Practical-8

Aim :

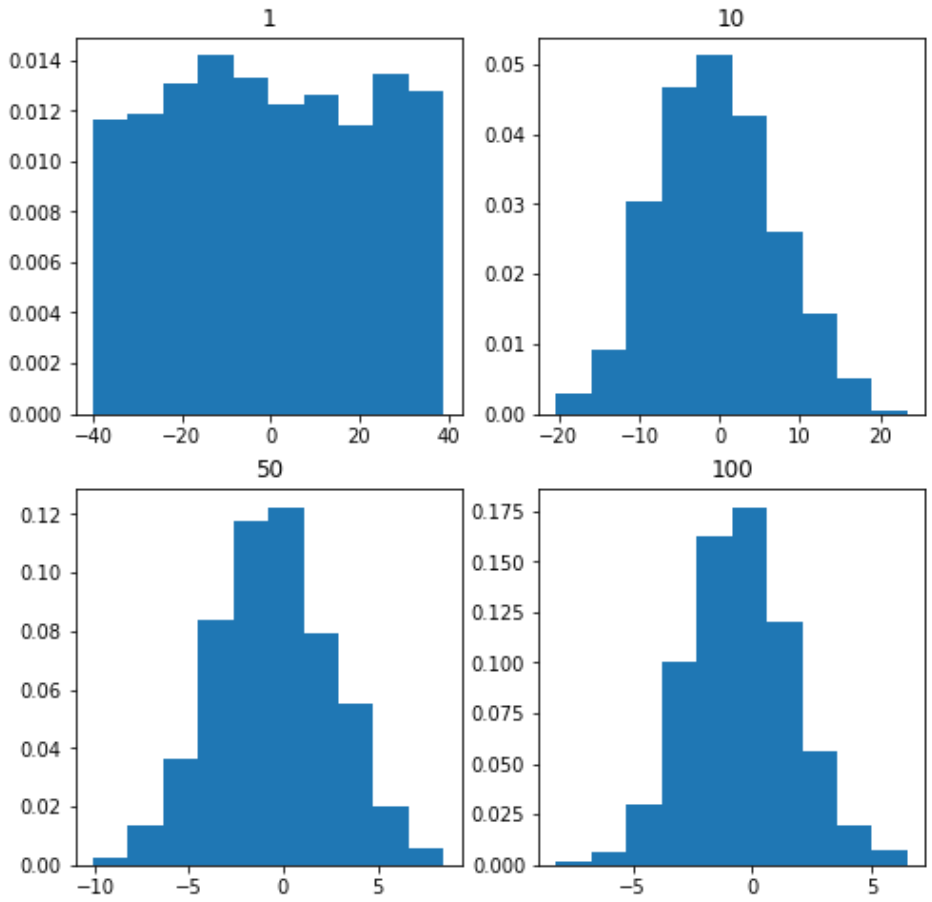
Write a python program to compute and explore normal distribution, central limit theorem, point estimate, interval estimation and hypothesis testing.

```
from scipy.stats import norm
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sbn

print("12002040701067")
a, b = 4.32, 3.18
rv = norm(a, b)
print ("RV : \n", rv)
quantile = np.arange (0.01, 1, 0.1)
# Random Variates
R = norm. rvs (a, b)
print ("Random Variates : \n", R)
# PDF
R = norm.pdf(a, b, quantile)
print ("\nProbability Distribution : \n", R)
```

```
12002040701067
RV :
 <scipy.stats._distn_infrastructure.rv_frozen object at 0x7f05a36d3950>
Random Variates :
 5.9127620974870405

Probability Distribution :
 [0.00000000e+00 1.72515030e-23 7.57695185e-07 1.48928058e-03
 2.03862091e-02 6.43213365e-02 1.14069546e-01 1.54822160e-01
 1.82936660e-01 2.00024445e-01]
```

```
#Python implementation of the Central Limit Theorem
# number of sample
num = [1, 10, 50, 100]
# list of sample means
means = []
# Generating 1, 10, 30, 100 random numbers from -40 to 40
# taking their mean and appending it to list means.
for j in num:
    # Generating seed so that we can get same result
    # every time the loop is run...
    np.random.seed(1)
    x = [np.mean(
        np.random.randint(
            -40, 40, j)) for _i in range(1000)]
    means.append(x)
k = 0
# plotting all the means in one figure
```

```
fig, ax = plt.subplots(2, 2, figsize =(8, 8))
for i in range(0, 2):
    for j in range(0, 2):
        # Histogram for each x stored in means
        ax[i, j].hist(means[k], 10, density = True)
        ax[i, j].set_title(label = num[k])
        k = k + 1
plt.show()
```




```
from pylab import *
import scipy.stats
N=1000.      # sample size
gamma=0.95  # confidence level
mu=10.      # true mean
sigma=2.    # true standard diviation
# x=randn(N)*sigma+mu  # surrogate data
mu_hat=mean(x)           # sample mean
sigma_hat=std(x, ddof=1) # sample standard deviation
print('sample mean mu_hat                  : %f' % mu_hat)
print('sample standard deviation sigma_hat : %f' % sigma_hat)
l=scipy.stats.t.ppf( (1-gamma)/2, N-1)    # lower percentile
u=scipy.stats.t.ppf( 1-(1-gamma)/2, N-1)  # upper percentile
print('confidence interval mu_hat          : (%f, %f)' %
      (mu_hat+l*sigma_hat/sqrt(N), mu_hat+u*sigma_hat/sqrt(N)))
l=scipy.stats.chi2.ppf( (1-gamma)/2, N-1)    # lower percentile
u=scipy.stats.chi2.ppf( 1-(1-gamma)/2, N-1)  # upper percentile
print('confidence interval sigma_hat       : (%f, %f)' %
      ( sqrt((N-1)/u)*sigma_hat, sqrt((N-1)/l)*sigma_hat))
```

```
sample mean mu_hat                  : -0.495240
sample standard deviation sigma_hat : 2.199148
```

```
confidence interval mu_hat          : (-0.631707, -0.358773)
confidence interval sigma_hat       : (2.106811, 2.300013)
```

```
#Hypothesis Testing
from scipy.stats import ttest_ind
data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [1.142, -0.432, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.075, -0.169]
stat, p = ttest_ind(data1, data2)
print('stat=%.3F, p-%.3F' % (stat, p))
if p > 0.05:
  print("Probably the same distribution")
else:
  print("Probably different distributions")
```

```
stat=-0.326, p-0.748
Probably the same distribution
```