Q1.

1. Randomly generate a 2*2 matrix and return the minimum value of x along the second axis.
2. Randomly generate a 2*5 matrix and calculate the difference between the maximum and the minimum of x along the second axis.
3. Randomly generate a 2*3 matrix and get the values and indices of the elements that are bigger than 2 in x.

In [1]:

```
import numpy as np
#Q1.1
x = np.random.rand(2,2)

min_val = np.min(x, axis=1)

print(min_val)

#Q1.2
x = np.random.rand(2,5)

max_val = np.max(x, axis=1)
min_val = np.min(x, axis=1)

diff = max_val - min_val

print(diff)

#Q1.3
x = np.random.randint(0, 5, (2,3))

idx = np.where(x > 2)

values = x[idx]

print(idx)
print(values)
```

```
[0.23948097 0.7395979 ]
[0.69872106 0.7117342 ]
(array([0, 0, 1, 1]), array([0, 2, 1, 2]))
[3 4 3 4]
```

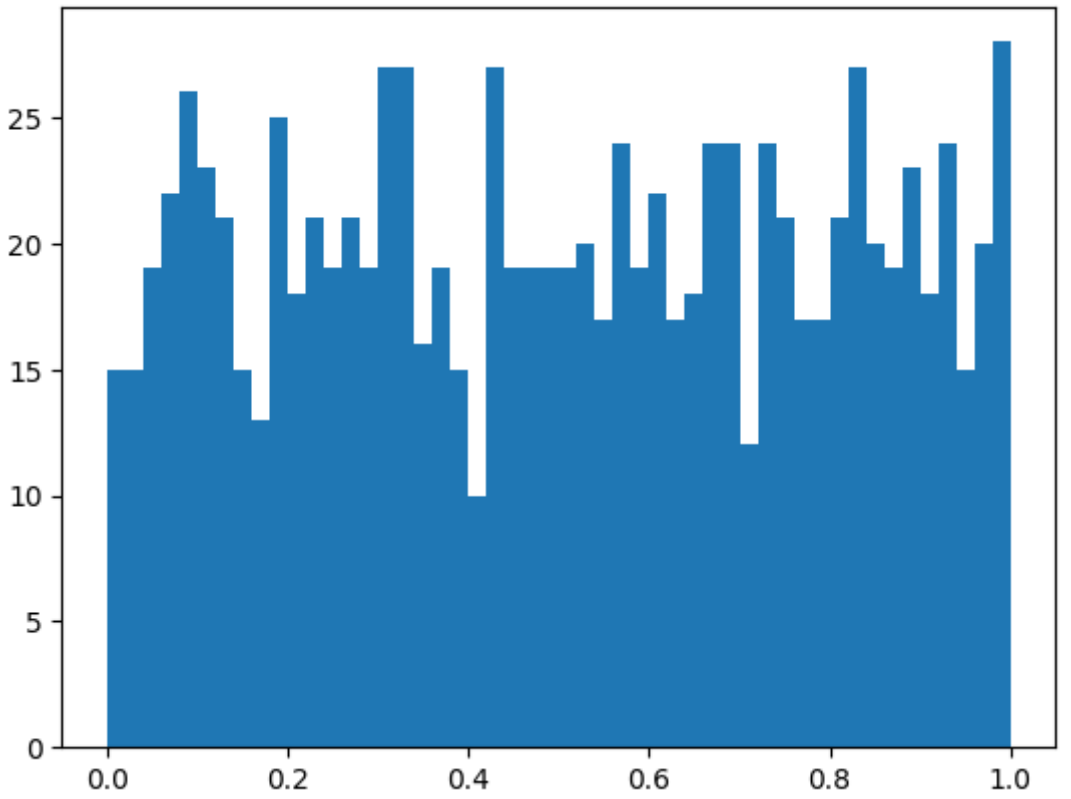
Q2.

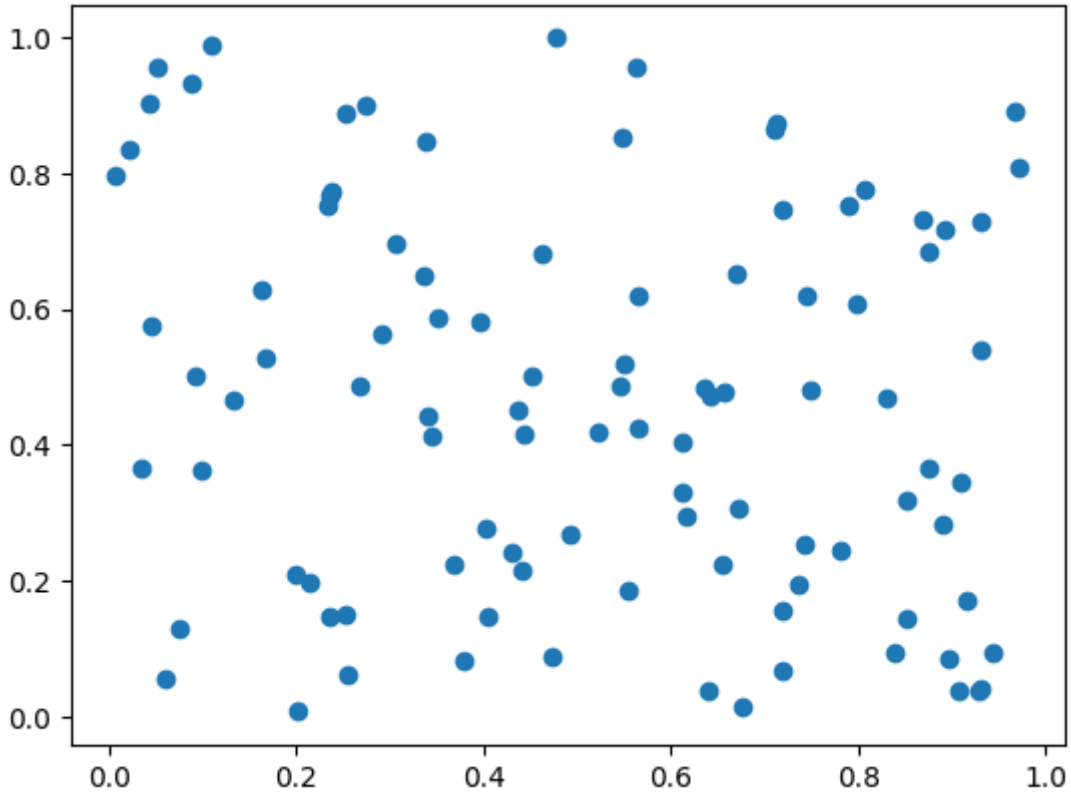
1. Randomly generate a vector and draw corresponding histogram
2. Randomly generate two vectors and draw scatter plot

In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
#Q2.1
x = np.random.rand(1000)

plt.hist(x, bins=50)
plt.show()

#Q2.2
```

```
x = np.random.rand(100)
y = np.random.rand(100)

plt.scatter(x, y)
plt.show()
```





Q3. Predict the output of the following code: x = [1,2] y = [[4, 1], [2, 2]] print(np.dot(x, y)) print(np.dot(y, x)) print(np.inner(x, y)) print(np.inner(y, x)) And type it in Python to see if they

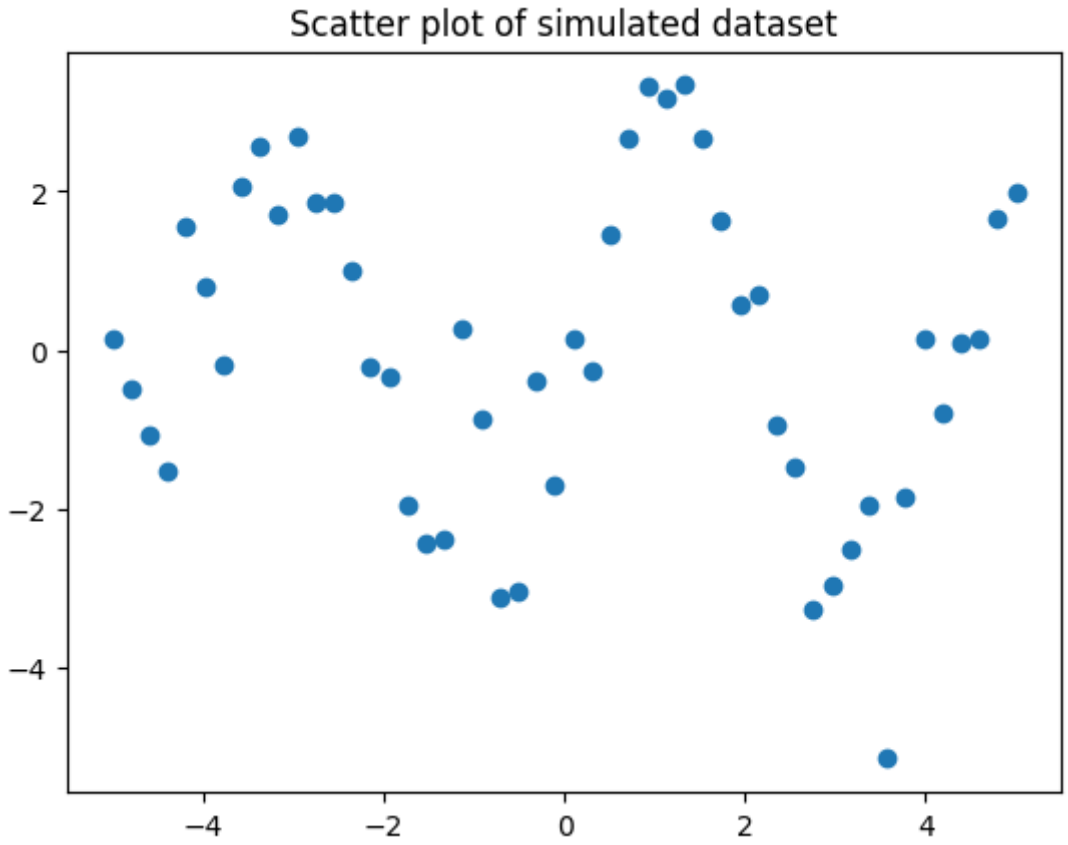
match your prediction

In [4]:
```
import numpy as np

x = [1,2]
y = [[4, 1], [2, 2]]

print(np.dot(x, y))
print(np.dot(y, x))
print(np.inner(x, y))
print(np.inner(y, x))
```

```
[8 5]
[6 6]
[6 6]
[6 6]
```

Q4. Curve fitting for a sin function. First of all, randomly generate a dataset following sin function (you can refer to the code below for the generation of such a simulated dataset. You can change the parameter values in this data generation process). x_data = np.linspace(-5, 5, num=50) y_data = 2.9 *np.sin(1.5* x_data) + np.random.normal(size=50) plot this dataset using scatter plot. Then use curve_fit function in numpy to fit a sin function specified below: $y = a \cdot Sin(b \cdot x)$. Here a and b are the two parameter we want to estimate using curve_fit.

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

x_data = np.linspace(-5, 5, num=50)
y_data = 2.9 * np.sin(1.5 * x_data) + np.random.normal(size=50)

plt.scatter(x_data, y_data)
plt.title("Scatter plot of simulated dataset")
plt.show()

def sin_func(x, a, b):
    return a * np.sin(b * x)

popt, pcov = curve_fit(sin_func, x_data, y_data)

print("Estimated value of a:", popt[0])
print("Estimated value of b:", popt[1])

plt.scatter(x_data, y_data)
plt.plot(x_data, sin_func(x_data, *popt), 'r-', label='fit')
plt.title("Scatter plot of simulated dataset with fitted sin function")
plt.legend()
plt.show()
```




```
Estimated value of a: 2.496255859783352
Estimated value of b: 1.4477924563947293
```