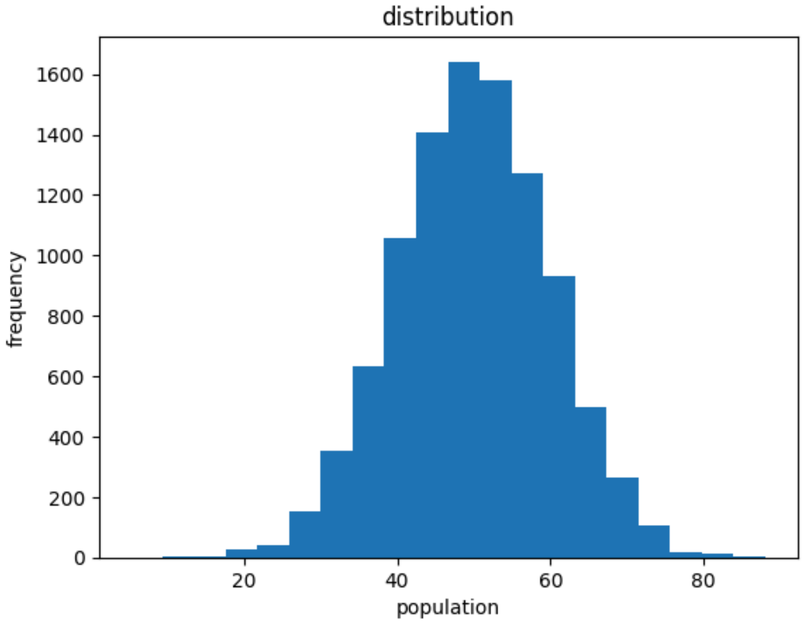
```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt


population = np.random.normal(loc = 50,scale = 10,size=10000)


mean_1 = np.mean(population)
std_1 = np.std(population)
print(f"the mean is{mean_1:.2f}")
print(f"the standard deviation is{std_1:.2f}")
```

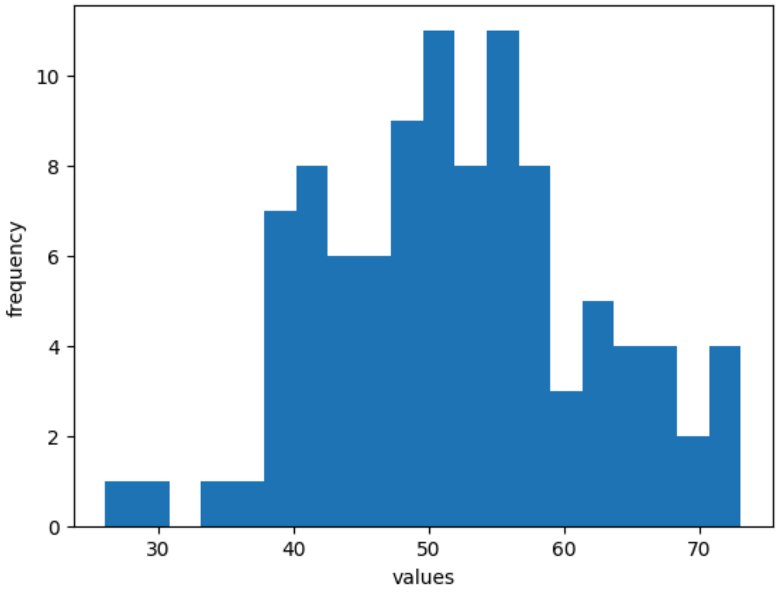
```
the mean is49.92
the standard deviation is9.95
```

```
plt.hist(population,bins = 20)
plt.xlabel("population")
plt.ylabel("frequency")
plt.title("distribution")
plt.show()
```



```
def uniform_sampling(population,sample_size):
  ind = np.random.choice(len(population), sample_size,replace = False)
  sample = [population[i] for i in ind]
  return sample


sample_size = 100
sample = uniform_sampling(population,sample_size)
print(sample)
plt.hist(sample,bins = 20)
plt.xlabel("values")
plt.ylabel("frequency")
plt.show()
```

```
[73.0774717744474, 56.56958946412946, 50.34973672554679, 42.54865735058151, 38.28070304435949, 56.13915597113421, 59.86
```



```
sample_mean = np.mean(sample)
sample_std = np.std(sample,ddof = 1)
print(f"the sample mean is{sample_mean:.2f}")
print(f"the sample standard deviation is{sample_std:.2f}")
```

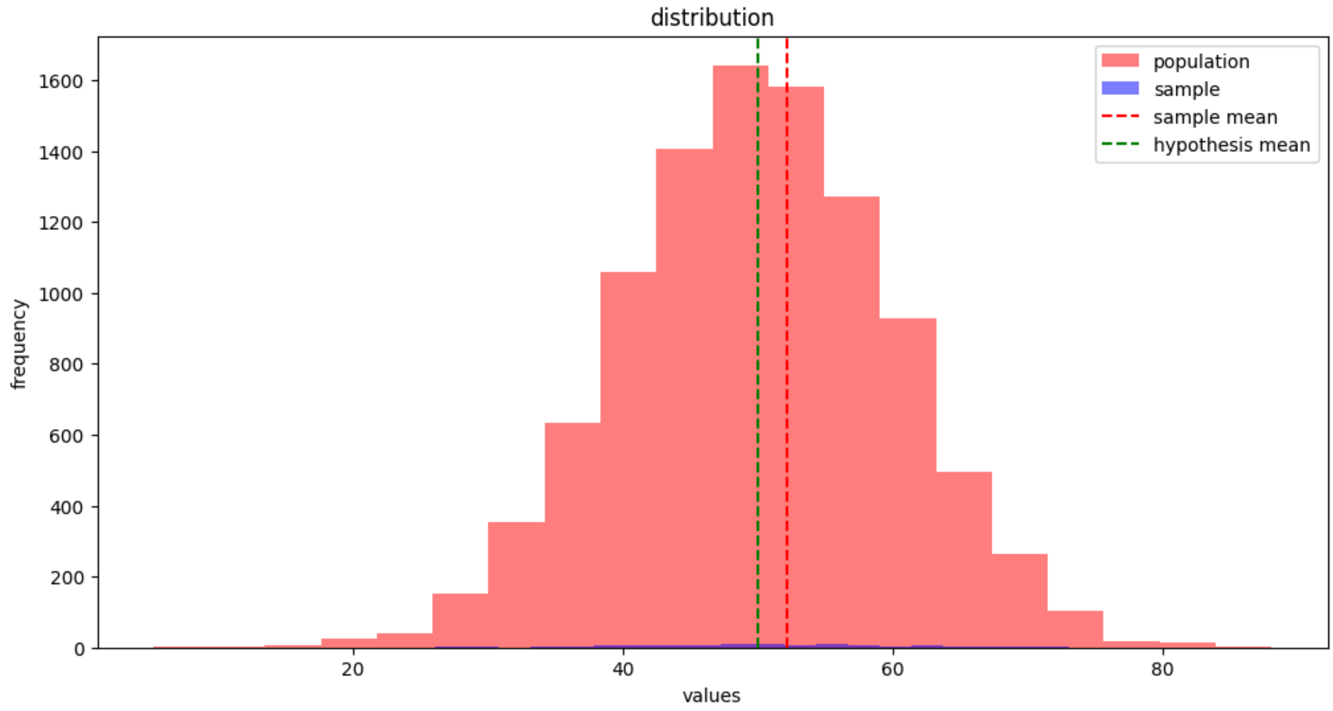
```
the sample mean is52.12
the sample standard deviation is9.77
```

```
confidence_level = 0.95
alpha = 1-confidence_level
t_critical = stats.t.ppf(1-alpha/2,df=sample_size-1)
margin_of_error = t_critical*(sample_std/np.sqrt(sample_size))
confidence_interval =  (sample_mean-margin_of_error,sample_mean+margin_of_error)


print(f"the sample mean is {sample_mean:.2f}")
print(f"the standard deviation is {sample_std:.2f}")
print(f"{confidence_level*100:.0f}% Confidence Interval for the mean:{confidence_interval}")
hypothesis_mean = 50
t_statistics,P_value = stats.ttest_1samp(sample,hypothesis_mean)
print(f"the t-statistics is{t_statistics:.2f}")
print(f"the P-value is{P_value:.2f}")
```

```
the sample mean is 52.12
the standard deviation is 9.77
95% Confidence Interval for the mean:(50.18328894770478, 54.06113864896432)
the t-statistics is2.17
the P-value is0.03
```

```
if P_value<alpha:
  print(f"Reject the null hypothesis: the sample differs from {hypothesis_mean}")
else:
  print(f"Fail to reject the null hypothesis: the sample mean does not significantly differ form {hypothesis_mean}")

plt.figure(figsize=(12,6))
plt.hist(population,bins = 20,alpha = 0.5,label = "population",color= 'red')
plt.hist(sample,bins = 20,alpha= 0.5,label = "sample",color = 'blue')
plt.axvline(x = sample_mean,color = 'red',linestyle = '--',label = "sample mean")
plt.axvline(x = hypothesis_mean,color = 'green',linestyle = '--',label = "hypothesis mean")
plt.xlabel("values")
plt.ylabel("frequency")
plt.title("distribution")
plt.legend()
plt.show()
```

```
Reject the null hypothesis: the sample differs from 50
```



Start coding or generate with AI.