



# SciPy



**tutorialspoint**  
SIMPLY EASY LEARNING

[www.tutorialspoint.com](http://www.tutorialspoint.com)



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

## About the Tutorial

---

SciPy, a scientific library for Python is an open source, BSD-licensed library for mathematics, science and engineering. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The main reason for building the SciPy library is that, it should work with NumPy arrays. It provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization.

This is an introductory tutorial, which covers the fundamentals of SciPy and describes how to deal with its various modules.

## Audience

---

This tutorial is prepared for the readers, who want to learn the basic features along with the various functions of SciPy. After completing this tutorial, the readers will find themselves at a moderate level of expertise, from where they can take themselves to higher levels of expertise.

## Prerequisites

---

Before proceeding with the various concepts given in this tutorial, it is being expected that the readers have a basic understanding of Python. In addition to this, it will be very helpful, if the readers have some basic knowledge of other programming languages.

SciPy library depends on the NumPy library, hence learning the basics of NumPy makes the understanding easy.

## Copyright and Disclaimer

---

© Copyright 2017 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at [contact@tutorialspoint.com](mailto:contact@tutorialspoint.com)

## Table of Contents

---

About the Tutorial .....	i
Audience.....	i
Prerequisites.....	i
Copyright and Disclaimer .....	i
Table of Contents .....	ii
<b>1. SciPy – Introduction .....</b>	<b>1</b>
<b>2. SciPy – Environment Setup .....</b>	<b>3</b>
<b>3. SciPy – Basic Functionality .....</b>	<b>4</b>
NumPy Vector.....	4
Intrinsic NumPy Array Creation .....	4
Matrix .....	6
<b>4. SciPy – Cluster .....</b>	<b>7</b>
K-Means Implementation in SciPy.....	7
Compute K-Means with Three Clusters.....	8
<b>5. SciPy – Constants .....</b>	<b>10</b>
SciPy Constants Package.....	10
List of Constants Available.....	10
<b>6. SciPy – Fftpack .....</b>	<b>14</b>
Fast Fourier Transform .....	14
Discrete Cosine Transform .....	15
<b>7. SciPy – Integrate .....</b>	<b>17</b>
Single Integrals .....	18
Multiple Integrals .....	18
Double Integrals .....	18
<b>8. SciPy – Interpolate .....</b>	<b>20</b>
What is Interpolation?.....	20
1-D Interpolation .....	21
Splines .....	22
<b>9. SciPy – Input &amp; Output.....</b>	<b>25</b>
<b>10. SciPy – Linalg.....</b>	<b>27</b>
Linear Equations.....	27
Finding a Determinant.....	28
Eigenvalues and Eigenvectors .....	29
Singular Value Decomposition.....	29
<b>11. SciPy – Ndimage.....</b>	<b>31</b>
Opening and Writing to Image Files .....	31
Filters.....	35
Edge Detection .....	37
<b>12. SciPy – Optimize .....</b>	<b>40</b>

Nelder–Mead Simplex Algorithm .....	40
Least Squares.....	41
Root finding .....	42
<b>13. SciPy – Stats .....</b>	<b>44</b>
Normal Continuous Random Variable .....	44
Uniform Distribution .....	45
Descriptive Statistics.....	46
T-test .....	47
<b>14. SciPy – CSGraph .....</b>	<b>49</b>
Graph Representations.....	49
Obtaining a List of Words .....	51
<b>15. SciPy – Spatial .....</b>	<b>54</b>
Delaunay Triangulations .....	54
Coplanar Points .....	55
Convex hulls.....	55
<b>16. SciPy – ODR.....</b>	<b>57</b>
<b>17. SciPy – Special Package .....</b>	<b>60</b>

# 1.SciPy – Introduction

SciPy, pronounced as Sigh Pi, is a scientific python open source, distributed under the BSD licensed library to perform Mathematical, Scientific and Engineering Computations.

The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays and provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install and are free of charge. NumPy and SciPy are easy to use, but powerful enough to depend on by some of the world's leading scientists and engineers.

## SciPy Sub-packages

SciPy is organized into sub-packages covering different scientific computing domains. These are summarized in the following table:

<a href="#"><u>scipy.cluster</u></a>	Vector quantization / Kmeans
<a href="#"><u>scipy.constants</u></a>	Physical and mathematical constants
<a href="#"><u>scipy.fftpack</u></a>	Fourier transform
<a href="#"><u>scipy.integrate</u></a>	Integration routines
<a href="#"><u>scipy.interpolate</u></a>	Interpolation
<a href="#"><u>scipy.io</u></a>	Data input and output
<a href="#"><u>scipy.linalg</u></a>	Linear algebra routines
<a href="#"><u>scipy.ndimage</u></a>	n-dimensional image package
<a href="#"><u>scipy.odr</u></a>	Orthogonal distance regression
<a href="#"><u>scipy.optimize</u></a>	Optimization
<a href="#"><u>scipy.signal</u></a>	Signal processing
<a href="#"><u>scipy.sparse</u></a>	Sparse matrices

<a href="#"><u>scipy.spatial</u></a>	Spatial data structures and algorithms
<a href="#"><u>scipy.special</u></a>	Any special mathematical functions
<a href="#"><u>scipy.stats</u></a>	Statistics

## Data Structure

The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for Linear Algebra, Fourier Transforms and Random Number Generation, but not with the generality of the equivalent functions in SciPy.

# 2.SciPy – Environment Setup

Standard Python distribution does not come bundled with any SciPy module. A lightweight alternative is to install SciPy using the popular Python package installer,

```
pip install pandas
```

If we install the **Anaconda Python package**, Pandas will be installed by default. Following are the packages and links to install them in different operating systems.

## Windows

**Anaconda** (from <https://www.continuum.io>) is a free Python distribution for the SciPy stack. It is also available for Linux and Mac.

**Canopy** (<https://www.enthought.com/products/canopy/>) is available free, as well as for commercial distribution with a full SciPy stack for Windows, Linux and Mac.

**Python (x,y)**: It is a free Python distribution with SciPy stack and Spyder IDE for Windows OS. (Downloadable from <http://python-xy.github.io/>)

## Linux

Package managers of respective Linux distributions are used to install one or more packages in the SciPy stack.

### Ubuntu

We can use the following path to install Python in Ubuntu.

```
sudo apt-get install python-numpy python-scipy python-  
matplotlibpythonipython-notebook python-pandas python-sympy python-nose
```

### Fedora

We can use the following path to install Python in Fedora.

```
sudo yum install numpyscipy python-matplotlibpython python-pandas sympy  
python-nose atlas-devel
```

# 3.SciPy – Basic Functionality

By default, all the NumPy functions have been available through the SciPy namespace. There is no need to import the NumPy functions explicitly, when SciPy is imported. The main object of NumPy is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy, dimensions are called as **axes**. The number of axes is called as **rank**.

Now, let us revise the basic functionality of Vectors and Matrices in NumPy. As SciPy is built on top of NumPy arrays, understanding of NumPy basics is necessary. As most parts of linear algebra deals with matrices only.

## NumPy Vector

---

A Vector can be created in multiple ways. Some of them are described below.

### Converting Python array-like objects to NumPy

Let us consider the following example.

```
import numpy as np
list = [1,2,3,4]
arr = np.array(list)
print arr
```

The output of the above program will be as follows.

```
[1 2 3 4]
```

## Intrinsic NumPy Array Creation

---

NumPy has built-in functions for creating arrays from scratch. Some of these functions are explained below.

### Using zeros()

The zeros(shape) function will create an array filled with 0 values with the specified shape. The default dtype is float64. Let us consider the following example.

```
import numpy as np
print np.zeros((2, 3))
```

The output of the above program will be as follows.

```
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```



## Using ones()

The ones(shape) function will create an array filled with 1 values. It is identical to zeros in all the other respects. Let us consider the following example.

```
import numpy as np
print np.ones((2, 3))
```

The output of the above program will be as follows.

```
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

## Using arange()

The arange() function will create arrays with regularly incrementing values. Let us consider the following example.

```
import numpy as np
print np.arange(7)
```

The above program will generate the following output.

```
array([0, 1, 2, 3, 4, 5, 6])
```

## Defining the data type of the values

Let us consider the following example.

```
import numpy as np
arr = np.arange(2, 10, dtype=np.float)
print arr
print "Array Data Type :",arr.dtype
```

The above program will generate the following output.

```
[ 2.  3.  4.  5.  6.  7.  8.  9.]
Array Data Type : float64
```

## Using linspace()

The linspace() function will create arrays with a specified number of elements, which will be spaced equally between the specified beginning and end values. Let us consider the following example.

```
import numpy as np
print np.linspace(1., 4., 6)
```

The above program will generate the following output.

```
array([ 1. ,  1.6,  2.2,  2.8,  3.4,  4. ])
```

## Matrix

A matrix is a specialized 2-D array that retains its 2-D nature through operations. It has certain special operators, such as `*` (matrix multiplication) and `**` (matrix power). Let us consider the following example.

```
import numpy as np
print np.matrix('1 2; 3 4')
```

The above program will generate the following output.

```
matrix([[1, 2],
        [3, 4]])
```

## Conjugate Transpose of Matrix

This feature returns the (complex) conjugate transpose of **self**. Let us consider the following example.

```
import numpy as np
mat = np.matrix('1 2; 3 4')
print mat.H
```

The above program will generate the following output.

```
matrix([[1, 3],
        [2, 4]])
```

## Transpose of Matrix

This feature returns the transpose of self. Let us consider the following example.

```
import numpy as np
mat = np.matrix('1 2; 3 4')
mat.T
```

The above program will generate the following output.

```
matrix([[1, 3],
        [2, 4]])
```

When we transpose a matrix, we make a new matrix whose rows are the columns of the original. A conjugate transposition, on the other hand, interchanges the row and the column index for each matrix element. The inverse of a matrix is a matrix that, if multiplied with the original matrix, results in an identity matrix.

End of ebook preview

If you liked what you saw...

Buy it from our store @ <https://store.tutorialspoint.com>