Practical No 12

Roll No 2049

❖ Constant in Scipy

1) Print the constant value of PI

```
from scipy import constants
print(constants.pi)
```

Output:

```
3.141592653589793
```

2) List all constants

```
from scipy import constants
print(dir(constants))
```

Output:

```
['Avogadro', 'Boltzmann', 'Btu', 'Btu_IT',
'Btu_th', 'C2F', 'C2K', 'ConstantWarning',
```

3) Example of Metric Unit

```
from scipy import constants
print(constants.yotta)
print(constants.zetta)
print(constants.exa)
print(constants.peta)
print(constants.tera)
print(constants.giga)
print(constants.mega)
print(constants.kilo)
print(constants.hecto)
print(constants.deka)
print(constants.deci)
print(constants.centi)
print(constants.milli)
print(constants.micro)
print(constants.nano)
print(constants.pico)
print(constants.femto)
print(constants.atto)
print(constants.zepto)
```

Output:

```
1e+24
1e+21
1e+18
1000000000000000.0
1000000000000.0
1000000000.0
1000000.0
1000.0
100.0
10.0
0.1
0.01
0.001
1e-06
1e-09
1e-12
1e-15
1e-18
1e-21
```

4) Example of Binary, Mass & Angle constant

```
from scipy import constants
print(constants.kibi)
print(constants.mebi)
print(constants.gibi)
print(constants.tebi)
print(constants.pebi)
print(constants.exbi)
print(constants.zebi)
print(constants.yobi)

# Mass constants

from scipy import constants
print(constants.gram)
print(constants.metric_ton)
print(constants.grain)
print(constants.lb)
print(constants.pound)
print(constants.oz)
print(constants.ounce)
print(constants.stone)
print(constants.long_ton)
print(constants.short_ton)
print(constants.troy_ounce)
print(constants.troy_pound)
print(constants.carat)
print(constants.atomic_mass)
print(constants.m_u)
print(constants.u)

# Angle Constants

from scipy import constants
print(constants.degree)
print(constants.arcmin)
print(constants.arcminute)
print(constants.arcsec)
print(constants.arcsecond)
```

Output:

```
1024
1048576
1073741824
1099511627776
1125899906842624
1152921504606846976
1180591620717411303424
1208925819614629174706176
```

```
0.001
1000.0
6.479891e-05
0.45359236999999997
0.45359236999999997
0.028349523124999998
0.028349523124999998
6.3502931799999995
1016.0469088
907.1847399999999
0.031103476799999998
0.37324172159999996
0.0002
1.66053904e-27
1.66053904e-27
1.66053904e-27
```

```
0.017453292519943295
0.0002908882086657216
0.0002908882086657216
4.84813681109536e-06
4.84813681109536e-06
```

❖ Optimizer in Scipy

5) Find root of the equation `x + cos(x)`

```
from scipy.optimize import root

from math import cos

def eqn(x):

  return x + cos(x)

myroot = root(eqn, 0)

print(myroot.x)
```

Output:

```
[-0.73908513]
```

6) Minimize the function `x^2 + x + 2` with `BFGS`

```
from scipy.optimize import minimize

def eqn(x):

  return x**2 + x + 2

mymin = minimize(eqn, 0, method='BFGS')

print(mymin)
```

Output:

```
     fun: 1.75
hess_inv: array([[ 0.50000001]])
     jac: array([ 0.])
 message: 'Optimization terminated successfully.'
    nfev: 12
     nit: 2
    njev: 4
  status: 0
 success: True
       x: array([-0.50000001])
```

❖ Sparce Data in Scipy

7) Create a CSR matrix from an array

```
import numpy as np

from scipy.sparse import
csr_matrix

arr = np.array([0, 0, 2, 3, 0, 1, 1,
0, 2])

print(csr_matrix(arr))
```

Output:

```
  (0, 2)        2
  (0, 3)        3
  (0, 5)        1
  (0, 6)        1
  (0, 8)        2
```

8) Counting nonzeros with the `count_nonzero()` method

```
import numpy as np

from scipy.sparse import
csr_matrix

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

print(csr_matrix(arr).count_nonzer
o())
```

Output:

```
3
```

❖ Graph in Scipy

9) Find all of the connected components withthe `connected_components()` method

```
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, 1],

  [2, 0, 1]

])

newarr = csr_matrix(arr)

print(connected_components(newarr))
```

Output:

```
(1, array([0, 0, 0], dtype=int32))
```

❖ Spatial data in Scipy

10) Create a triangulation from following points

```
import sys

import matplotlib

matplotlib.use('Agg')

import numpy as np

from scipy.spatial import Delaunay

import matplotlib.pyplot as plt

points = np.array([

  [2, 4],

  [3, 4],

  [3, 0],

  [2, 2],

  [4, 1]

])

simplices = Delaunay(points).simplices

plt.triplot(points[:, 0], points[:, 1], simplices)

plt.scatter(points[:, 0], points[:, 1], color='r')
```

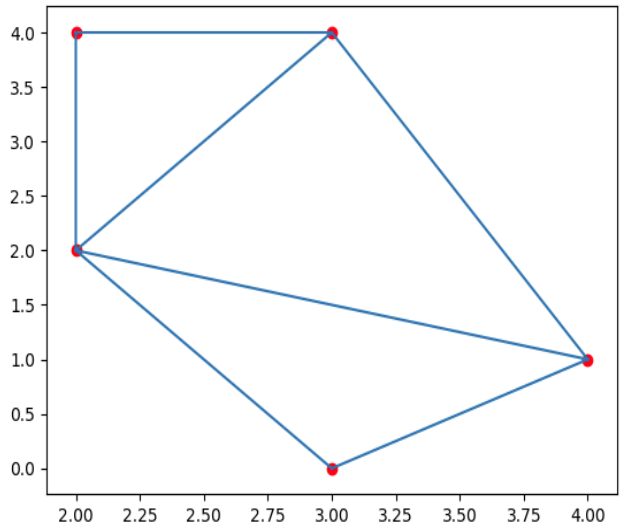
plt.show()

#Two lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

Output:



❖ SciPy Matlab Arrays

11) Import the array from following mat file

from scipy import io

import numpy as np

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

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

mydata = io.loadmat('arr.mat')

print(mydata)

Output:

```
{
  '__header__': b'MATLAB 5.0 MAT-file Platform: nt,
  '__version__': '1.0',
  '__globals__': [],
  'vec': array([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]])
}
```

❖ Interpolation in Scipy

For given xs and ys interpolate values from 2.1, 2.2... to 2.9

from scipy.interpolate import interp1d

import numpy as np

xs = np.arange(10)

ys = 2*xs + 1

interp_func = interp1d(xs, ys)

newarr = interp_func(np.arange(2.1, 3, 0.1))

print(newarr)

Output:

```
[ 5.2  5.4  5.6  5.8  6.   6.2  6.4  6.6  6.8]
```