simplilearn | PURDUE UNIVERSITY

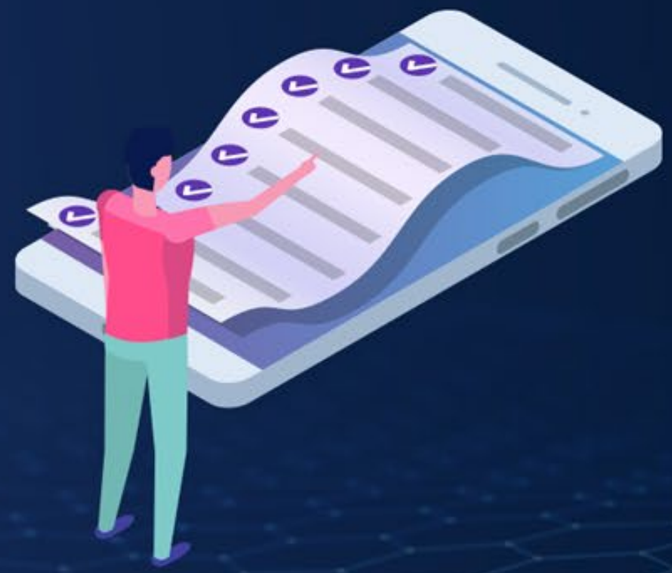
Data Science With Python

Scientific Computing with Python (SciPy)

Learning Objectives

By the end of this lesson, you will be able to:

- Explain the importance of SciPy
- List the characteristics of SciPy
- Explain sub-packages of SciPy
- Discuss SciPy sub-packages, such as optimization, integration, linear algebra, statistics, weave, and IO

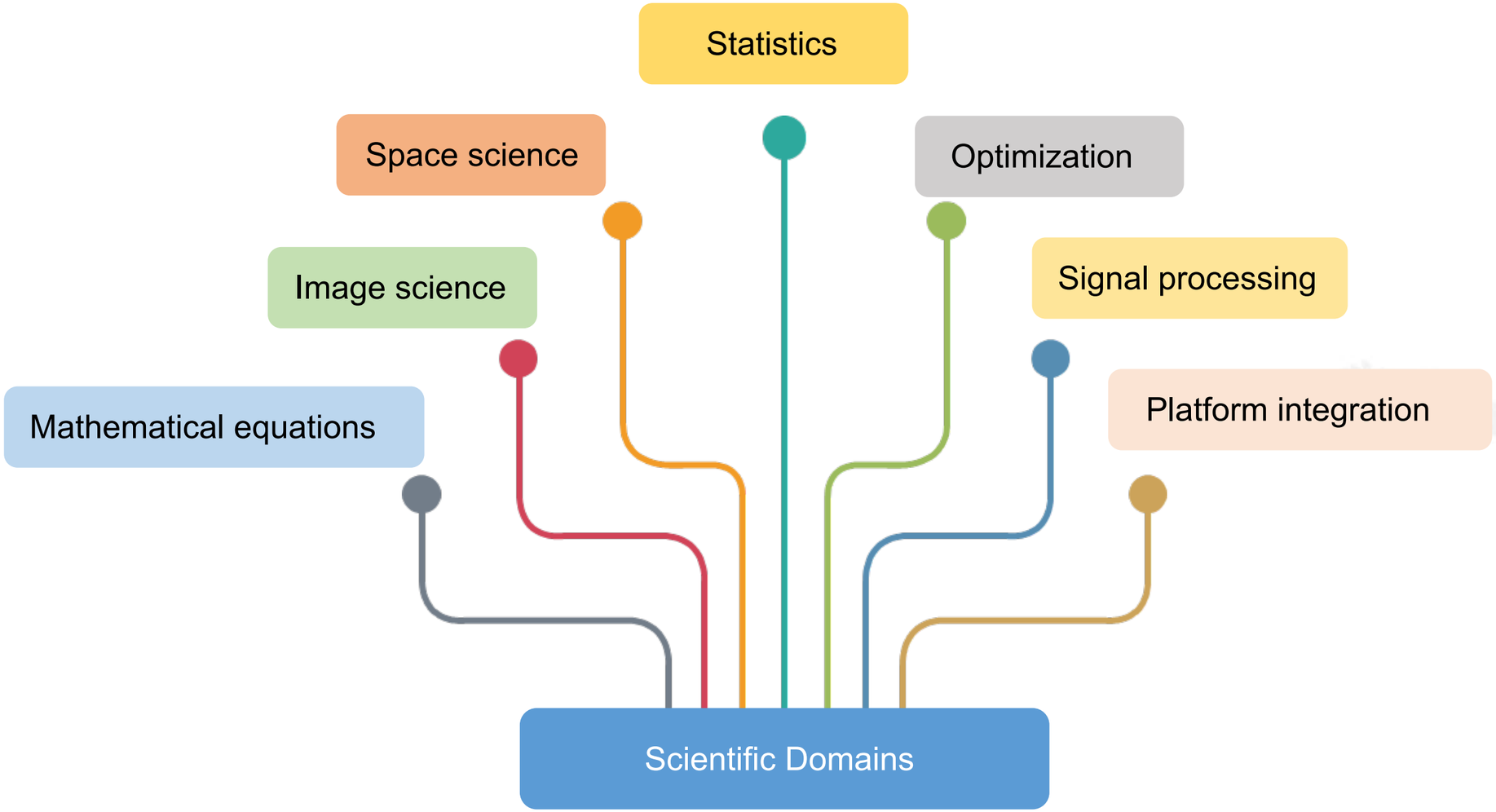


DATA AND
ARTIFICIAL INTELLIGENCE

SciPy and Its Characteristics

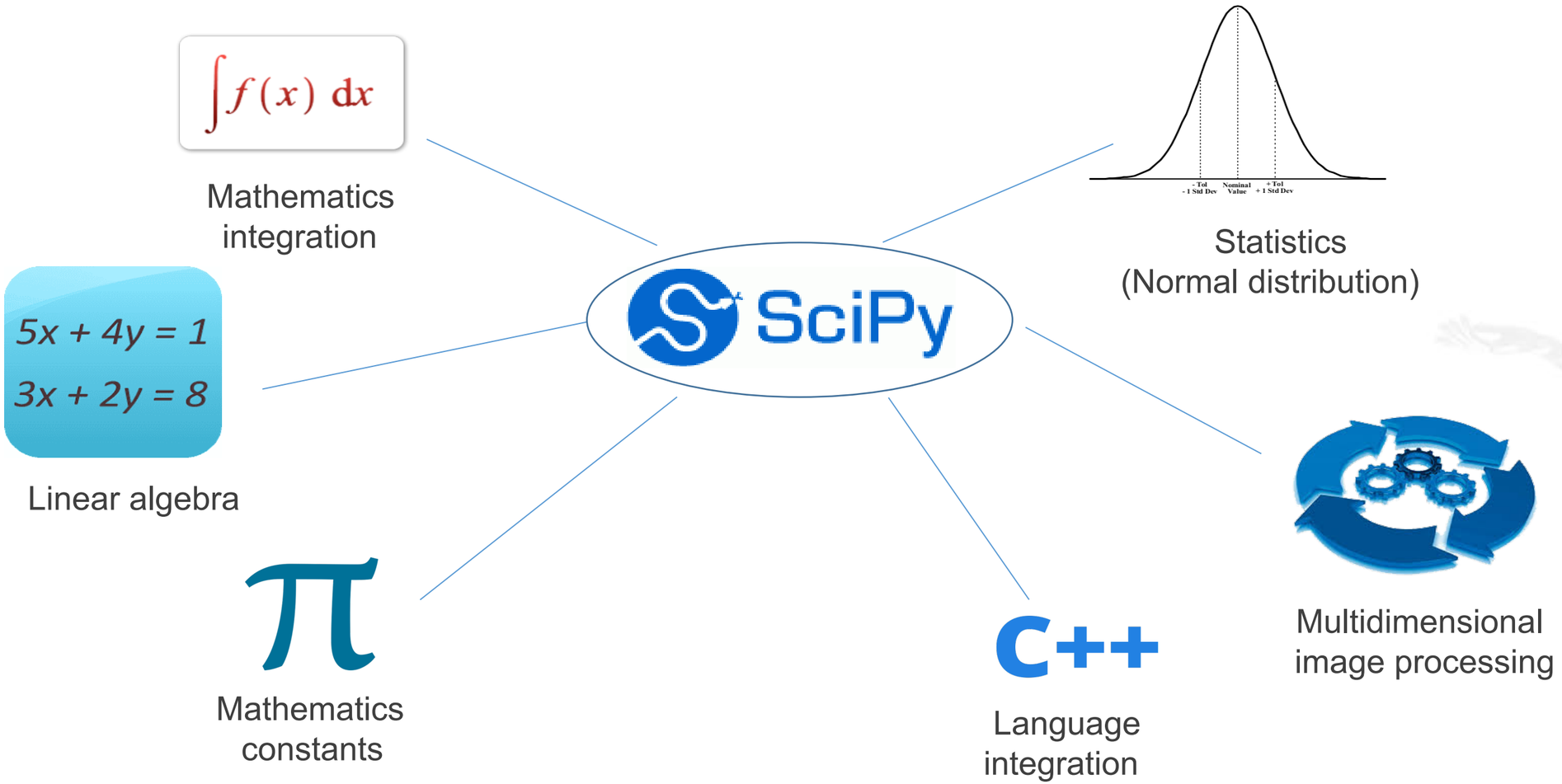
Multiple Scientific Domains

How to handle multiple scientific domains? The solution is SciPy.



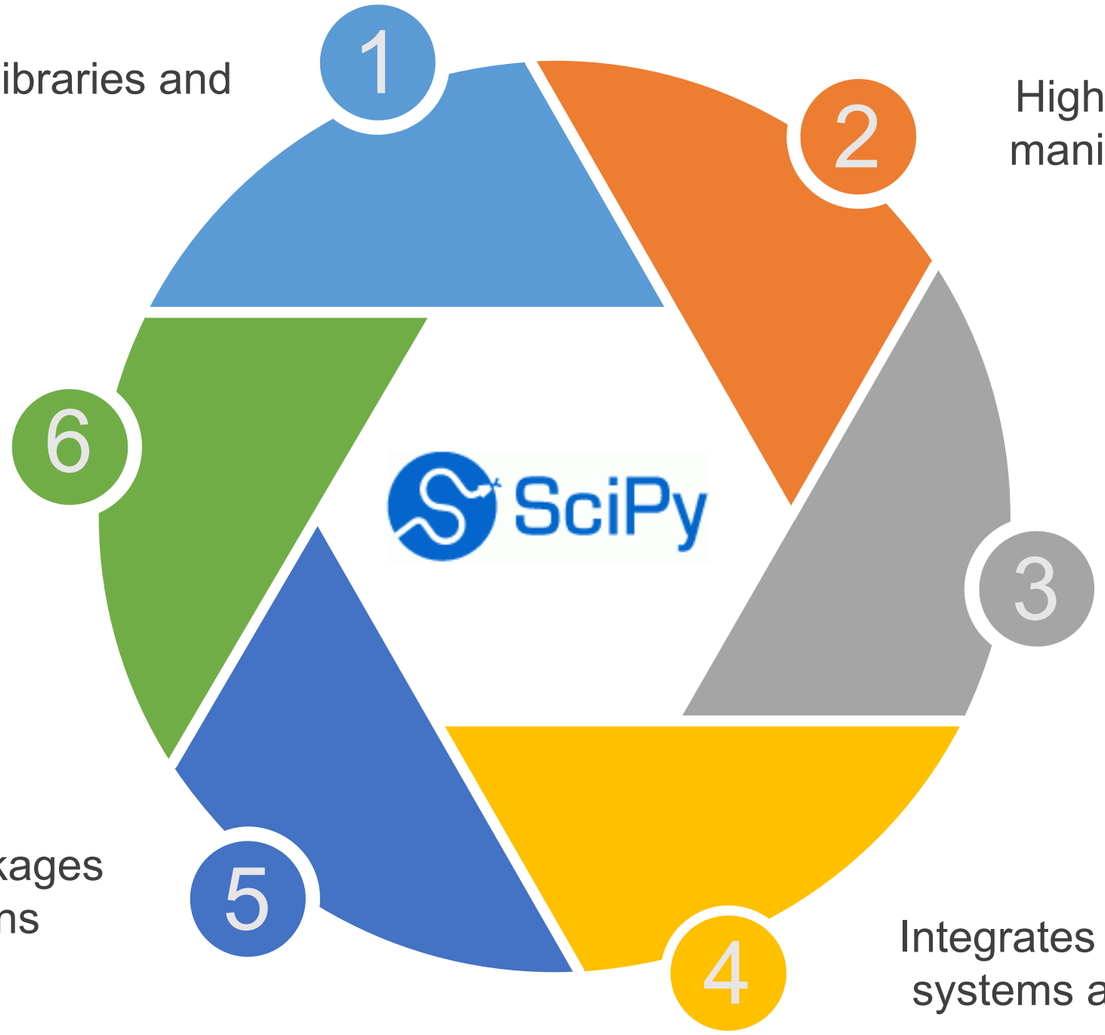
SciPy

SciPy has built-in packages that help in handling the scientific domains.



SciPy and Its Characteristics

1. Built-in mathematical libraries and functions
2. High-level commands for data manipulation and visualization
3. Efficient and fast data processing
4. Integrates well with multiple systems and environments
5. Large collection of sub-packages for different scientific domains
6. Simplifies scientific application development



SciPy Packages

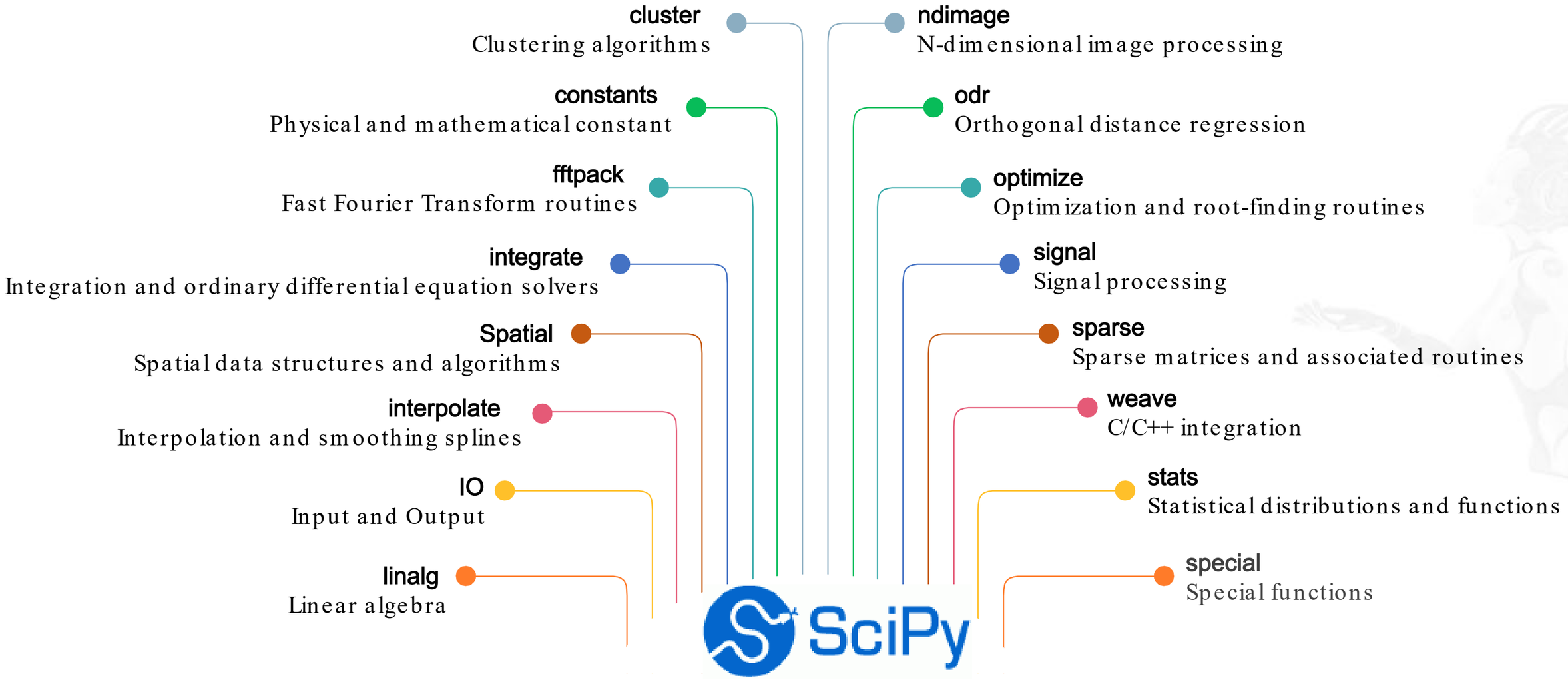
Some widely used packages are:

DATA AND
ARTIFICIAL INTELLIGENCE

Introduction of SciPy Sub -Package

SciPy Sub-Package

SciPy has multiple sub -packages which handle different scientific domains.

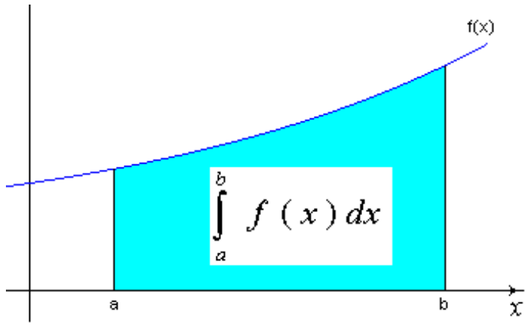


SciPy Sub -Package: Integration

SciPy provides integration techniques that solve mathematical sequences and series, or perform function approximation.

General integration (quad)

integrate.quad(f, a, b)



General multiple integration (dblquad, tplquad, nquad)

- integrate.dblquad()
- integrate.tplquad()
- integrate.nquad()

The limits of all inner integrals need to be defined as functions.

SciPy Sub -Package: Integration

This example shows how to perform quad integration.

```
In [13]: from scipy.integrate import quad
```

Import quad from integrate sub-package

```
In [14]: def integrateFunction(x):
             return x
```

Define function for integration of x

```
In [15]: quad(integrateFunction,0,1)
Out[15]: (0.5, 5.551115123125783e-15)
```

Perform quad integration for function of x for limit 0 to 1

```
In [16]: def integrateFn(x,a,b):
             return x*a+b
```

Define function for ax + b

```
In [17]: a=3
         b=2
```

Declare value of a and b

```
In [18]: quad(integrateFn,0,1,args=(a,b))
Out[18]: (3.5, 3.885780586188048e-14)
```

Perform quad integration and pass functions and arguments

SciPy Sub -Package: Integration

This example shows you how to perform multiple integration.

```
In [20]: import scipy.integrate as integrate
```

Import integrate package sub-package

```
In [21]: def f(x, y):
             return x + y
         integrate.dblquad(f, 0, 1,lambda x: 0, lambda x: 2)
```

Define function for x + y

```
Out[21]: (3.0, 3.3306690738754696e-14)
```

Perform multiple integration using the lambda built -in function

DATA AND
ARTIFICIAL INTELLIGENCE

SciPy Sub -Package: Optimization

SciPy Sub -Package: Optimization

Optimization is a process to improve performance of a system mathematically by fine -tuning the process parameters.

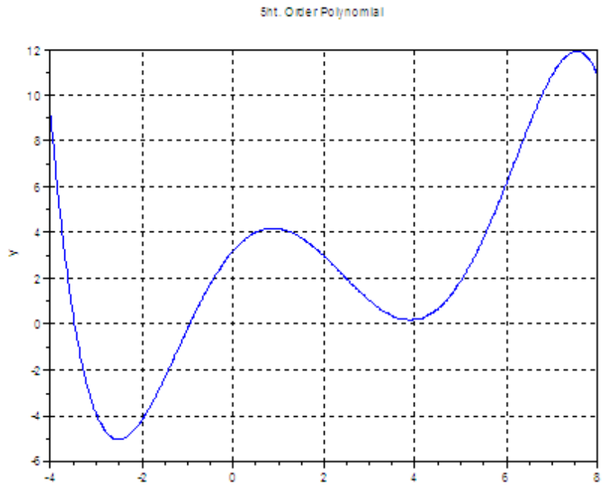
SciPy provides several optimization algorithms, such as bfgs, Nelder -Mead simplex, Newton Conjugate Gradient, COBYLA, or SLSQP.

Minimization functions

optimize.minimize(f, x0, method='BFGS')

lower limit in a given range

Root finding, Curve fitting



root(f, x0, method='hybr')

optimize.curve_fit(f, xdata, ydata)

SciPy Sub -Package: Optimization

```
In [32]: import numpy as np
         from scipy import optimize
```

Import numpy and optimize from SciPy

```
In [33]: def f(x):
             return x**2 + 5*np.sin(x)
```

Define function for X^2 + 5 sin x

```
In [34]: minimaValue = optimize.minimize(f,x0=2,method='bfgs',options={'disp':True})

Optimization terminated successfully.
         Current function value: -3.246394
         Iterations: 4
         Function evaluations: 24
         Gradient evaluations: 8
```

Perform optimize minimize function using bfgs method and options

```
In [35]: minimaValueWithoutOpt = optimize.minimize(f,x0=2,method='bfgs')

In [36]: minimaValueWithoutOpt

Out[36]:       fun: -3.2463942726915382
         hess_inv: array([[ 0.15430551]])
              jac: array([ -8.94069672e-08])
          message: 'Optimization terminated successfully.'
             nfev: 24
              nit: 4
             njev: 8
           status: 0
          success: True
                x: array([-1.11051051])
```

Perform optimize minimize function using bfgs method and without options

SciPy Sub -Package: Optimization

```
In [118]: import numpy as np
          from scipy.optimize import root
          def rootfunc(x):
              return x + 3.5 * np.cos(x)
```

Define function for X+ 3.5 Cos x

```
In [119]: rootValue = root(rootfunc, 0.3)
```

Pass x value in argument for root

```
In [120]: rootValue

Out[120]:      fjac: array([[-1.]])
                fun: array([  2.22044605e-16])
            message: 'The solution converged.'
               nfev: 14
                qtf: array([ -8.32889313e-13])
                  r: array([-4.28198145])
             status: 1
            success: True
                  x: array([-1.21597614])
```

Function value and array values

DATA AND
ARTIFICIAL INTELLIGENCE

SciPy Sub -Package: Linear Algebra

SciPy Sub -Package: Linear Algebra

SciPy provides rapid linear algebra capabilities and contains advanced algebraic functions.

Inverse of matrix	Finding Determinant	Solve Linear systems	Single Value Decomposition (SVD)

This function is used to compute the inverse of the given matrix.

Let's look at the inverse matrix operation.

```
In [65]: import numpy as np
         from scipy import linalg
         matrix = np.array([[10,6],[2,7]])
         matrix

Out[65]: array([[10,  6],
                [ 2,  7]])

In [66]: type(matrix)

Out[66]: numpy.ndarray

In [67]: linalg.inv(matrix)

Out[67]: array([[ 0.12068966, -0.10344828],
                [-0.03448276,  0.17241379]])
```

Import linalg and Define a numpy matrix or array

View the type

Use inv function to inverse the matrix

SciPy Sub -Package: Linear Algebra

SciPy provides rapid linear algebra capabilities and contains advanced algebraic functions.

Inverse of matrix	Finding Determinant	Solve Linear systems	Single Value Decomposition (SVD)

With this function you can compute the value of the determinant for the given matrix.

```
In [68]: import numpy as np
         from scipy import linalg
         matrix = np.array([[4,9],[3,5]])
         matrix

Out[68]: array([[4, 9],
                [3, 5]])

In [69]: linalg.det(matrix)

Out[69]: -7.000000000000001
```

Import linalg and
Define an numpy matrix or array

Use det function to find the determinant value of the matrix

SciPy Sub -Package: Linear Algebra

SciPy provides rapid linear algebra capabilities and contains advanced algebraic functions.

Inverse of matrix	Finding Determinant	Solve Linear systems	Single Value Decomposition (SVD)

Linear equations

$$2x + 3y + z = 21$$
$$-x + 5y + 4z = 9$$
$$3x + 2y + 9z = 6$$

```
In [88]: import numpy as np
         from scipy import linalg

In [89]: numArray = np.array([[2,3,1],[-1,5,4],[3,2,9]])

In [90]: numArrValue = np.array([21,9,6])

In [91]: linalg.solve(numArray,numArrValue)
Out[91]: array([ 4.95,  4.35, -1.95])
```

Import linalg

Use solve method

SciPy Sub -Package: Linear Algebra

SciPy provides rapid linear algebra capabilities and contains advanced algebraic functions.

Inverse of matrix	Finding Determinant	Solve Linear systems	Single Value Decomposition (SVD)

```
In [103]: import numpy as np
          from scipy import linalg
```

Import linalg

```
In [104]: numSvdArr = np.array([[3,5,1],[9,5,7]])
```

Define matrix

```
In [105]: numSvdArr.shape
Out[105]: (2L, 3L)
```

Find shape of ndarray which is 2X3 matrix

```
In [106]: linalg.svd(numSvdArr)
Out[106]: (array([[-0.37879831, -0.92547925],
                  [-0.92547925,  0.37879831]]),
           array([ 13.38464336,   3.29413449]),
           array([[-0.7072066 , -0.4872291 , -0.51231496],
                  [ 0.19208294, -0.82977932,  0.52399467],
                  [-0.68041382,  0.27216553,  0.68041382]]))
```

Use svd function

U (Unitary matrix)

Sigma or square root of eigenvalues

VH is values collected into unitary matrix

Calculate Eigenvalues and Eigenvectors

Problem Statement: Demonstrate how to calculate eigenvalues and eigenvectors

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login** .

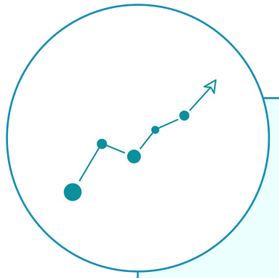
ASSISTED PRACTICE

DATA AND
ARTIFICIAL INTELLIGENCE

SciPy Sub -Package: Statistics

SciPy Sub -Package: Statistics

SciPy provides a very rich set of statistical functions which are:



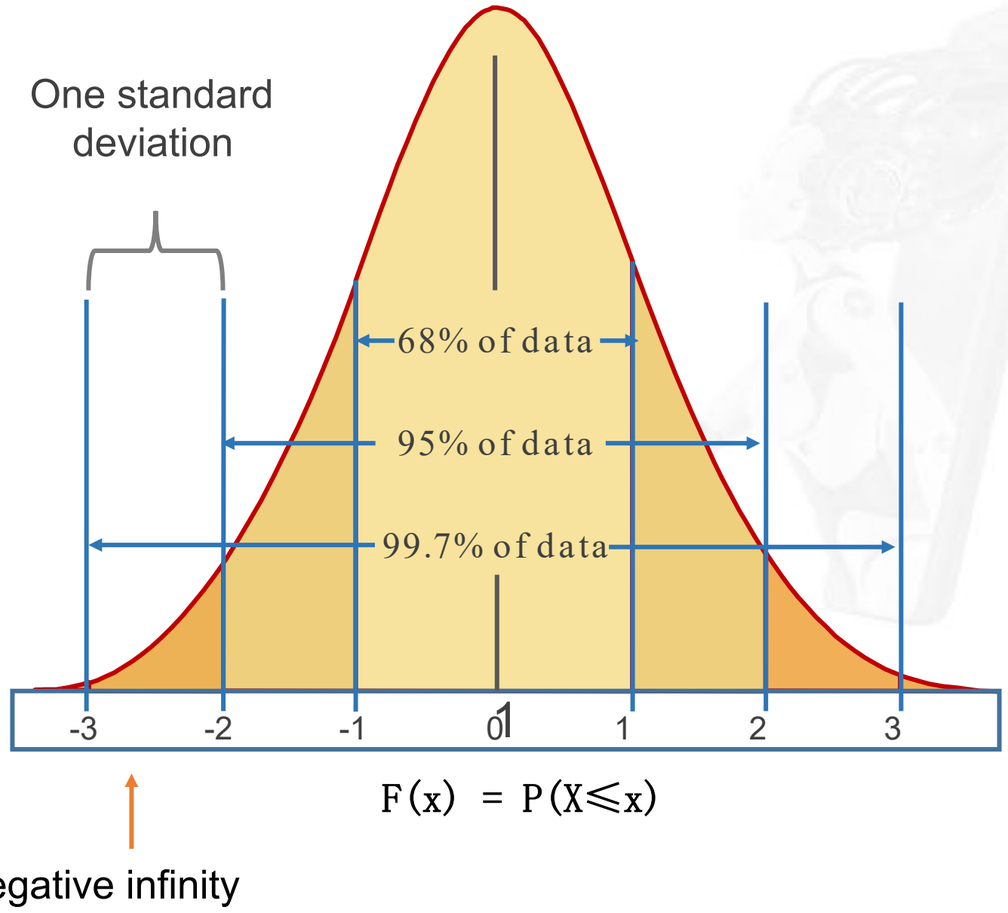
- This package contains distributions for which random variables are generated.
- These packages enable the addition of new routines and distributions. It also offers convenience methods such as pdf(), cdf()
- Following are the statistical functions for a set of data:
  - linear regression: linregress()
  - describing data: describe(), normaltest()

SciPy Sub -Package: Statistics

CDF or Cumulative Distribution Function provides the cumulative probability associated with a function.

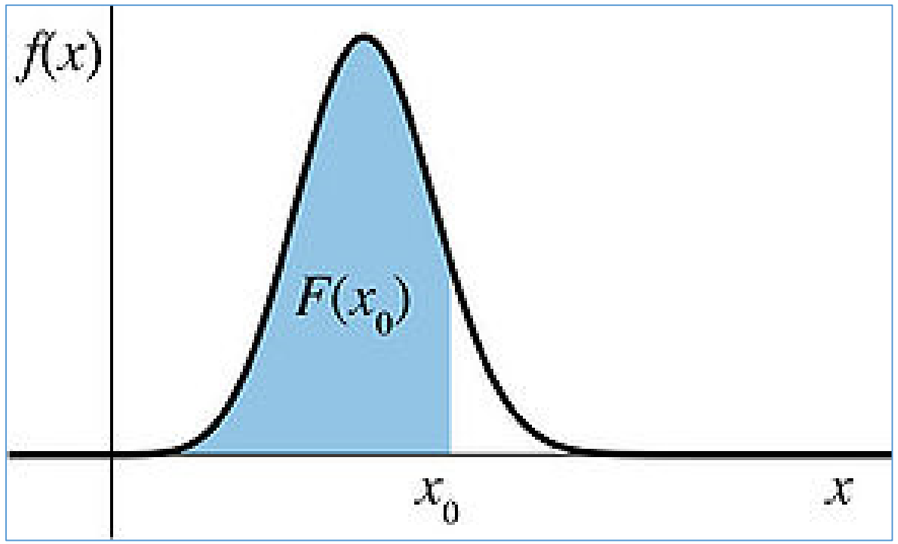
Age Range	Frequency	Cumulative Frequency
0-10	19	19
10-20	55	74
21-30	23	97
31-40	36	133
41-50	10	143
51-60	17	160

Total number of persons within this age



SciPy Sub -Package: Statistics

Probability Density Function, or **PDF,** of a continuous random variable is the derivative of its Cumulative Distribution Function, or CDF.



$$f(x) = \frac{dF(x)}{dx}$$

Derivative of CDF

SciPy Sub -Package: Statistics

Shown here are functions used to perform Normal Distribution:

```
In [108]: from scipy.stats import norm
```

Import norm for normal distribution

```
In [110]: norm.rvs(loc=0,scale=1,size=10)

Out[110]: array([-0.16337774,  0.39039561,  0.85642826,  0.30134358, -1.86009474,
                 -0.29621603,  0.03863757,  0.23727056, -1.42395316, -0.5730162 ])
```

rvs for Random variables

```
In [112]: norm.cdf(5,loc=1,scale=2)

Out[112]: 0.9772498680518207
```

cdf for Cumulative Distribution Function

```
In [113]: norm.pdf(9,loc=0,scale=1)

Out[113]: 1.027977357166891e-18
```

pdf for Probability Density Function for random distribution

loc and **scale** are used to adjust the location and scale of the data distribution.

DATA AND
ARTIFICIAL INTELLIGENCE

SciPy Sub -Package: Weave and IO

SciPy Sub -Package: Weave

The weave package provides ways to modify and extend any supported extension libraries.

Features of Weave Package:

- Includes C/C++ code within Python code
- Speed ups of 1.5x to 30x compared to algorithms written in pure Python

Two main functions of weave::

- inline() compiles and executes C/C++ code on the fly
- blitz() compiles NumPy Python expressions for fast execution

SciPy Sub -Package: IO

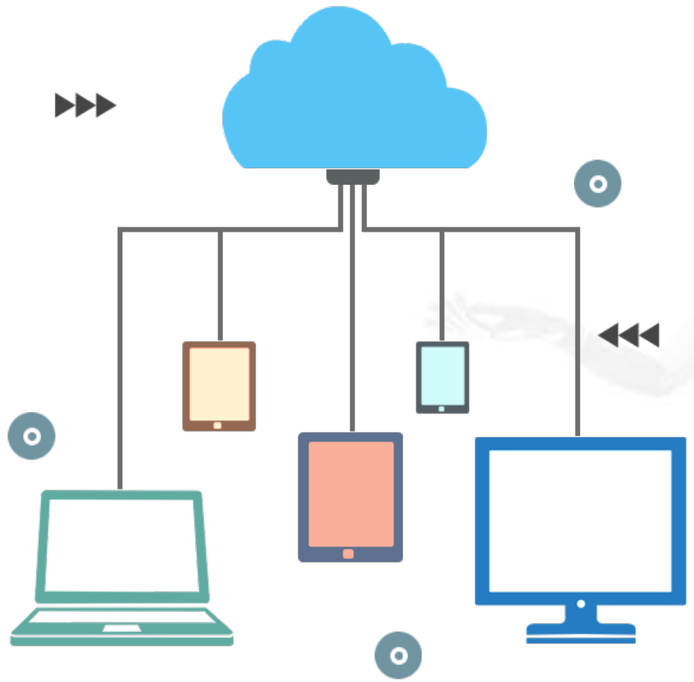
The IO package provides a set of functions to deal with several kinds of file formats.

It offers a set of functions to deal with file formats that include:

- MatLab file
- IDL files
- Matrix market files
- Wav sound files
- Arff files
- Netcdf files

Package provides additional files and its corresponding methods such as:

- Numpy.loadtxt()/Numpy.savetxt()
- Numpy.genfromtxt()/Numpy.recfromcsv()
- Numpy.save()/Numpy.load()



Using SciPy to Solve a Linear Algebra Problem

Problem Statement:

There is a test with 30 questions worth 150 marks. The test has two types of questions:

1. True or false – carries 4 marks each
2. Multiple choice – carries 9 marks each

Find the number of true or false and multiple -choice questions.

Common instructions:

- If you are new to Python, download the “Anaconda Installation Instructions” document from the “Resources” tab to view the steps for installing Anaconda and the Jupyter notebook.
- Download the “Assignment 01” notebook and upload it on the Jupyter notebook to access it.
- Follow the cues provided to complete the assignment.

UNASSISTED PRACTICE

Using SciPy to Declare Random Values

Problem Statement:

Use SciPy to declare 20 random values for random values and perform the following:

1. CDF– Cumulative Distribution Function for 10 random variables.
2. PDF– Probability Density Function for 14 random variables.

Common instructions:

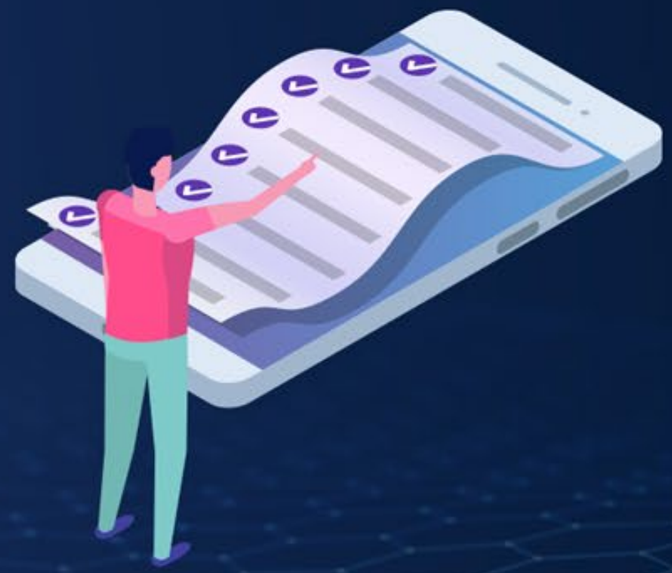
- If you are new to Python, download the “Anaconda Installation Instructions” document from the “Resources” tab to view the steps for installing Anaconda and the Jupyter notebook.
- Download the “Assignment 02” notebook and upload it on the Jupyter notebook to access it.
- Follow the cues provided to complete the assignment.

UNASSISTED PRACTICE

Key Takeaways

You are now able to:

- Explain the importance of SciPy
- List the characteristics of SciPy
- Explain sub-packages of SciPy
- Discuss SciPy sub-packages, such as optimization, integration, linear algebra, statistics, weave, and IO



DATA AND
ARTIFICIAL INTELLIGENCE

Knowledge Check

Knowledge Check

1

What are the specification limits provided for curve fitting function (optimize.curve.fit), during the optimization process?

a. Upper limit value

b. Lower limit value

c. Upper and lower limit values

d. Only the optimization method

Knowledge Check

1

What are the specification limits provided for curve fitting function (optimize.curve.fit), during the optimization process?

a. Upper limit value

b. Lower limit value

c. Upper and lower limit values

d. Only the optimization method

The correct answer is **C**

Both the upper and lower limit values should be specified for optimize.curve.fit function.

Knowledge Check

2

Which of the following function is used for inversing the matrix?

a. SciPy.special

b. SciPy.linalg

c. SciPy.signal

d. SciPy.stats

Knowledge Check

2

Which of the following function is used for inversing the matrix?

a. SciPy.special

b. SciPy.linalg

c. SciPy.signal

d. SciPy.stats

The correct answer is **b**

SciPy.linalg is used to inverse the matrix.

Knowledge Check

3

Which of the following is performed using SciPy?

a. Website

b. Plot data

c. Scientific calculations

d. System administration

Knowledge Check 3

Which of the following is performed using SciPy?

a. Website

b. Plot data

c. Scientific calculations

d. System administration

The correct answer is **c**

SciPy has been specially made to perform scientific calculations. Generally, Python is the programming language that has libraries to perform all listed activities.

Knowledge Check

4

Which of the following functions is used to calculate minima?

a. optimize.minimize()

b. integrate.quad()

c. stats.linregress()

d. linalg.solve()

Knowledge Check

4

Which of the following functions is used to calculate minima?

a. optimize.minimize()

b. integrate.quad()

c. stats.linregress()

d. linalg.solve()

The correct answer is **a**

The function optimize.minimize() is used to calculate minima. integrate.quad () is used for integral calculation, stats.linregress() is used for linear regression, and linalg.solve() is used to solve a linear system.

Knowledge Check

5

Which of the following syntaxes is used to generate 100 random variables from a t-distribution with df = 10?

a. stats.t.pmf(df=10, size=100)

b. stats.t.pdf(df=10, size=100)

c. stats.t.rvs(df=10, size=100)

d. stats.t.rand(df=10, size=100)

Knowledge Check

5

Which of the following syntaxes is used to generate 100 random variables from a t-distribution with df = 10?

a. stats.t.pmf(df=10, size=100)

b. stats.t.pdf(df=10, size=100)

c. stats.t.rvs(df=10, size=100)

d. stats.t.rand(df=10, size=100)

The correct answer is **c**

The stats.t.rvs() function is used to generate random variables. stats.t.pmf() function is used to generate the probability of mass function, and stats.t.pdf() is used to generate probability density function. Note that stats.t.rand () does not exist.

Knowledge Check

6

Which of the following functions is used to run C or C++ codes in SciPy?

a. io.loadmat()

b. weave.inline()

c. weave.blitz()

d. io.whosmat()

Knowledge Check

6

Which of the following functions is used to run C or C++ codes in SciPy?

a. io.loadmat()

b. weave.inline()

c. weave.blitz()

d. io.whosmat()

The correct answer is **b**

inline() function accepts C codes as string and compiles them for later use. loadmat() loads variables from .mat file. whosmat() checks the variables inside a .mat file.blitz(), and then compiles NumPy expressions for faster running, but it can't accept C codes.

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You

simplilearn