follows Normal distribution log transformation
Reciprocal transformation
sqrt transformation
exponential transformation
box-cox transformation
yeo-jhanosn transformation

������ – 1
:

Read the packages

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
```

������ – 2
:

Read the data

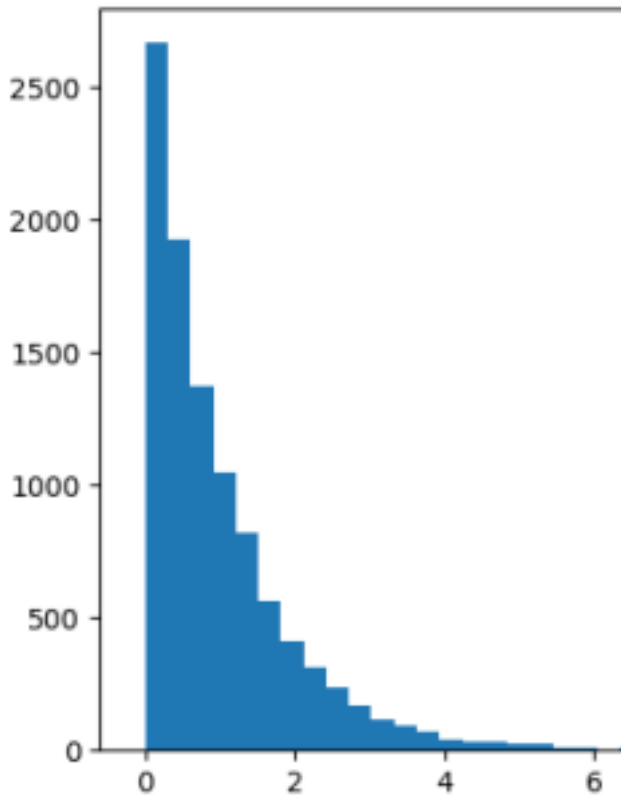
In [16]: In [19]:
```
data=np.random.exponential(size=10000)
# we are taking a random values from exponential distribution # 1000 samples we are taking
```

Converting skewed distribution to Normal distribution
All maths developed by assumption as data

```
data[:10]
```

Out[19]:
```
array([0.64916484, 0.56807349, 0.3212592 , 2.6500663 , 0.65805154,
       0.58082261, 1.26697857, 2.29447263, 0.86718717, 1.30088923])
```

In [18]:
```
plt.hist(data,bins=40,label='Exponential'
)
plt.legend()
plt.show()
```

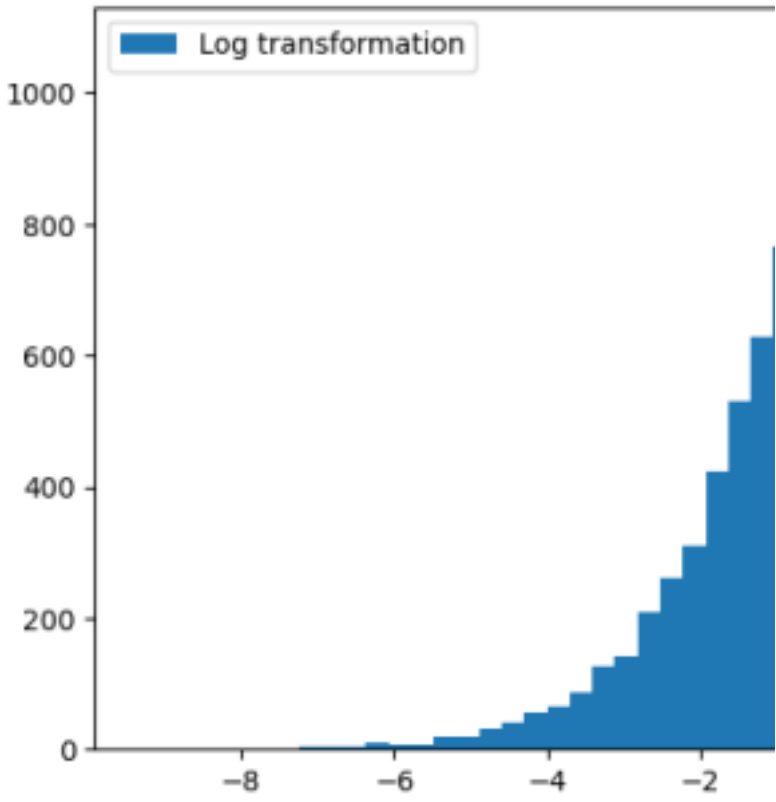


[illegible] [illegible] –
[illegible] 3 :
[illegible] :

Log transformation

np.log represents natural logorithm
natural logorithm means base e
exponential will multiply with log base e
Natural logorithms will works postive data
log transformation will remove the skew It will not convert into Normal distribution

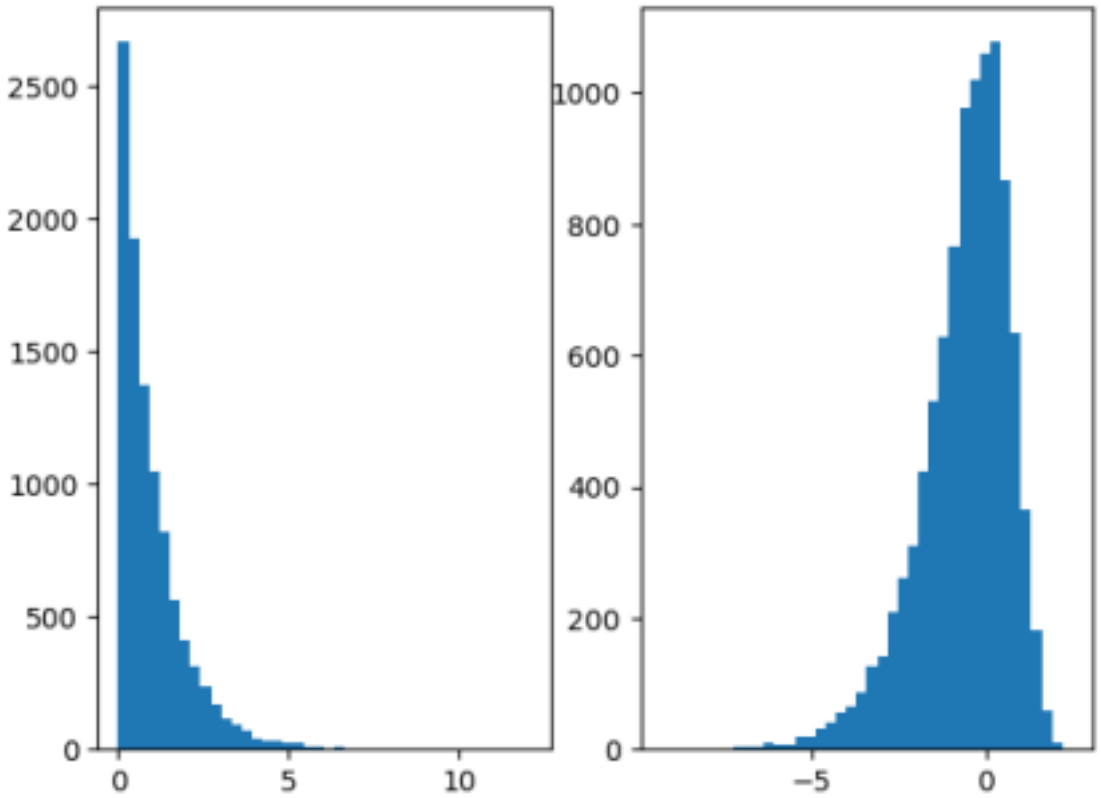
In [21]:
```
log_data=np.log(data)
plt.hist(log_data,bins=40,label='Log transformation')
plt.legend()
plt.show()
```



In [25]:
```
log_data
```

Out[25]:
```
array([-0.43206861, -0.56550448, -1.13550699, ..., -0.33326556,
       0.390394 , -1.01071406])
```

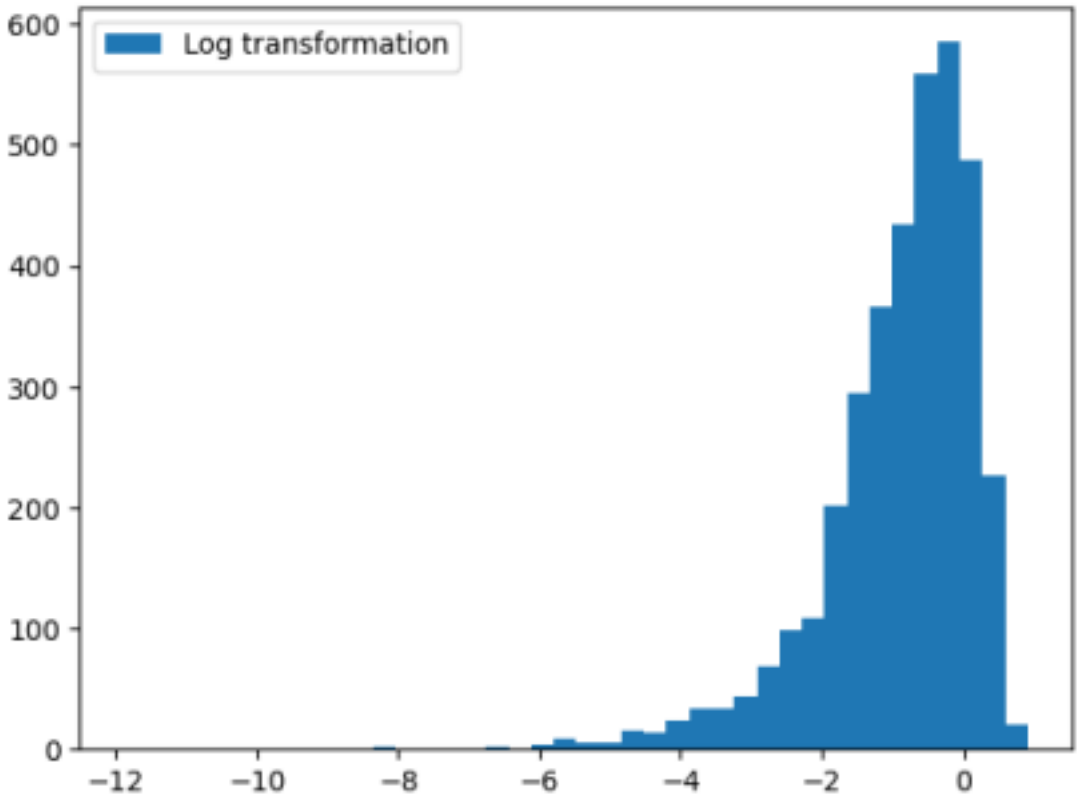
```
In [22]: plt.subplot(1,2,1).hist(data,bins=40)
plt.subplot(1,2,2).hist(log_data,bins=40)
plt.show()
```



In [24]: In [30]:

```
log_log_data=np.log(log_data)
plt.hist(log_log_data,bins=40,label='Log transformation') plt.legend()
plt.show()
```

```
C:\Users\omkar\AppData\Local\Temp\ipykernel_128148\4251579950.py:1: Runtim
eWarning: invalid value encountered in log
log_log_data=np.log(log_data)
```



– 4
:

Reciprocal Transformation

If data has zero , it will fail

```
print(data[:5])
print(1/data[:5])
```

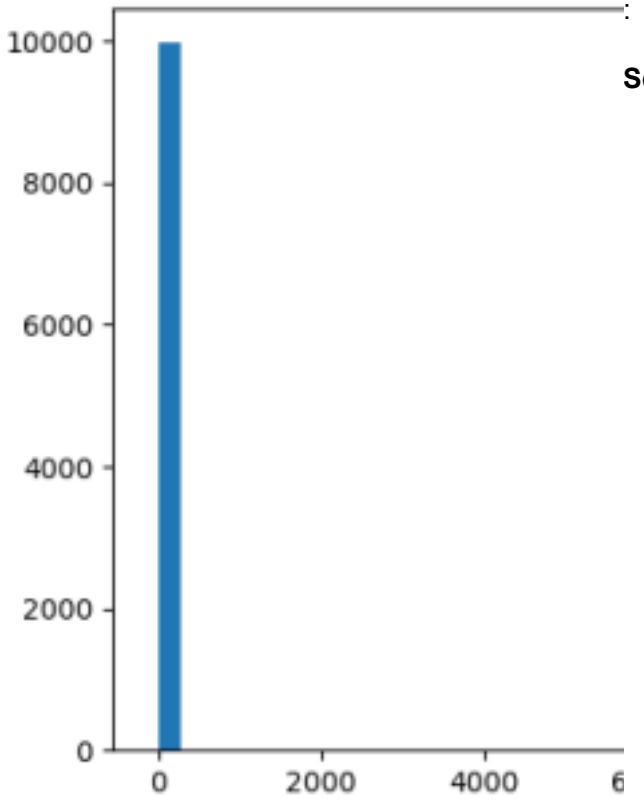
```
[0.64916484 0.56807349 0.3212592 2.6500663 0.65805154] [1.5404408 1.76033562
3.11275128 0.37734905 1.51963781]
```

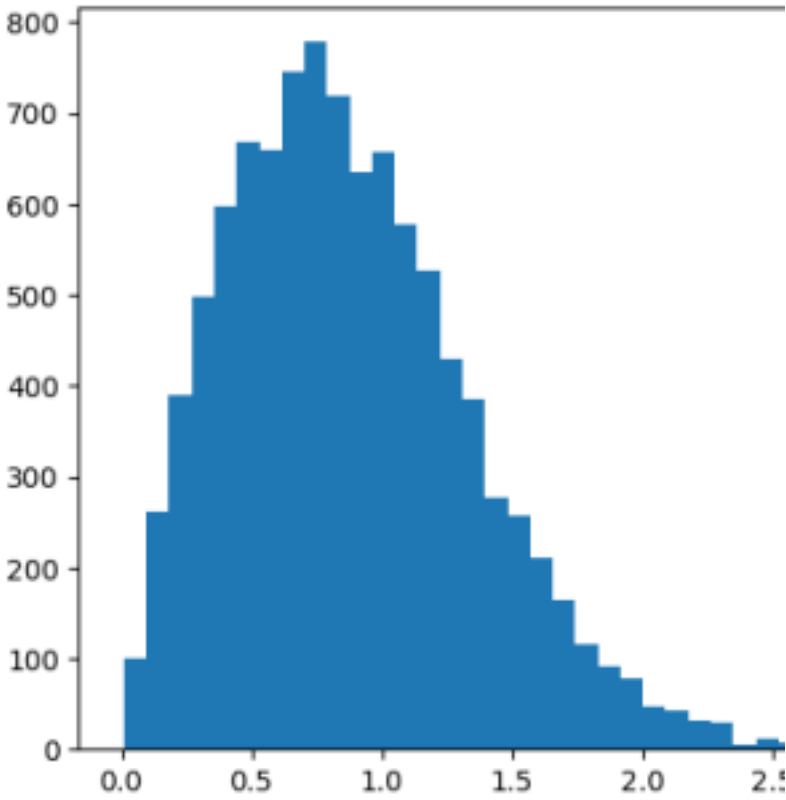

In [28]:

```
rec_data=1/data
plt.hist(rec_data,bins=40,label='Reciproc
al transformation') plt.legend()
plt.show()
```

In [31]:

Square root transformation






```
a=(10,20)
b=[(1,2),(6,7),(8,9)]
```

In [ ]:

```
# distance between two points
```

Power Transformer

it is in sklearn.preprocessing

method argument

box-cox

ye-jhonson

In [ ]: In [ ]:

```
sqrt_data=np.sqrt(data)
plt.hist(sqrt_data,bins=40,label='Square
root Transformation') plt.show()
```