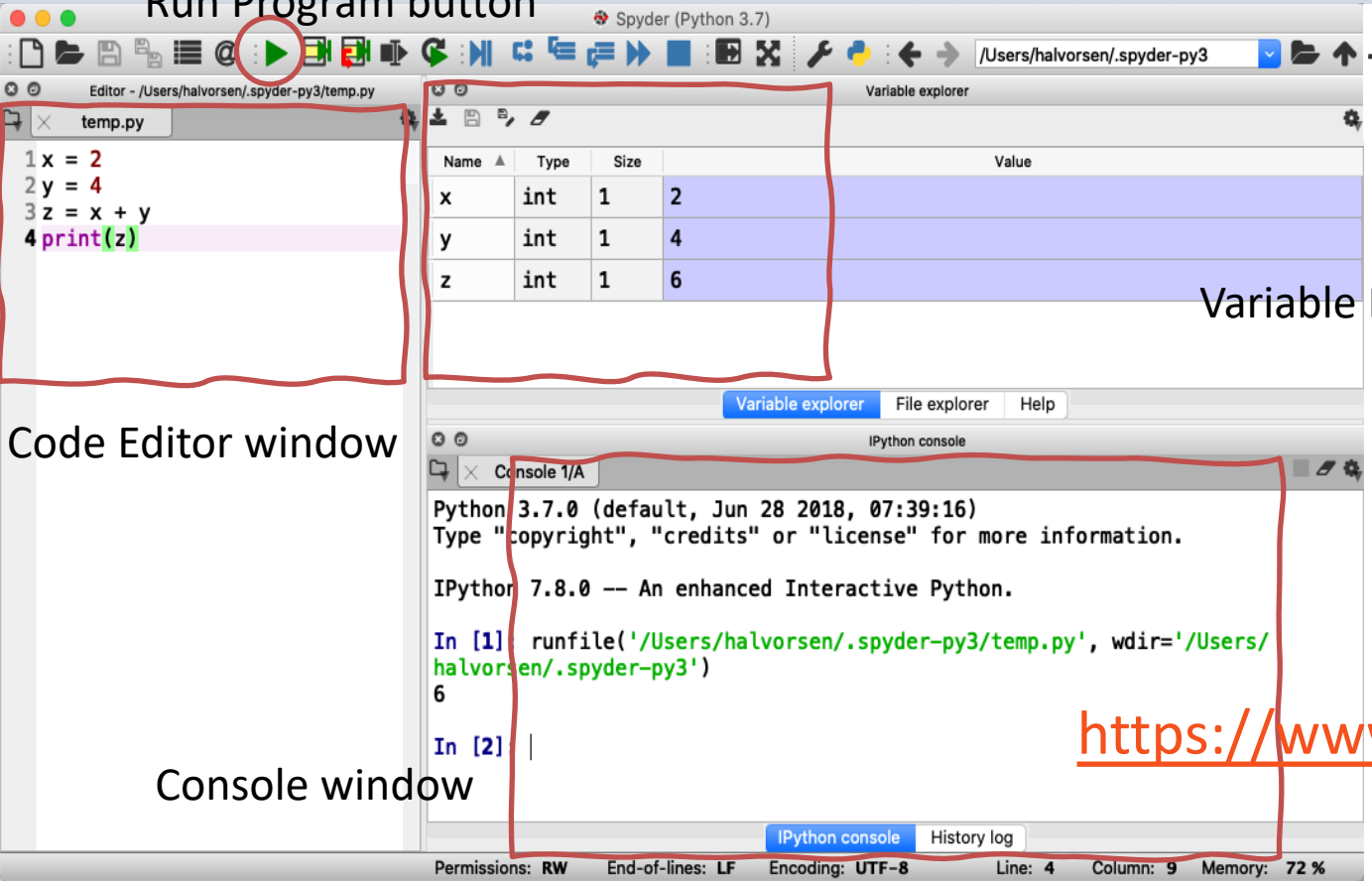
https://www.halvorsen.blog/documents/programming/python/

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

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)
```

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

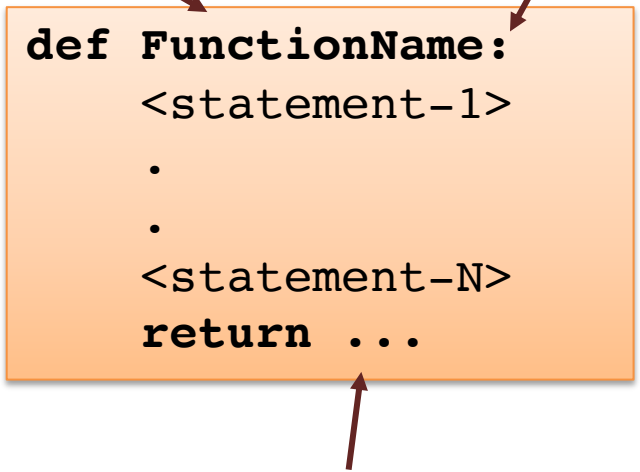
Create Functions

- 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.

The Name of the function

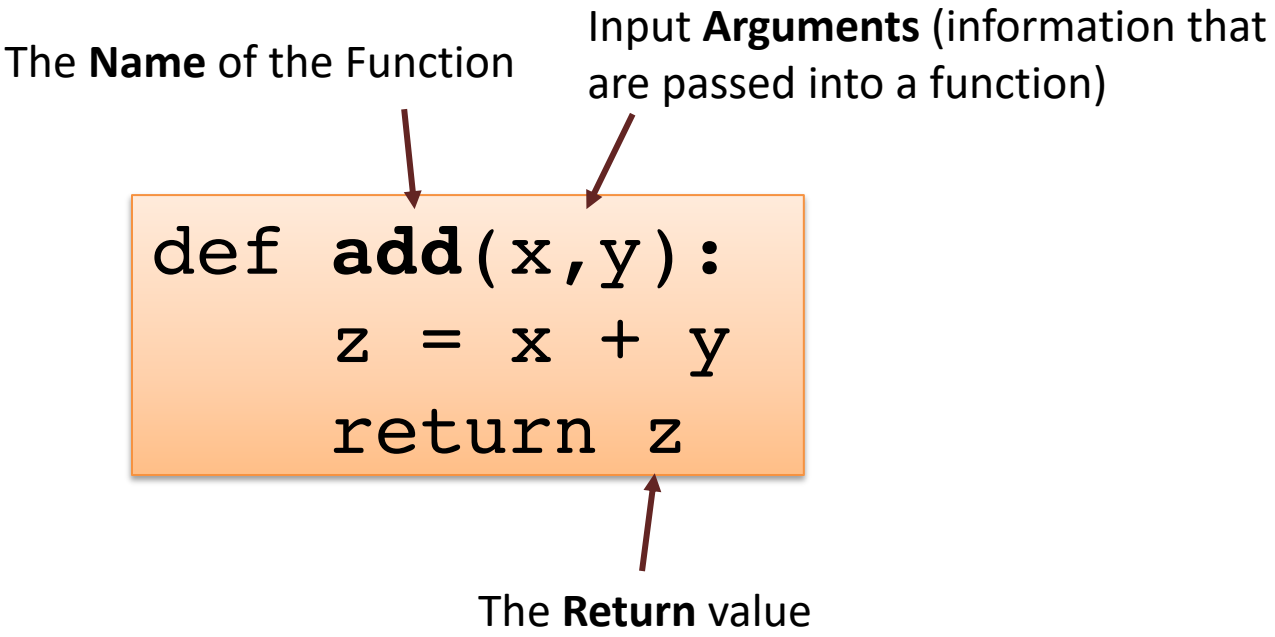


Note! Python uses indentation (spaces)

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

The return value should be stated here

Function Example



Create Functions

Create the Function:

```
def add(x,y):

    return x + y
```

Using the Function within the same script:

```
def add(x,y):

    return x + y

# 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)
```

Multiple Return Values

Create Function

```
def stat(x):

    totalsum = 0

    #Find the Sum of all the numbers
    for x in data:
      totalsum = totalsum + x

    #Find the Mean or Average of all the numbers
    N = len(data)
    mean = totalsum/N
    return totalsum, mean
```

Functions with multiple return values:
Typically we want to return more than one value from a function

Use Function

```
# Using the function
data = [1, 5, 6, 3, 12, 3]

totalsum, mean = stat(data)
print(totalsum, mean)
```

In general, it is recommended to create the function(s) in separate File(s) as shown in the previous example

Creating Python Modules

- As your program gets longer, you may want to split it into several files for easier maintenance. You may also want to use a handy function that you have written in several programs without copying its definition into each program.
- To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter (the Python Console window).

Creating Python Modules

Example:
We want to create a Python Module that has functions for converting from **Celsius to Fahrenheit** and from **Fahrenheit to Celsius**

Necessary Formulas to implement in the Python code:

$$T_f = (T_c \times 9/5) + 32$$

$$T_c = \left(T_f - 32\right) \times (5/9)$$

Example cont.

1

First, we create a Python module with the following functions (“fahrenheit.py“):

```
def c2f(Tc):

    Tf = (Tc * 9/5) + 32
    return Tf


def f2c(Tf):

    Tc = (Tf - 32)*(5/9)
    return Tc
```

2 Then, we create a Python script for testing the functions (e.g., “testfahrenheit.py“):

```
from fahrenheit import c2f, f2c

Tc = 0
Tf = c2f(Tc)
print("Fahrenheit: " + str(Tf))


Tf = 32
Tc = f2c(Tf)
print("Celsius: " + str(Tc))
```

Example cont.

Different options:

```
from fahrenheit import c2f, f2c

Tc = 0
Tf = c2f(Tc)
print("Fahrenheit: " + str(Tf))

Tf = 32
Tc = f2c(Tf)
print("Celsius: " + str(Tc))
```

Basically, we use the Module we have created just like an external Python Library like **NumPy**, etc.

```
import fahrenheit as fa

Tc = 0
Tf = fa.c2f(Tc)
print("Fahrenheit: " + str(Tf))


Tf = 32
Tc = fa.f2c(Tf)
print("Celsius: " + str(Tc))
```

```
from fahrenheit import *

Tc = 0
Tf = c2f(Tc)
print("Fahrenheit: " + str(Tf))

Tf = 32
Tc = f2c(Tf)
print("Celsius: " + str(Tc))
```

Creating Python Modules

- For larger Python applications you should definitely divide your code into different Python Modules
- The Python Modules should be divided into different topics, like one Module for, e.g., Statistics, one for Complex Numbers, …
- In that way the structure of your application becomes much better
- And you can reuse the Modules in other applications
- You only need to change the code one place
- It is easier to find Bugs
- etc.

https://www.halvorsen.blog

Advanced Functions

Hans-Petter Halvorsen

Arbitrary Arguments, *args

If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.

Example:

```
# Create/Define the Function:
def cars(*car):
    n = len(car) #Find Number of cars
    return n

# Using the Function:
n = cars("Ford", "Toyota", "Tesla")
print(n)

n = cars("Ford", "Tesla")
print(n)

n = cars("Ford", "Tesla", "Volvo", "Toyota", "VW")
print(n)
```

Arbitrary Arguments, *args

Modified Example:

```
# Create/Define the Function:
def cars(*car):
    number = len(car) #Find Number of cars
    carnames = ""
    for x in car:
        carnames = carnames + ", " + x

    return number, carnames

# Using the Function:
n, names = cars("Ford", "Toyota", "Tesla")
print(n, names)

n, names = cars("Ford", "Tesla")
print(n, names)

n, names = cars("Ford", "Tesla", "Volvo", "Toyota", "VW")
print(n, names)
```

Key – Value Arguments

Another option is to send arguments with the *key* = *value* syntax. See example below:

```
# Create/Define the Function:
def cars(regno, cartype, carcolor):
   carinfo = regno + " - " + cartype + " - " + carcolor
   return carinfo

# Using the Function:
info = cars(cartype="Ford", carcolor="Blue", regno="AR30675")
print(info)

info = cars(cartype="Toyota", carcolor="Green", regno="NE30675")
print(info)
```

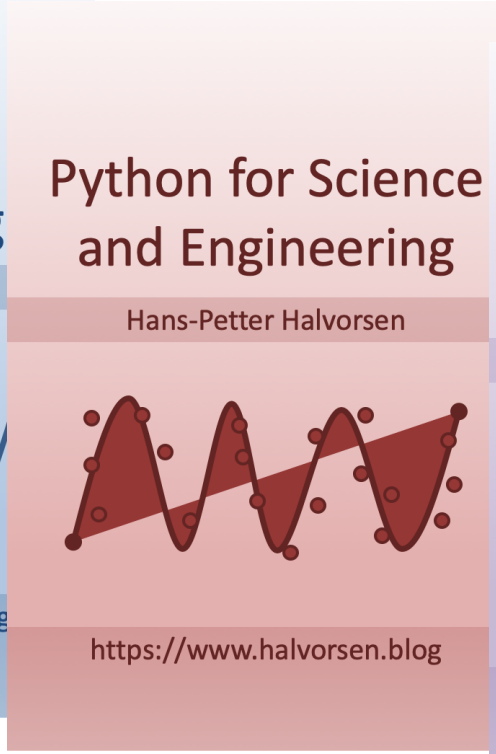
Note That the order of the arguments does not matter in this case

Key – Value Arguments

If you do not know how many Keyword arguments that will be passed into your function, add two asterisk: ** before the parameter name in the function definition. See example below:

```
# Create/Define the Function:
def cars(**cardata):
    carinfo = ""

    for x in cardata:
        carinfo = carinfo + x + ": " + cardata[x] + ", "

    return carinfo


# Using the Function:
info = cars(cartype = "Ford", carcolor = "Blue", regno = "AR30675")
print(info)

info = cars(carcolor = "Green", cartype = "Toyota")
print(info)

info = cars(cartype = "Tesla", carmodel = "Model S", carcolor="Black")
print(info)
```

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