Multinomial Softmax Function in Python

```
import math


def softmax(z):
  # this computes a list of exp(z_k) for each z_k in z.
  u = [math.exp(z_k) for z_k in z]

  # the sum of all elements in u.
  Z = sum(u)

  # compute the probabilities.
  p = [u_k / Z for u_k in u]

  return p


z = [0.6, 1.0, -1.5, 1.2, 10.0, -1.1]
p = softmax(z)
print('The softmax probabilities are:')
for e in p:
  print(f'{e:.4f}', end = ' ')
```

```
The softmax probabilities are:
0.0001 0.0001 0.0000 0.0002 0.9996 0.0000
```

Softmax with temperature T

```
def tsoftmax(z, T):
  # this computes a list of exp(z_k) for each z_k in z.
  u = [math.exp(z_k / T) for z_k in z]

  # the sum of all elements in u.
  Z = sum(u)

  # compute the probabilities.
  p = [u_k / Z for u_k in u]

  return p


z = [0.6, 1.0, -1.5, 1.2, 10.0, -1.1, -0.5, -0.1, 2.1, 3.1]
p1 = tsoftmax(z, 1)
for e in p:
  print(f'{e:.5f}', end = ' ')
```

```
0.00008 0.00012 0.00001 0.00015 0.99817 0.00002 0.00003 0.00004 0.00037 0.00101
```

```
p2 = tsoftmax(z, 2)
for e in p2:
  print(f'{e:.5f}', end = ' ')
```

```
0.00825 0.01008 0.00289 0.01114 0.90727 0.00353 0.00476 0.00581 0.01747 0.02880
```

```
p2 = tsoftmax(z, 10)
for e in p2:
  print(f'{e:.5f}', end = ' ')
```

```
0.08627 0.08980 0.06993 0.09161 0.22086 0.07279 0.07729 0.08044 0.10024 0.11078
```

```
p2 = tsoftmax(z, 1000000)
for e in p2:
  print(f'{e:.5f}', end = ' ')
```

```
0.10000 0.10000 0.10000 0.10000 0.10000 0.10000 0.10000 0.10000 0.10000 0.10000
```

```
p2 = tsoftmax(z, 1)
for e in p2:
```

```
  print(f'{e:.5f}', end = ' ')
```

```
0.00008 0.00012 0.00001 0.00015 0.99817 0.00002 0.00003 0.00004 0.00037 0.00101
```

```
p2 = tsoftmax(z, 0.5)
for e in p2:
  print(f'{e:.5f}', end = ' ')
```

```
0.00000 0.00000 0.00000 0.00000 1.00000 0.00000 0.00000 0.00000 0.00000 0.00000
```

The Ground Truth Matrix

We use the SciPy coo_matrix function.

```
import numpy as np
from scipy.sparse import coo_matrix

# N training examples.
N = 4

# The vector of N labels.
y = np.array([1, 1, 2, 0])

groundTruth = coo_matrix((np.ones(N, dtype = np.uint8), (y, np.arange(N)))).toarray()
```

```
groundTruth
```

```
array([[0, 0, 0, 1],
       [1, 1, 0, 0],
       [0, 0, 1, 0]], dtype=uint8)
```