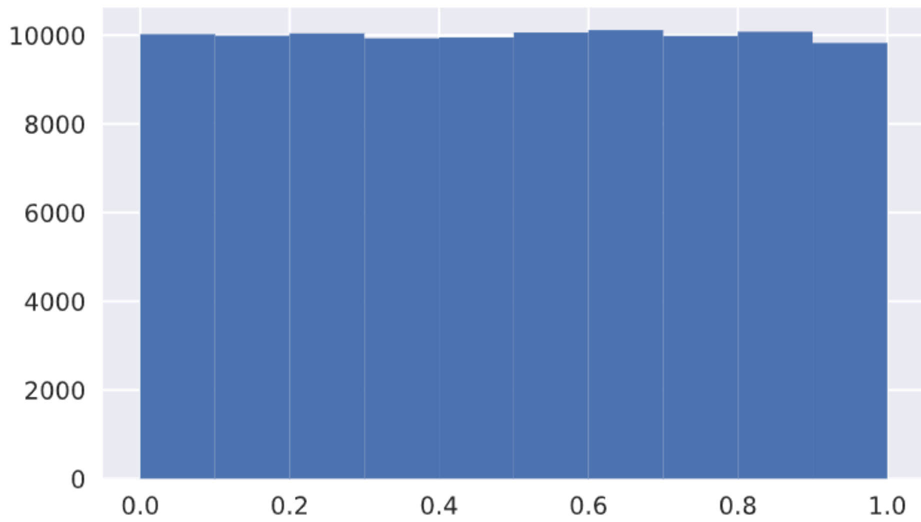
Statistical Thinking in Python Part 1

3). Thinking probabilistically-- Discrete variables

a). Generating random numbers using the np.random module

```
# Seed the random number generator
np.random.seed(42)

# Initialize random numbers: random_numbers
random_numbers = np.empty(100000)

# Generate random numbers by looping over range(100000)
for i in range(100000):
    random_numbers[i] = np.random.random()

# Plot a histogram
_ = plt.hist(random_numbers)

# Show the plot
plt.show()
```

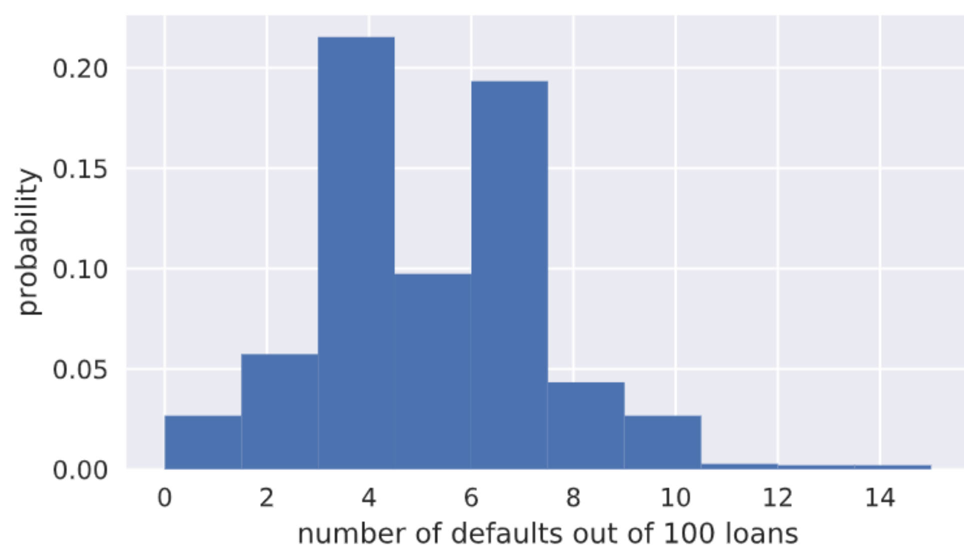


b). <u>The np.random module and Bernoulii trials</u>

```
def perform_bernoulli_trials(n, p):
    """Perform n Bernoulli trials with success probability p
    and return number of successes."""
    # Initialize number of successes: n_success
    n_success = 0



    # Perform trials
    for i in range(n):
        # Choose random number between zero and one: random_number
        random_number = np.random.random()

        # If less than p, it's a success so add one to n_success
        if random_number < p:
            n_success +=1

    return n_success
```

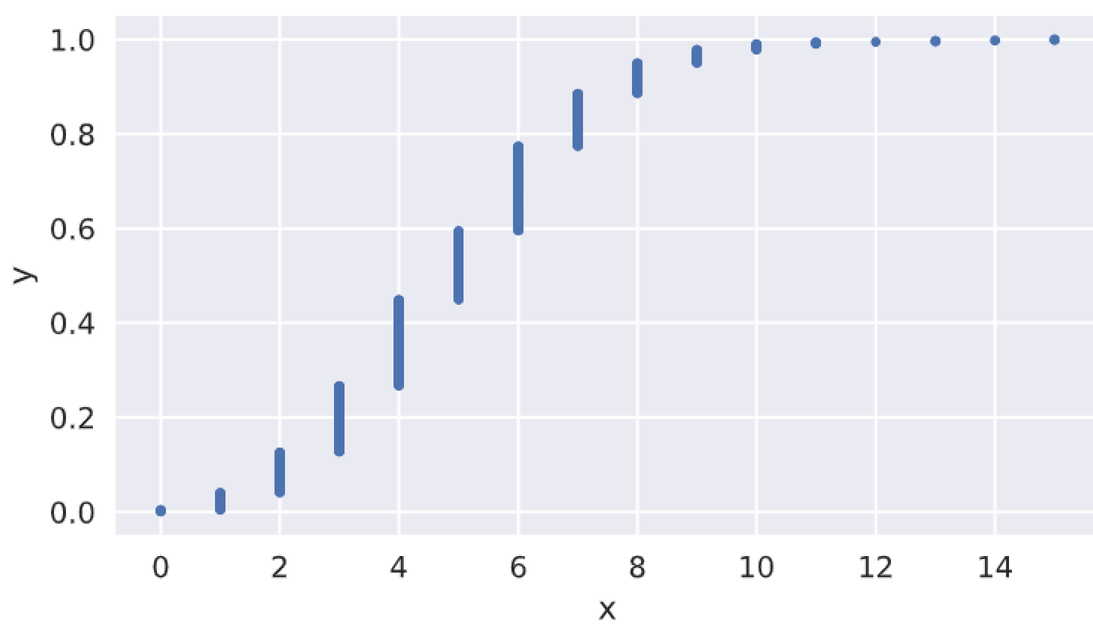
c). **How many defaults we might expect**

```
# Seed random number generator
np.random.seed(42)

# Initialize the number of defaults: n_defaults
n_defaults= np.empty(1000)

# Compute the number of defaults
for i in range(1000):
    n_defaults[i] = perform_bernoulli_trials(100,0.05)



# Plot the histogram with default number of bins; label your axes
_ = plt.hist(n_defaults, normed=True)
_ = plt.xlabel('number of defaults out of 100 loans')
_ = plt.ylabel('probability')

# Show the plot
plt.show()
```



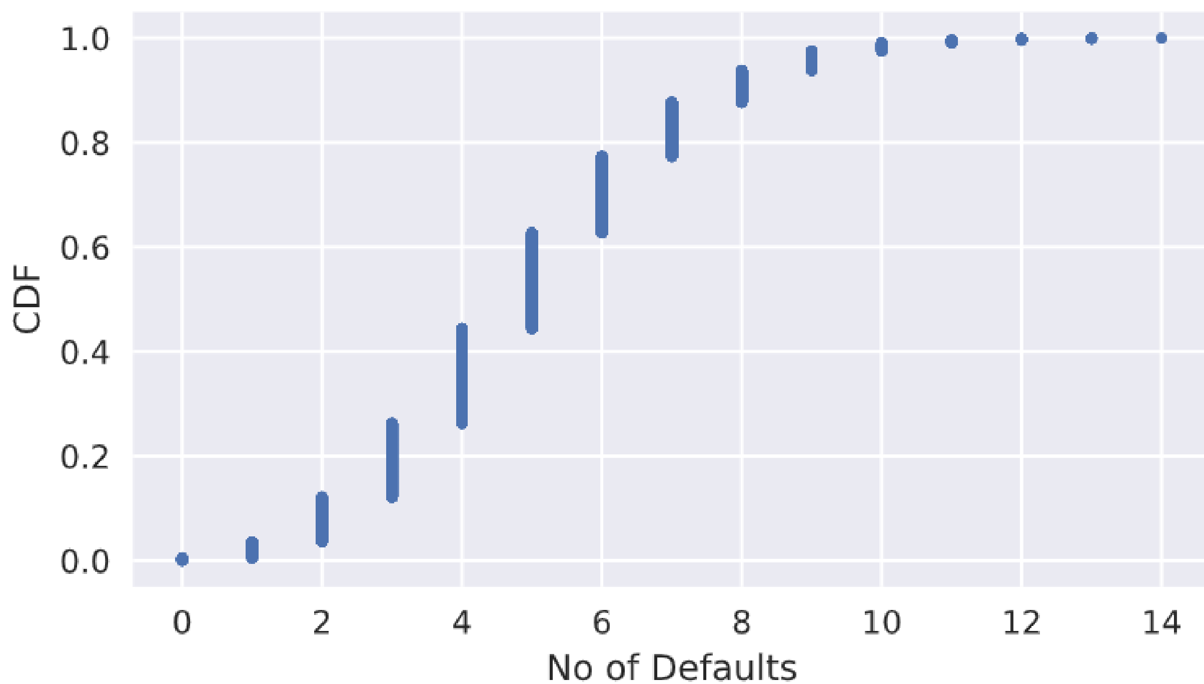
d). Will the bank fail

```
# Compute ECDF: x, y
x,y= ecdf(n_defaults)

# Plot the ECDF with labeled axes
plt.plot(x, y, marker='.', linestyle='none')
plt.xlabel('x')
plt.ylabel('y')

# Show the plot
plt.show()

# Compute the number of 100-loan simulations with 10 or more defaults: n_lose_money
n_lose_money=np.sum(n_defaults >= 10)

# Compute and print probability of losing money
print('Probability of losing money =', n_lose_money / len(n_defaults))
```



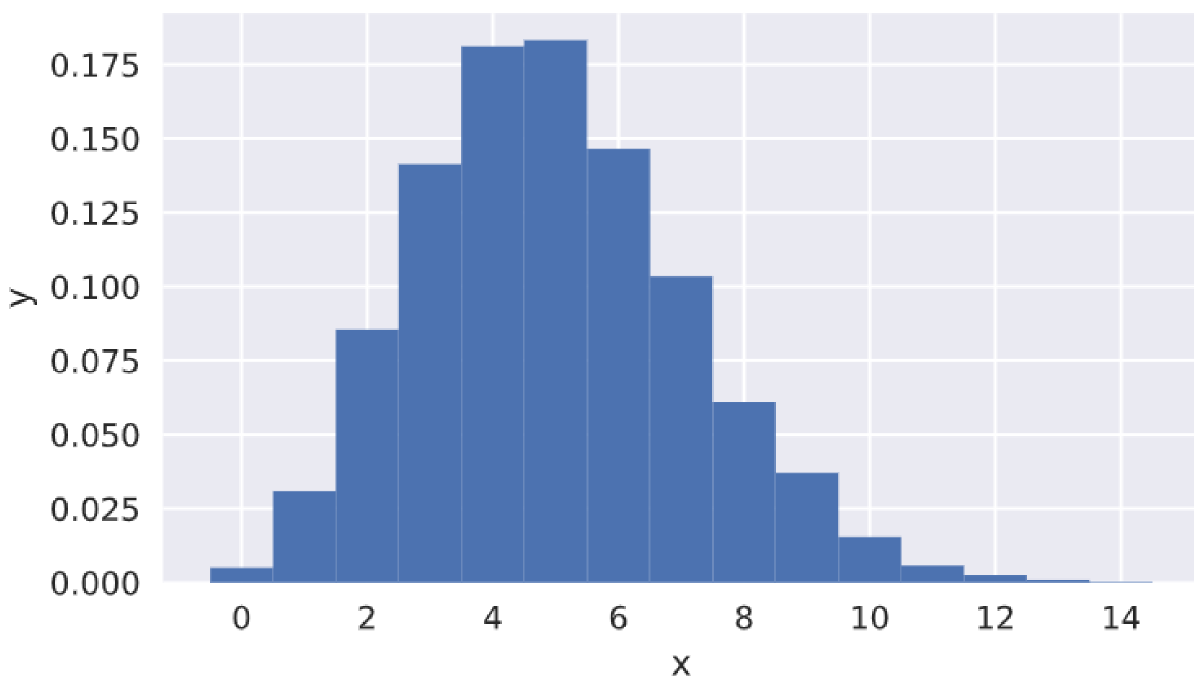
e). Sampling out of the Binomial Distribution

```
# Take 10,000 samples out of the binomial distribution: n_defaults
n_defaults=np.random.binomial(n=100, p=0.05, size=10000)

# Compute CDF: x, y
x,y=ecdf(n_defaults)

# Plot the CDF with axis labels
_=plt.plot(x, y, marker='.', linestyle='none')
_=plt.xlabel('No of Defaults')
_=plt.ylabel('CDF')

# Show the plot
plt.show()
```



f). Plotting the binomial PMF

```
# Compute bin edges: bins
bins = np.arange(min(n_defaults), max(n_defaults) + 1.5) - 0.5

# Generate histogram
plt.hist(n_defaults, normed=True, bins=bins)

# Label axes
plt.xlabel('x')
plt.ylabel('y')

# Show the plot
plt.show()
```



g). Relationship between Bionomial and Poisson Distribution:

```
# Draw 10,000 samples out of Poisson distribution: samples_poisson
samples_poisson= np.random.poisson(10, size= 10000)

# Print the mean and standard deviation
print('Poisson:     ', np.mean(samples_poisson), np.std(samples_poisson))

# Specify values of n and p to consider for Binomial: n, p
n=[20,100,1000]
p=[0.5,0.1,0.01]

# Draw 10,000 samples for each n,p pair: samples_binomial
for i in range(3):
    samples_binomial = np.random.binomial(n[i], p[i],10000)

    # Print results
    print('n =', n[i], 'Binom:', np.mean(samples_binomial), np.std(samples_binomial))
```

<script.py> output:

```
    Poisson:      10.0186 3.144813832327758
    n = 20 Binom: 9.9637 2.2163443572694206
    n = 100 Binom: 9.9947 3.0135812433050484
    n = 1000 Binom: 9.9985 3.139378561116833
```

h). Was 2015 anomalous

```
# Draw 10,000 samples out of Poisson distribution: n_nohitters
n_nohitters= np.random.poisson(251/115, size=10000)

# Compute number of samples that are seven or greater: n_large
n_large = np.sum(n_nohitters >= 7)

# Compute probability of getting seven or more: p_large
p_large= n_large/10000

# Print the result
print('Probability of seven or more no-hitters:', p_large)
```

<script.py> output:

```
Probability of seven or more no-hitters: 0.0067
```