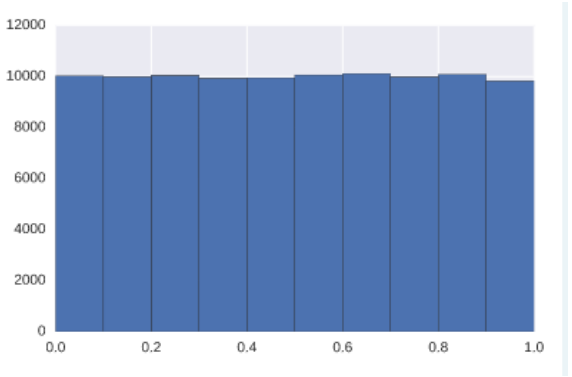
Ch 3 Thinking probabilistically-- Discrete variables

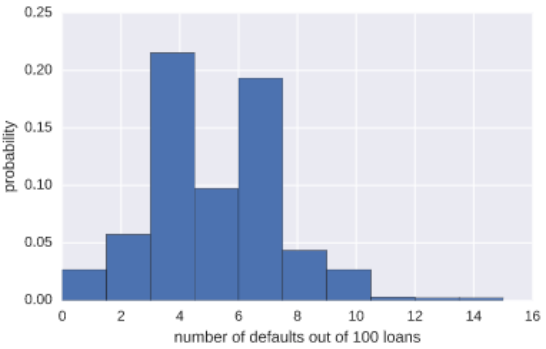
November-11-17 4:14 PM

```
#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()
```

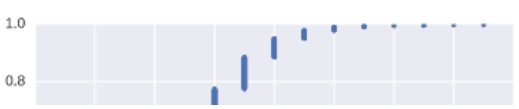



```
#The np.random module and Bernoulli trials
def perform_bernoulli_trials(n, p):
    """Perform n Bernoulli trials with success probability p
    and return number of successes."""
    # Initialize number of successes: n_success
    n_success = 0


    # 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
```

```
#How many defaults might we expect?
# Seed random number generator
np.random.seed (42)

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

#p = probability of defaults
p = 0.05

#n = 100 loans
n = 100

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


# 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 ()
```



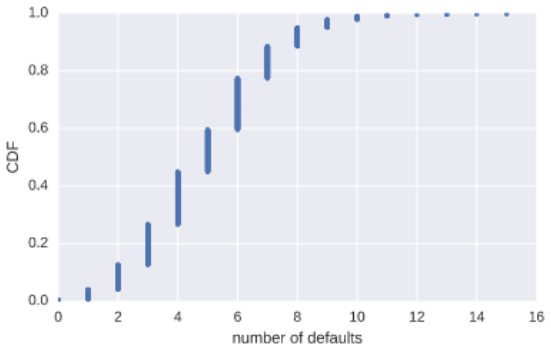

```
#Will the bank fail?
# Compute ECDF: x, y
x, y = ecdf (n_defaults)
```



#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 ('number of defaults')
_ = plt.ylabel ('CDF')

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)

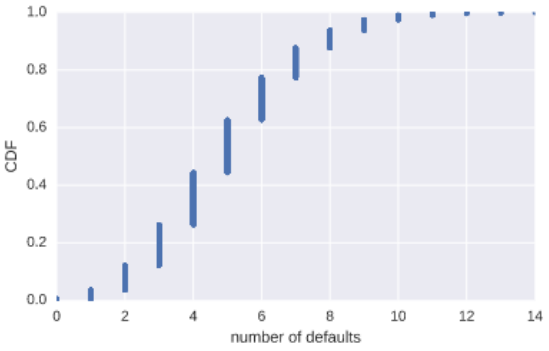
# Compute and print probability of losing money
print('Probability of losing money =', n_lose_money / len(n_defaults))

#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 ('number of defaults')
_= plt.ylabel ('CDF')

Show the plot
plt.show ()



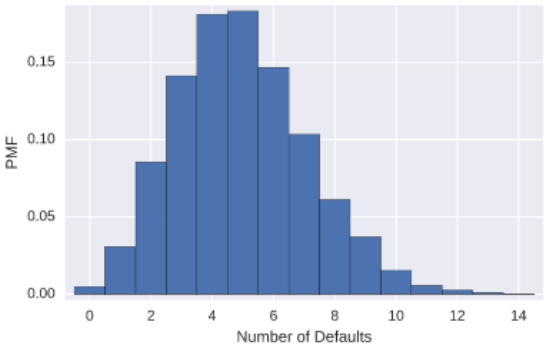
#Plotting the Binomial PMF
# Compute bin edges: bins
bins = np.arange(0, max(n_defaults) + 2) - 0.5

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

Set margins
plt.margins (0.02)

Label axes
plt.xlabel ('Number of Defaults')
plt.ylabel ('PMF')

Show the plot
plt.show ()



#Relationship between Binomial and Poisson distributions
# 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))

```
#Was 2015 anomalous?
# Draw 10,000 samples out of Poisson distribution: n_nohitters
n_nohitters = np.random.poisson (251/115, 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)
```