SciPy in Python

What is Scipy?

SciPy stands for **Scientific Python.**

Functions used often in NumPy and data science have been improved and introduced to SciPy.

SciPy Constants:

When dealing with data science, you may find these constants useful.

```
from scipy import constants

print(constants.pi)
```

Constant Units:

The **dir()** function can be used to view a list of all the units in the constants module.

```
from scipy import constants

print(dir(constants))
```

SciPy Graphs:

Floyd Warshall:

Find the shortest path between each pair of components by using the floyd warshall() technique.

Discover the shortest route between each pair of items.

```
import numpy as np
from scipy.sparse.csgraph import floyd_warshall
from scipy.sparse import csr_matrix

arr = np.array([
  [0, 1, 2],
  [1, 0, 0],
  [2, 0, 0]
])

newarr = csr_matrix(arr)

print(floyd_warshall(newarr, return_predecessors=True))
```

Depth First Order:

A depth first traversal from a node is returned by the depth first order() method.

```
import numpy as np
from scipy.sparse.csgraph import depth_first_order
from scipy.sparse import csr_matrix

arr = np.array([
  [0, 1, 0, 1],
  [1, 1, 1, 1],
  [2, 1, 1, 0],
  [0, 1, 0, 1]
])

newarr = csr_matrix(arr)

print(depth_first_order(newarr, 1))
```

Breadth First Order:

The breadth_first_order() method returns a breadth first traversal from a node.

```
import numpy as np
from scipy.sparse.csgraph import breadth_first_order
from scipy.sparse import csr_matrix

arr = np.array([
  [0, 1, 0, 1],
  [1, 1, 1, 1],
  [2, 1, 1, 0],
  [0, 1, 0, 1]
])

newarr = csr_matrix(arr)

print(breadth_first_order(newarr, 1))
```

Connected Components:

Find all of the connected components with the connected_components() method.

```python
import numpy as np
from scipy.sparse.csgraph import connected_components
from scipy.sparse import csr_matrix

arr = np.array([
  [0, 1, 2],
  [1, 0, 0],
  [2, 0, 0]
])

newarr = csr_matrix(arr)

print(connected_components(newarr))
```

Bellman Ford:

The bellman_ford can find the shortest path and can handle negative weights as well.

```python
import numpy as np
from scipy.sparse.csgraph import bellman_ford
from scipy.sparse import csr_matrix

arr = np.array([
  [0, -1, 2],
  [1, 0, 0],
  [2, 0, 0]
])

newarr = csr_matrix(arr)

print(bellman_ford(newarr, return_predecessors=True, indices=0))
```

SciPy Matlab Arrays:

Exporting Data in Matlab:

The savemat() function allows us to export data in Matlab format.

```
from scipy import io
import numpy as np

arr = np.arange(10)

io.savemat('arr.mat', {"vec": arr})
```

Import Data from Matlab:

The loadmat() function allows us to import data from a Matlab file.

```
from scipy import io
import numpy as np

arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9,])

# Export:
io.savemat('arr.mat', {"vec": arr})

# Import:
mydata = io.loadmat('arr.mat')

print(mydata)
```

Output:

```
{
   '__header__': b'MATLAB 5.0 MAT-file Platform: nt, Created on: Tue
Sep 22 13:12:32 2020',
   '__version__': '1.0',
   '__globals__': [],
   'vec': array([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]])
}
```

To display simply the array from the matlab data, use the variable name "vec":

```
...

print(mydata['vec'])
```

Output: [[0 1 2 3 4 5 6 7 8 9]]

```
# Import:
mydata = lo.loadmat('arr.mat', squeeze_me=True)

print(mydata['vec'])
```

Output: [0 1 2 3 4 5 6 7 8 9]

Did you find it useful ?

Let us know in the comments

FOLLOW FOR MORE