Trng

July 20, 2025

1 Testing Generated Random Light Intensity Data

1.1 Importing The Libraries

[18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from trng import LightRandom
```

1.2 Generating True Random Numbers

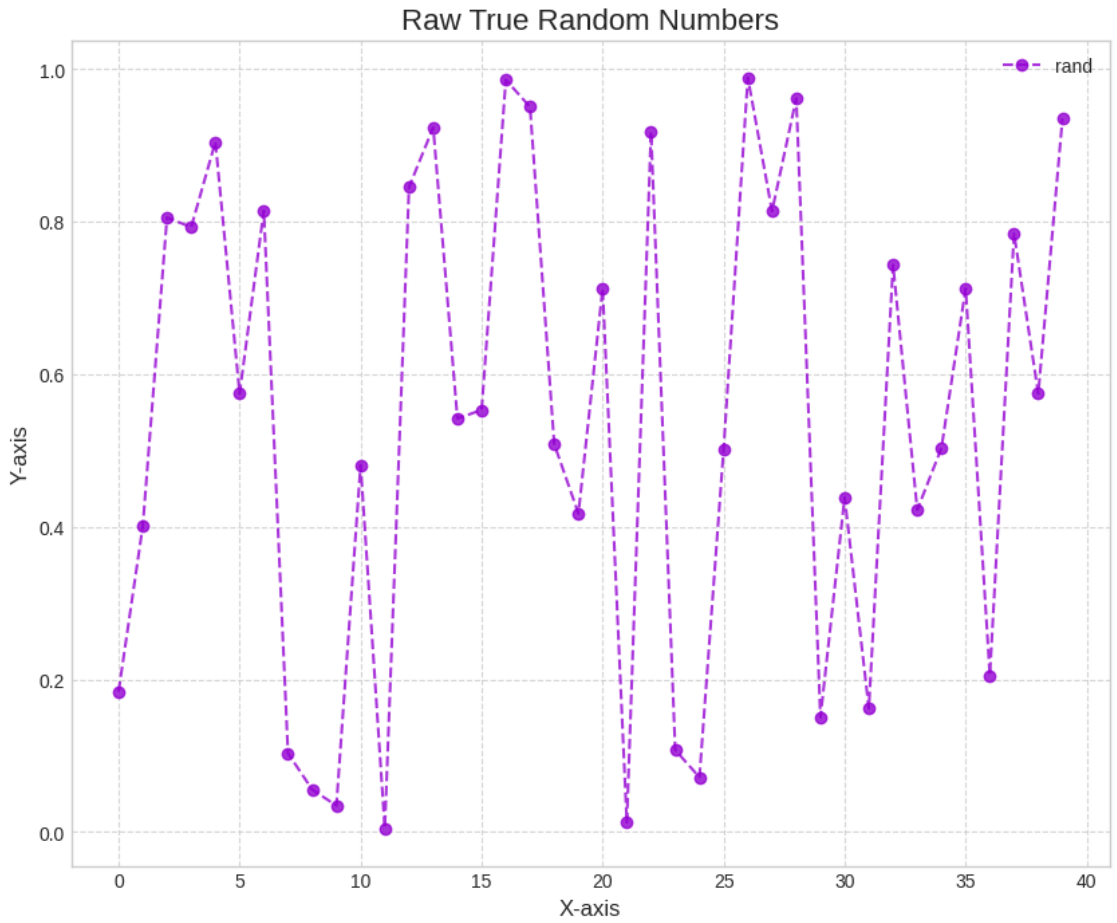
[19]:
```
n_samples = 40
```

[20]:
```
lrng = LightRandom()
data_rand = lrng.rand((n_samples))
data_randint = lrng.randint(low=-10, high=10, size=(n_samples))
data_randn = lrng.randn((n_samples))
DATA = pd.DataFrame(np.column_stack([data_rand, data_randint, data_randn]),␣
 ↪columns=["rand", "randint", "randn"])
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[20], line 2
      1 lrng = LightRandom()
----> 2 data_rand = lrng.rand((n_samples))
      3 data_randint = lrng.randint(low=-10, high=10, size=(n_samples))
      4 data_randn = lrng.randn((n_samples))

File ~/Python Codes/Fourier/TRNG-Pi5/test_notebooks/trng.py:51, in LightRandom.
 ↪rand(self, *shape)
     49 def rand(self, *shape):
     50     count = np.prod(shape) if shape else 1
---> 51     values = self._request_data(count)
     52     return values.astype(np.float64) / 0xFFFFFFFF

File ~/Python Codes/Fourier/TRNG-Pi5/test_notebooks/trng.py:17, in LightRandom.
 ↪_request_data(self, length)
     15 def _request_data(self, length):
```

```
     16     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
---> 17     client_socket.connect((self.host, self.port))
     18     client_socket.send(struct.pack('!I', length))  # Send 4-byte reques
     19     data = b""

KeyboardInterrupt:
```

[4]:
```
plt.style.use("seaborn-v0_8-whitegrid")
dpi=100
plt.figure(figsize=(10, 8), dpi=dpi)
plt.title("Raw True Random Numbers", fontsize=16)
plt.plot(np.array(range(len(data_rand))), data_rand, label="rand", marker="o",␣
 ↪linestyle="--", color="darkviolet", alpha=0.8)
plt.xlabel("X-axis", fontsize=12)
plt.ylabel("Y-axis", fontsize=12)
plt.grid(linestyle="--", alpha=0.8)
plt.legend()
plt.show()
```



[5]:
```
DATA.to_csv("Generated True Random Numbers.csv", index=False)
```

[6]:
```
DATA_np = pd.DataFrame({
    'rand': np.random.rand(n_samples),
    'randint': np.random.randint(-10, 10, n_samples),
    'randn': np.random.randn(n_samples)
})
DATA_np.to_csv("Generated Pseudo Random Numbers.csv", index=False)
```

1.3 Visualising True Random Numbers

[7]:
```
DATA = pd.read_csv("Generated True Random Numbers.csv")
DATA
```

[7]:

	rand	randint	randn
0	0.184071	-4.0	-1.099001
1	0.401181	1.0	-1.335428
2	0.805703	-1.0	1.740909
3	0.792947	7.0	-0.113003
4	0.904922	9.0	-0.525861
5	0.574647	-9.0	-0.261279
6	0.814738	-4.0	0.284464
7	0.103044	5.0	0.328243
8	0.055468	8.0	0.962014
9	0.034501	-5.0	-0.514300
10	0.481169	-2.0	-0.646812
11	0.004809	7.0	1.026157
12	0.845622	-7.0	-0.082526
13	0.923721	0.0	1.028029
14	0.542085	3.0	-0.420579
15	0.553256	1.0	-0.406830
16	0.986221	-3.0	-1.240920
17	0.950697	8.0	1.300961
18	0.508640	-5.0	1.144785
19	0.417450	3.0	-0.980659
20	0.712743	4.0	0.800247
21	0.012286	0.0	-1.908648
22	0.918486	-3.0	0.021659
23	0.107346	-7.0	-0.855209
24	0.071672	-2.0	-0.333097
25	0.502200	-8.0	-1.118087
26	0.989210	3.0	-0.033437
27	0.813914	5.0	0.190481
28	0.961957	-4.0	0.103846
29	0.150261	7.0	0.354436

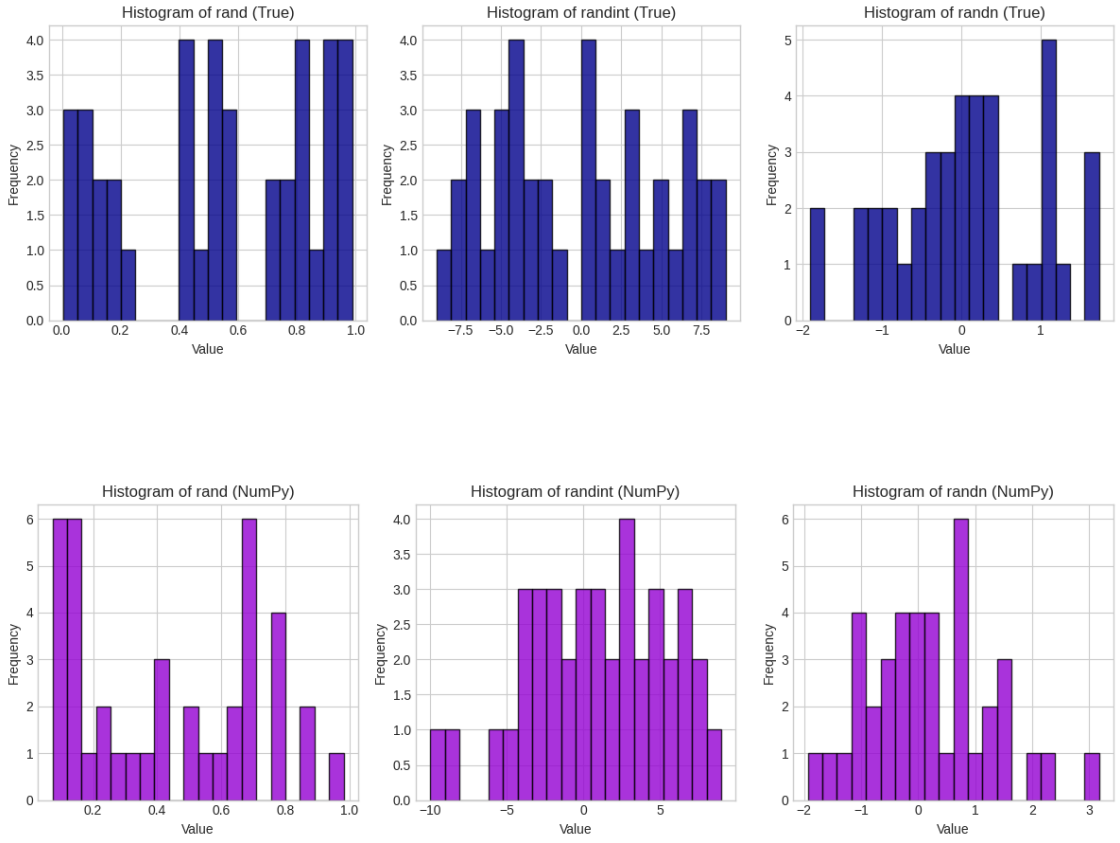
```
30  0.438780     -5.0 -1.817985
31  0.161948      0.0  0.271258
32  0.743658     -4.0  1.040433
33  0.422330     -6.0  1.577317
34  0.503418      0.0 -0.135739
35  0.712743      9.0  0.324374
36  0.205313     -8.0  0.226124
37  0.784739      6.0  1.030497
38  0.574647     -7.0  0.065303
39  0.935499      2.0  1.708411
```

[8]:
```
DATA.describe()
```

[8]:
```
            rand   randint      randn
count  40.000000  40.00000  40.000000
mean    0.540201  -0.15000   0.042514
std     0.325088   5.38064   0.929939
min     0.004809  -9.00000  -1.908648
25%     0.200003  -4.25000  -0.517190
50%     0.547671   0.00000   0.043481
75%     0.814120   4.25000   0.840689
max     0.989210   9.00000   1.740909
```

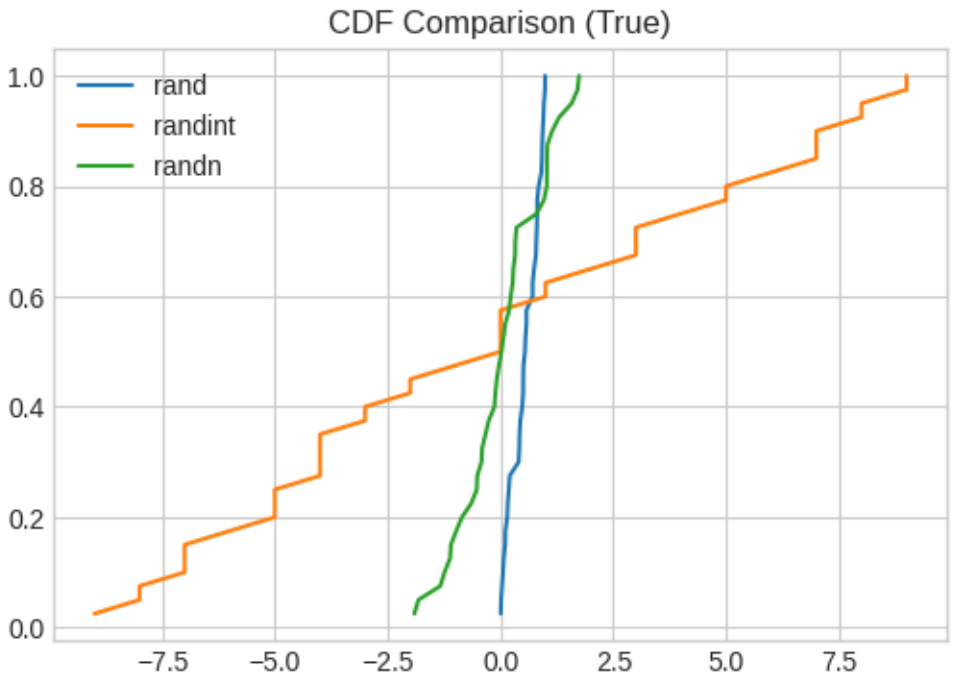
[9]:
```
plt.style.use("seaborn-v0_8-whitegrid")
dpi=100
plt.figure(figsize=(12, 4), dpi=dpi)

for i, col in enumerate(DATA.columns):
    plt.subplot(1, 3, i+1)
    plt.hist(DATA[col], bins=20, alpha=0.8, color="darkblue", edgecolor='black')
    plt.title(f"Histogram of {col} (True)")
    plt.xlabel("Value")
    plt.ylabel("Frequency")

plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 4), dpi=dpi)

for i, col in enumerate(DATA_np.columns):
    plt.subplot(1, 3, i+1)
    plt.hist(DATA_np[col], bins=20, alpha=0.8, color="darkviolet",␣
 ↪edgecolor='black')
    plt.title(f"Histogram of {col} (NumPy)")
    plt.xlabel("Value")
    plt.ylabel("Frequency")
```

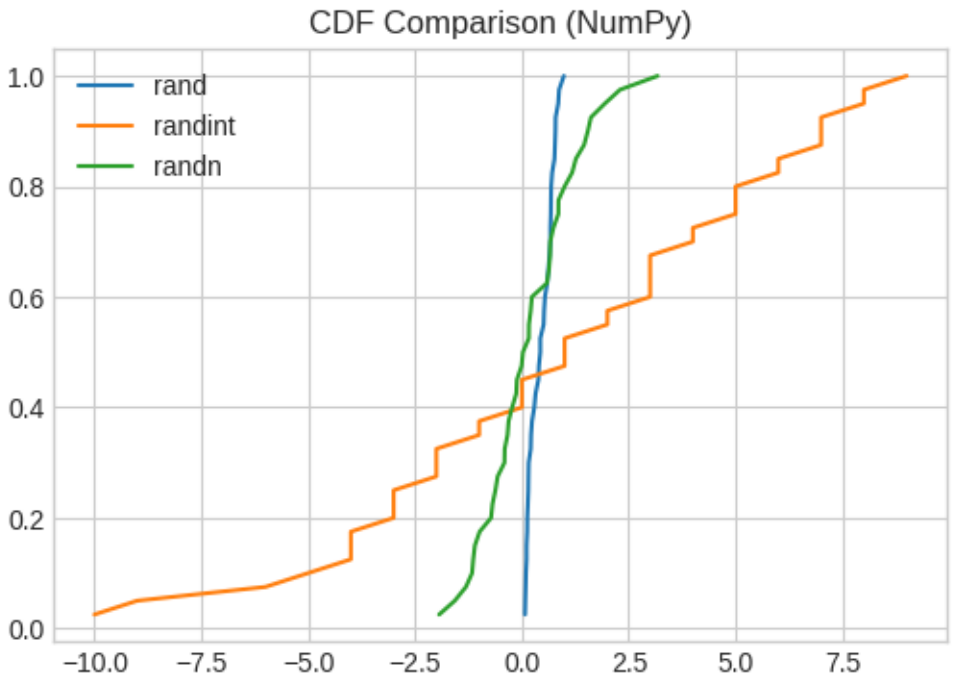
```
plt.tight_layout()
plt.show()
```



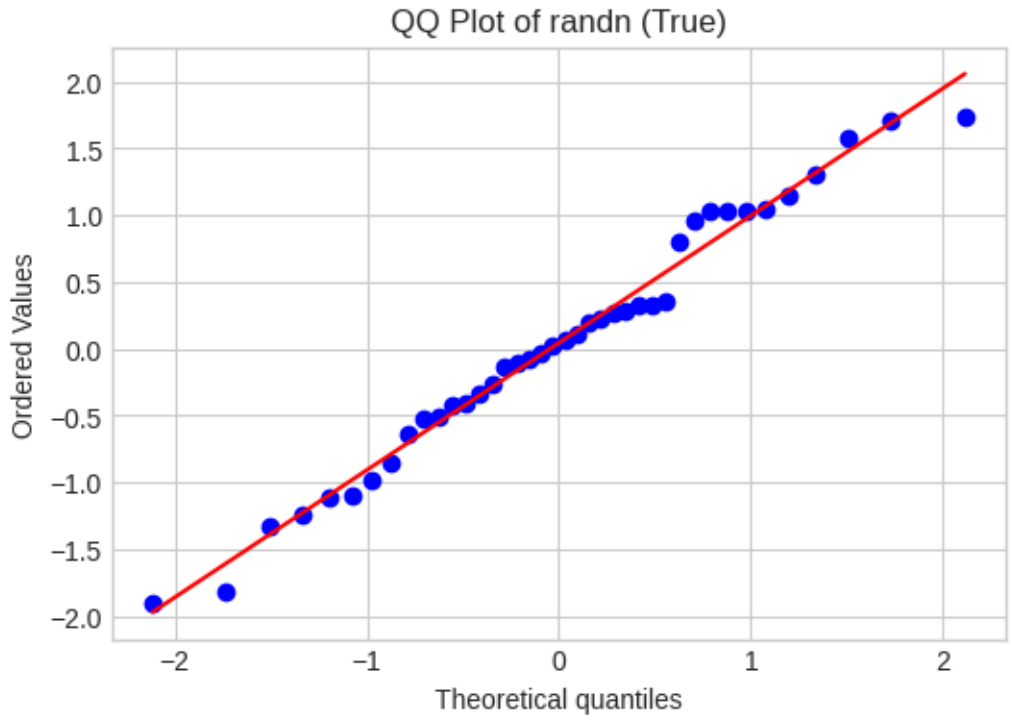
[10]:
```
def plot_cdf(data, label):
    sorted_data = np.sort(data)
    cdf = np.arange(1, len(sorted_data)+1) / len(sorted_data)
    plt.plot(sorted_data, cdf, label=label)

plt.figure(figsize=(6,4), dpi=dpi)
for col in DATA.columns:
    plot_cdf(DATA[col], col)
plt.title("CDF Comparison (True)")
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(6,4), dpi=dpi)
for col in DATA_np.columns:
    plot_cdf(DATA_np[col], col)
plt.title("CDF Comparison (NumPy)")
```

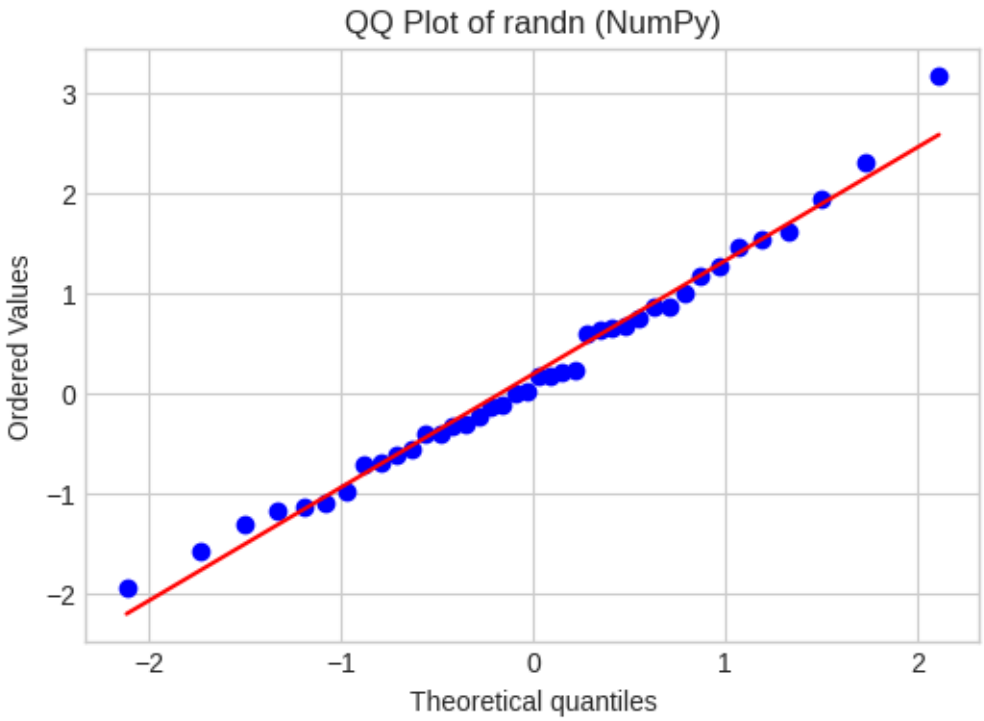
```
plt.legend()
plt.grid(True)
plt.show()
```



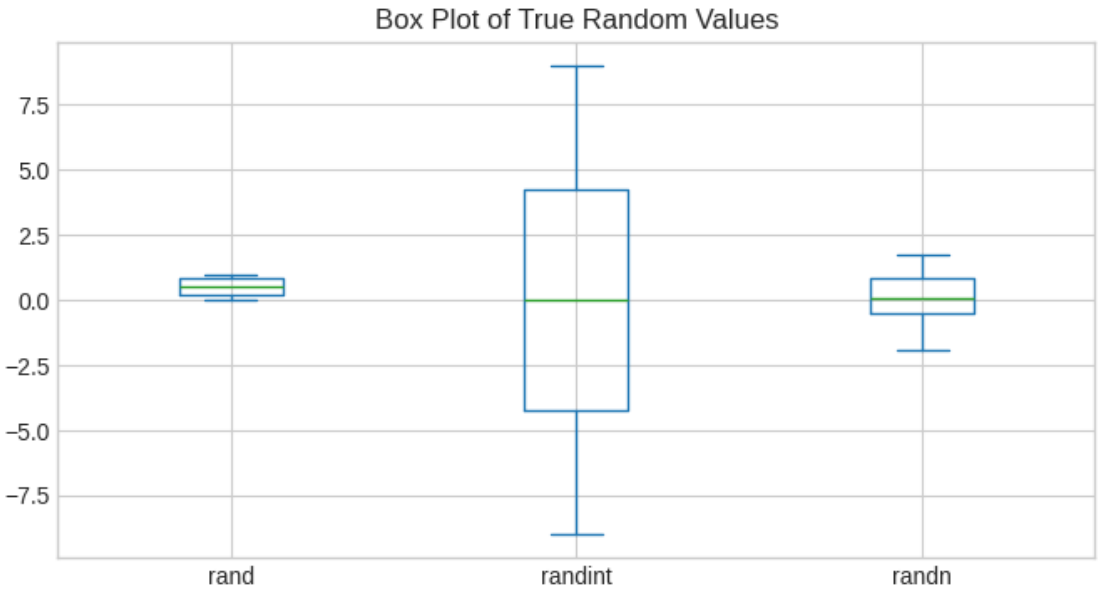


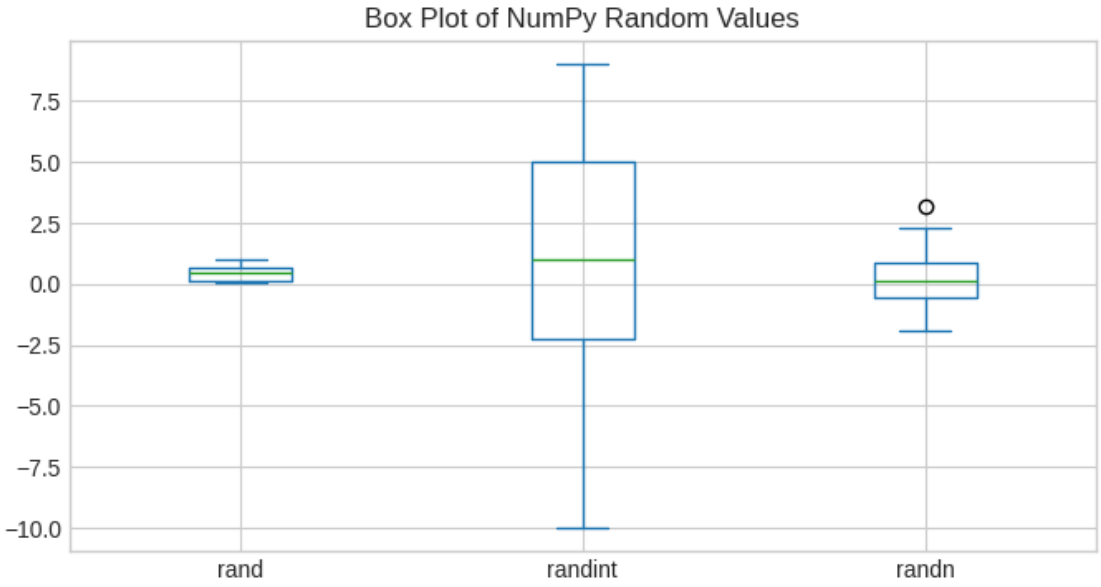
[11]:
```
import scipy.stats as stats

plt.figure(figsize=(6,4), dpi=dpi)
stats.probplot(DATA['randn'], dist="norm", plot=plt)
plt.title("QQ Plot of randn (True)")
plt.show()

plt.figure(figsize=(6,4), dpi=dpi)
stats.probplot(DATA_np['randn'], dist="norm", plot=plt)
plt.title("QQ Plot of randn (NumPy)")
plt.show()
```





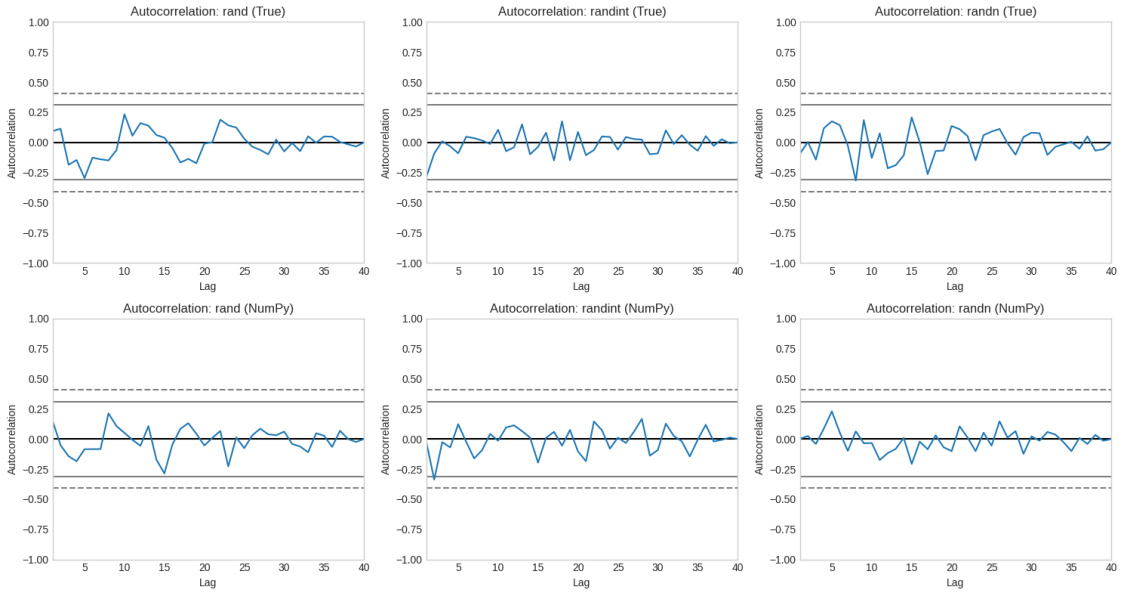
[12]:
```
DATA.plot(kind='box', figsize=(8, 4), title="Box Plot of True Random Values")
plt.grid(True)
plt.show()

DATA_np.plot(kind='box', figsize=(8, 4), title="Box Plot of NumPy Random␣
↪Values")
plt.grid(True)
plt.show()
```





[13]:
```
from pandas.plotting import autocorrelation_plot

cols = DATA.columns
n = len(cols)

fig, axs = plt.subplots(2, n, figsize=(5 * n, 8), dpi=dpi)
```

```
for i, col in enumerate(cols):
    plt.sca(axs[0, i])
    autocorrelation_plot(DATA[col])
    axs[0, i].set_title(f"Autocorrelation: {col} (True)")

    plt.sca(axs[1, i])
    autocorrelation_plot(DATA_np[col])
    axs[1, i].set_title(f"Autocorrelation: {col} (NumPy)")

plt.tight_layout()
plt.show()
```



[14]:
```
from scipy.stats import entropy

def calc_entropy(series, bins=20):
    counts, _ = np.histogram(series, bins=bins)
    probs = counts / counts.sum()
    return entropy(probs)

for col in DATA.columns:
    print(f"True {col} Entropy:", calc_entropy(DATA[col]))

def calc_entropy(series, bins=20):
    counts, _ = np.histogram(series, bins=bins)
    probs = counts / counts.sum()
    return entropy(probs)
```

```
for col in DATA_np.columns:
    print(f"NumPy {col} Entropy:", calc_entropy(DATA_np[col]))
```

```
True rand Entropy: 2.6099150724916775
True randint Entropy: 2.8394353820935447
True randn Entropy: 2.651336846633396
NumPy rand Entropy: 2.522932899087225
NumPy randint Entropy: 2.8001917713112037
NumPy randn Entropy: 2.631491227925561
```

[ ]:

[ ]: