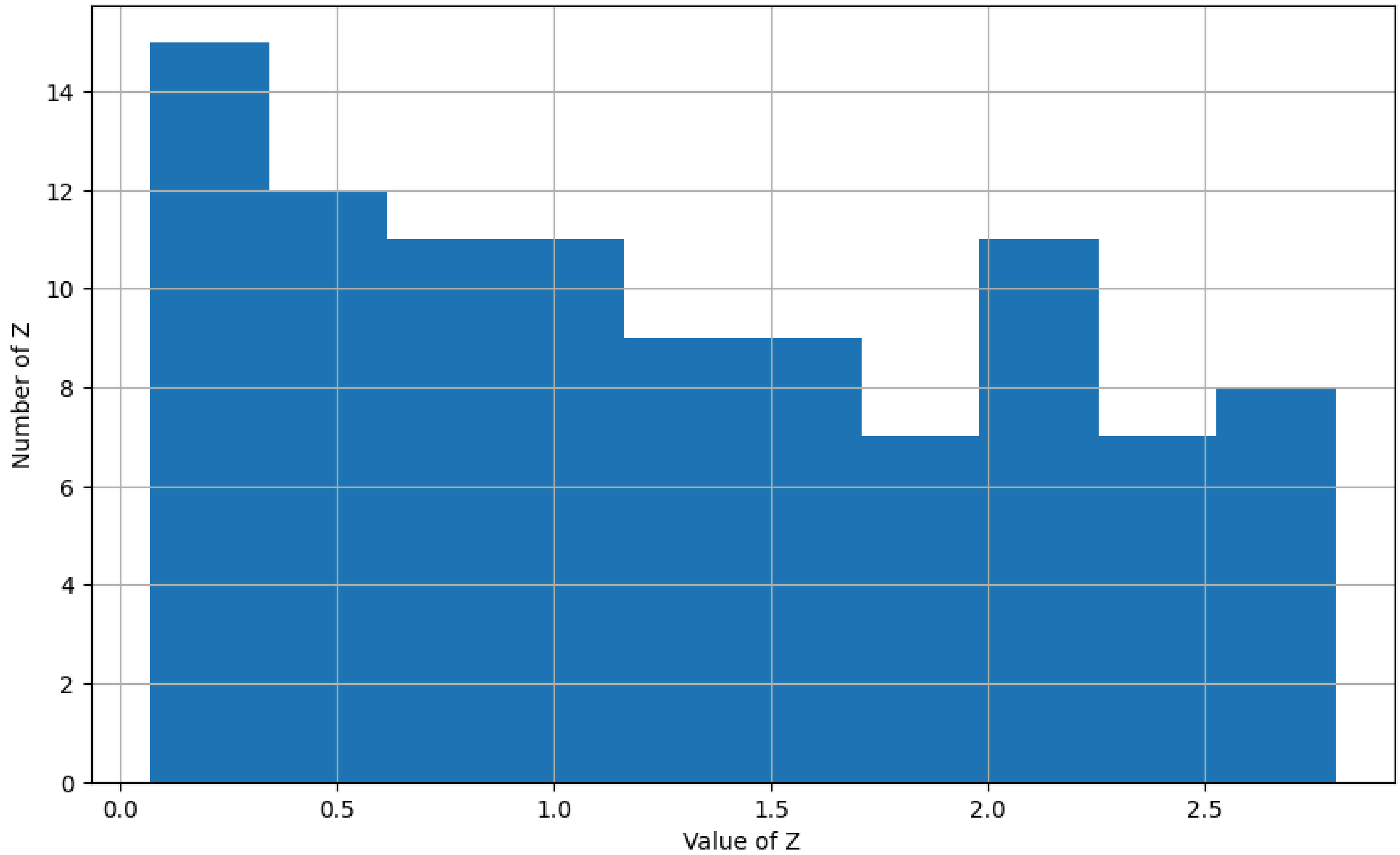
Colab link: https://colab.research.google.com/drive/1jRUXtVe11nNqQMTOk8Fnzl4CZO51xAzS?usp=sharing

a)

In [ ]:
```
import random
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(20)
def inverse_cdf(u):
    """Inverse CDF for the given distribution."""
    if u < 0.5:
        return 2 * u
    else:
        return 4 * u - 1
def create_sample(n):
    uniform = np.random.rand(n)
    samples = np.array([inverse_cdf(u) for u in uniform])
    return samples
Z_samples = create_sample(100)
plt.figure(figsize=(10, 6))
plt.hist(Z_samples)
plt.xlabel('Value of Z')
plt.ylabel('Number of Z')
plt.grid()
plt.show()
```



b)

In [ ]:
```
print(Z_samples)
```

```
[1.3525232  2.59085491 2.56612292 2.26334991 0.07177917 1.76703033
 0.75736188 1.07404378 1.63180586 0.38770044 0.5446328  1.87442373
 2.13201444 2.40131056 2.10097958 0.07332861 0.23338747 2.0051228
 0.47843643 0.50961203 2.43050212 2.7991161  1.24674743 0.35756104
 2.08100773 0.98476208 1.52501226 2.35799169 0.92207879 0.99588015
 1.71764447 1.60314366 0.53759048 0.13464933 2.08578055 0.96196826
 0.65841282 1.04256422 0.52725766 0.6210231  1.50741375 1.22979924
 0.63715912 0.78968644 0.51594917 1.32896449 0.32325742 1.39253528
 2.30329431 0.31278344 1.93720208 0.81728686 2.11475162 2.21588227
 2.14428577 1.36914807 1.65795681 1.58626915 0.85127296 1.05427334
 1.00503137 0.07416762 1.83246438 1.4817224  2.11123414 0.91881893
 0.75961111 0.58378441 1.22891545 0.1683272  1.52512668 2.77828195
 2.56495014 2.51433032 0.68950325 1.81240213 1.10935042 1.63726782
 0.53658127 1.43106832 0.09631474 1.78200256 0.59854376 0.33127111
 0.66450388 0.33087549 2.65855192 0.56805001 0.21521453 0.23400354
 2.0220944  2.35557783 0.07107405 0.24288754 2.56403222 1.06036894
 2.69767605 1.32497052 2.09302555 0.84535379]
```

In [ ]:
```
mean_Z = np.mean(Z_samples)
mean_Z
```

Out[ ]: 1.2801501814667406

f(z) = { 0; z<0 0.5; 0<=z<1 0.25; 1<=z<3 0; z>=3 } E[z] = integrate zf(z) dz E[z] = 1.25 the average of the samples = 1.28 is very close to the expected value = 1.25

c)

In [ ]:
```
#empirical variance
s2 = (Z_samples-mean_Z)**2
s2 = (np.sum(s2))/(100-1)
s2
```

Out[ ]:
```
0.6369537164449355
```

True Variance E[z^2] = integrate z^2f(z)dz E[z^2] = 2.3333 var[z] = E[z^2] - (E[z])^2 var[z] = 2.333-1.25^2 = 0.771

Therefore, the empirical Variance = 0.637 is very close to True variance of 0.771

d)

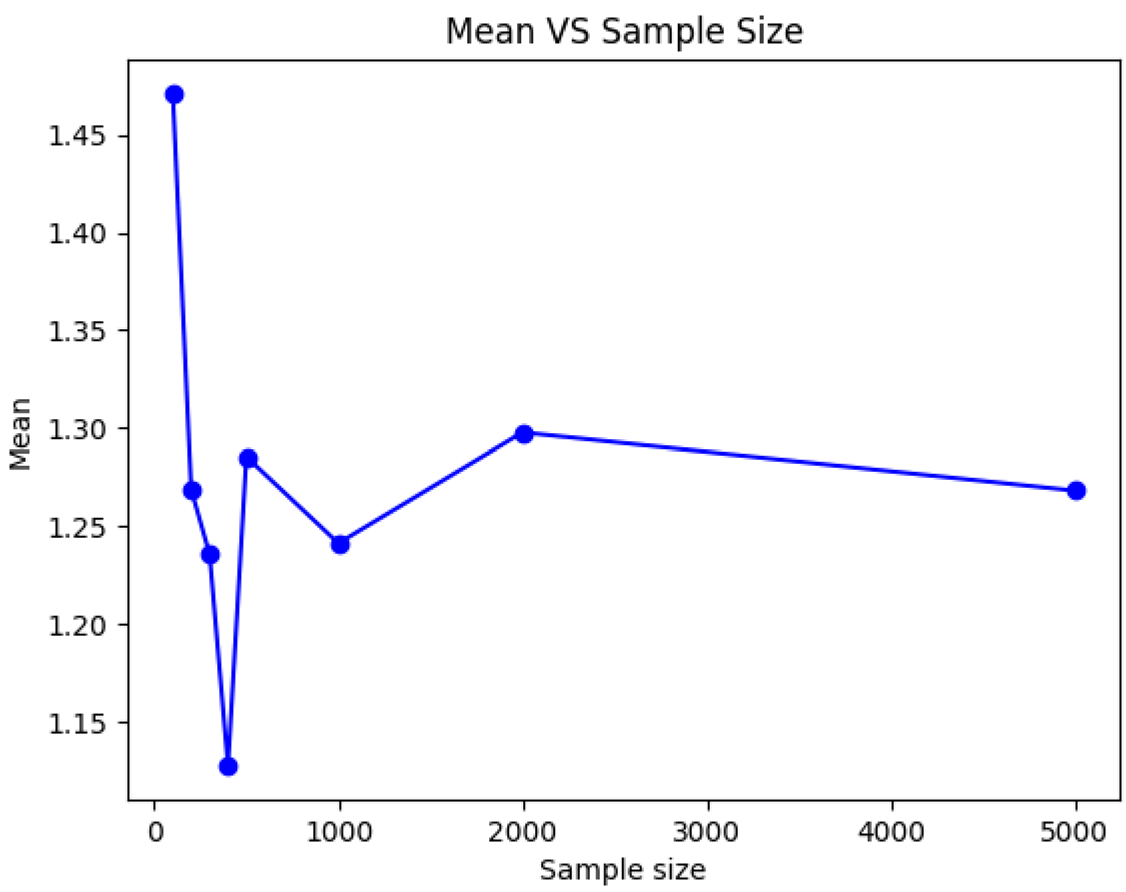
In [ ]:
```
def find_variance(Z, mean):
    s2 = (Z-mean)**2
    s2 = (np.sum(s2))/(len(Z)-1)
    return s2
```

In [ ]:
```
N = [100,200,300,400,500,1000,2000,5000]
mean_list = []
var_list = []
for i in N:
    Z = create_sample(i)
    mean_Z = np.mean(Z)
    variance_Z = find_variance(Z, mean_Z)
    mean_list.append(mean_Z)
    var_list.append(variance_Z)

plt.plot(N, mean_list, marker='o', linestyle='-', color='b')

# Add title and axis labels
plt.title('Mean VS Sample Size')
plt.xlabel('Sample size')
plt.ylabel('Mean')
```

Out[ ]:
```
Text(0, 0.5, 'Mean')
```



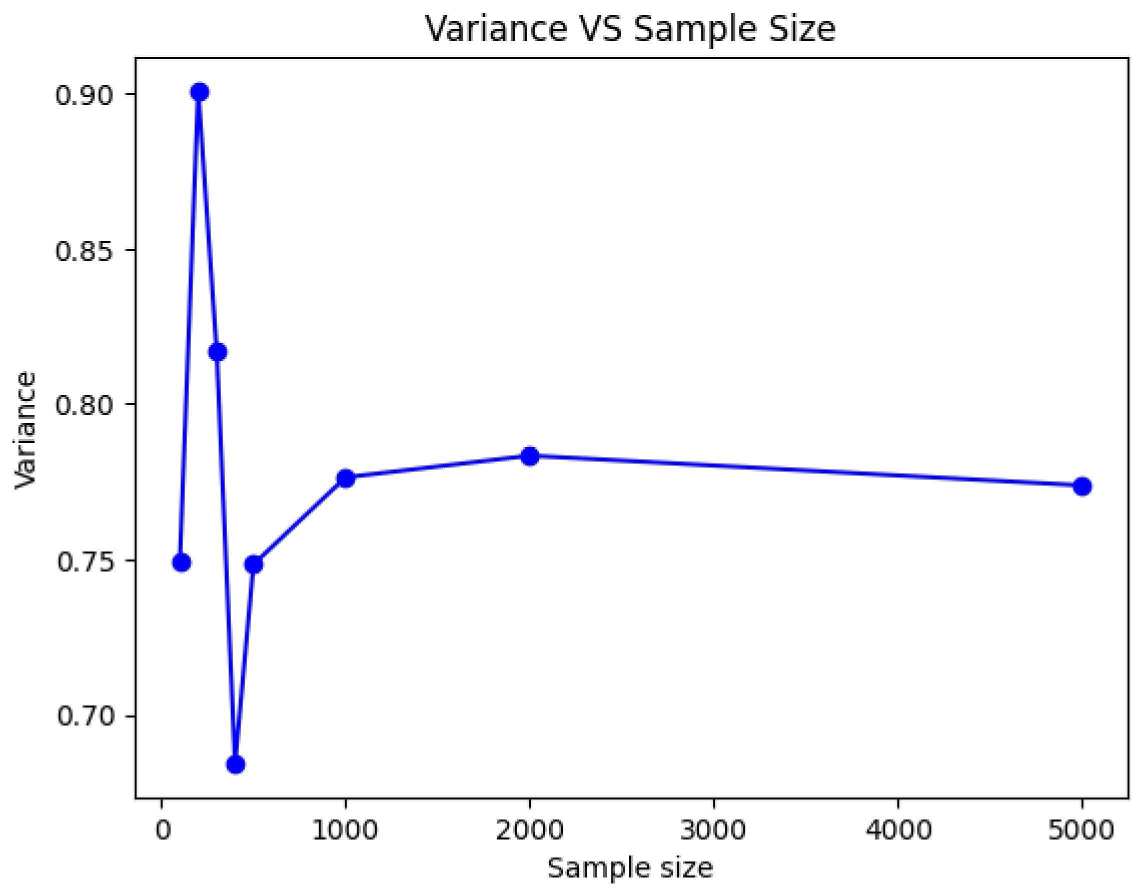
In [ ]:
```
print(mean_list)
```

```
[1.4712076045958156, 1.2687728552479312, 1.2356293593250294, 1.1276493098488232, 1.2854540998202464, 1.2412989523108637, 1.297804008547142, 1.268201416057972]
```

In [ ]:
```
plt.plot(N, var_list, marker='o', linestyle='-', color='b')

# Add title and axis labels
plt.title('Variance VS Sample Size')
plt.xlabel('Sample size')
plt.ylabel('Variance')
```

Out[ ]:
```
Text(0, 0.5, 'Variance')
```



In [ ]:
```
print(var_list)
```

```
[0.7490372496144646, 0.9010007293830482, 0.8173547296117948, 0.6842486706207207, 0.7484456943694631, 0.7764930611113341, 0.7835061793658915, 0.7739027962257388]
```

They converges: Mean -> 1.25 Variance -> 0.771

In [2]:
```
%%shell
jupyter nbconvert --to html /content/assignment1_Q1_2.ipynb
```

```
[NbConvertApp] Converting notebook /content/assignment1_Q1_2.ipynb to html
[NbConvertApp] WARNING | Alternative text is missing on 3 image(s).
[NbConvertApp] Writing 365147 bytes to /content/assignment1_Q1_2.html
```

Out[2]: