SciPy Cheatsheet

SciPy is a popular open-source Python library for scientific computing. It provides a wide range of tools for working with scientific data, including optimization, integration, interpolation, signal processing, and more. This cheatsheet provides a quick reference for some of SciPy's unique features, including code blocks for optimization, integration, interpolation, signal processing, and more. Additionally, it includes a list of resources for further learning.

Optimization

```python
import numpy as np
from scipy.optimize import minimize

# Define an objective function
def rosen(x):
    return sum(100.0 * (x[1:] - x[:-1] ** 2.0) ** 2.0 + (1 - x[:-1]) ** 2.0)

# Minimize the objective function
res = minimize(rosen, np.zeros(10), method='nelder-mead', options={'xtol': 1e-8,
'disp': True})
```

Integration

```python
from scipy.integrate import quad

# Define an integrand
def f(x):
    return np.exp(-x ** 2)

# Integrate the function from 0 to infinity
quad(f, 0, np.inf)
```

Interpolation

```python
from scipy.interpolate import interp1d

# Define some data
x = np.linspace(0, 10, num=11, endpoint=True)
y = np.cos(-x ** 2 / 9.0)

# Interpolate the data
f = interp1d(x, y, kind='cubic')
xnew = np.linspace(0, 10, num=41, endpoint=True)
ynew = f(xnew)
```

Signal Processing

```
from scipy import signal

# Define a signal
t = np.linspace(0, 1, 1000, endpoint=False)
x = signal.square(2 * np.pi * 5 * t)

# Apply a lowpass filter to the signal
b, a = signal.butter(4, 0.1, 'lowpass')
y = signal.filtfilt(b, a, x)
```

Other Useful Features

```
from scipy import stats

# Generate random samples from a normal distribution
x = stats.norm.rvs(size=1000)

# Compute the mean and standard deviation of the samples
stats.norm.fit(x)

# Compute the correlation between two variables
x = np.random.normal(size=100)
y = np.random.normal(size=100)
stats.pearsonr(x, y)
```

Resources

- SciPy documentation
- SciPy tutorial
- Python Data Science Handbook