*You are currently looking at **version 1.0** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the [Jupyter Notebook FAQ (https://www.coursera.org/learn/python-data-analysis/resources/0dhYG)](https://www.coursera.org/learn/python-data-analysis/resources/0dhYG) course resource.*

Distributions in Pandas

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
'''
Suppose we want to simulate the probability of flipping a fair coin 20 times
, and getting a number greater than or equal to 15. Use np.random.binomial(n, p,
size)
to do 10000 simulations of flipping a fair coin 20 times, then see what proporti
on of the simulations are 15 or greater.
'''
x = np.random.binomial(20, .5, 10000)
print((x>=15).mean())
```

```
0.0234
```

In [3]:
```
np.random.binomial(1, 0.5)
```

Out[3]:
```
1
```

In [4]:
```
np.random.binomial(5000, 0.35)/5000
```

Out[4]:
```
0.3548
```

In [5]:
```
x = np.random.ranf()
z = np.random.binomial(1000, x, 102)/1000
print(x)
print(z)
print(z.mean())
```

```
0.27826569985021343
[ 0.285  0.27   0.269  0.252  0.274  0.277  0.276  0.296  0.28   0.287
  0.272  0.274  0.298  0.303  0.284  0.288  0.253  0.297  0.279  0.28
  0.269  0.267  0.279  0.278  0.272  0.255  0.277  0.271  0.286  0.273
  0.274  0.279  0.277  0.307  0.276  0.296  0.284  0.29   0.305  0.284
  0.277  0.274  0.272  0.302  0.298  0.265  0.281  0.287  0.259  0.258
  0.268  0.277  0.267  0.289  0.269  0.247  0.269  0.293  0.262  0.286
  0.276  0.285  0.285  0.27   0.273  0.274  0.277  0.29   0.269  0.265
  0.265  0.28   0.297  0.279  0.321  0.275  0.261  0.277  0.282  0.273
  0.295  0.309  0.3    0.278  0.276  0.286  0.281  0.286  0.31   0.267
  0.29   0.259  0.252  0.294  0.272  0.266  0.275  0.282  0.278  0.264
  0.266  0.286]
0.27881372549
```

In [6]:
```
# unevenly weighted
chance_of_tornado = 0.01/100
np.random.binomial(100000, chance_of_tornado)
```

Out[6]: 15

In [7]:
```
chance_of_tornado = 0.01
sum = 0
x = 1000000/365 # number of years

tornado_events = np.random.binomial(1, chance_of_tornado, int(365*x))

print(tornado_events[0:100])
print(tornado_events.mean())

two_days_in_a_row = 0
for j in range(1,len(tornado_events)-1):
    if tornado_events[j]==1 and tornado_events[j-1]==1:
        two_days_in_a_row+=1

for i in range(1, len(tornado_events)-1):
    if tornado_events[i] == 1:
        sum += 1
#print('{} tornadoes back to back in {} years'.format(two_days_in_a_row, 1000000
/365))
print(f'{two_days_in_a_row} tornadoes back to back in {x} years')
print(f'{sum} tornadoe(s) in {x}years')
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
0.009911
119 tornadoes back to back in 2739.72602739726 years
9911 tornadoe(s) in 2739.72602739726years
```

In [8]:
```
np.random.uniform(0, 1)
```

Out[8]: 0.46235787224468294

In [9]:
```
np.random.normal(0.75)
```

Out[9]: 0.8462291338979055

Formula for standard deviation

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

In [10]:
```
distribution = np.random.normal(0.75,size=1000)

np.sqrt(np.sum((np.mean(distribution)-distribution)**2)/len(distribution))
```

Out[10]: 0.99374325582731005

In [11]:
```
distribution[0:10]
```

Out[11]:
```
array([ 0.02528767,  2.54654995,  1.21273655,  0.54863908,  1.45094085,
        2.73063173, -0.14579252,  2.3936933 ,  0.73879671,  0.03829581])
```

In [12]:
```
x = (distribution-np.mean(distribution))**2
print(x[0:10])
y = (np.mean(distribution)-distribution)**2
print(y[0:10])
print(np.mean(distribution))
```

```
[  5.26945543e-01   3.22328928e+00   2.13017999e-01   4.10300405e-02
   4.89640312e-01   3.91815864e+00   8.04591668e-01   2.69779143e+00
   1.53787345e-04   5.08229271e-01]
[  5.26945543e-01   3.22328928e+00   2.13017999e-01   4.10300405e-02
   4.89640312e-01   3.91815864e+00   8.04591668e-01   2.69779143e+00
   1.53787345e-04   5.08229271e-01]
0.751197815752
```

In [13]:
```
np.random.normal?
```

In [14]:
```
np.std(distribution)
```

Out[14]:
```
0.99374325582731005
```

In [15]:
```
import scipy.stats as stats
stats.kurtosis(distribution)
```

Out[15]:
```
0.057261953745207705
```

In [16]:
```
stats.skew(distribution)
```

Out[16]:
```
0.08946591005782702
```

In [17]:
```
print(stats.skew([1, 2, 3, 4, 5, 6]))
print(stats.skew([1, 4, 6, 8, 10, 12]))
print(stats.skew([1, 2**3, 3**3, 4**3, 5**3, 6**3]))
print(stats.skew([1, 5, 10, 15, 20]))
```

```
0.0
-0.17039575849885127
0.8568494780241179
0.08448539032773617
```

In [18]:
```
chi_squared_df2 = np.random.chisquare(2, size=10000)
stats.skew(chi_squared_df2)
```

Out[18]:
```
1.984520065939756
```

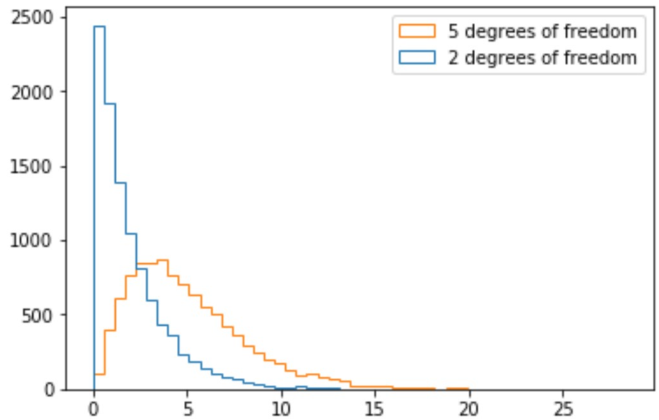
In [19]:
```
chi_squared_df5 = np.random.chisquare(5, size=10000)
stats.skew(chi_squared_df5)
```

Out[19]:
```
1.2152057290912823
```

In [20]:
```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

output = plt.hist([chi_squared_df2,chi_squared_df5], bins=50, histtype='step',
                  label=['2 degrees of freedom','5 degrees of freedom'])
plt.legend(loc='upper right')
```

Out[20]: `<matplotlib.legend.Legend at 0x7f734a4efdd8>`



Hypothesis Testing

In [21]:
```
df = pd.read_csv('grades.csv')
```

In [22]:
```
df.head()
```

Out[22]:

	student_id	assignment1_grade	assignment1_submission	assignment2_grade	a
0	B73F2C11-70F0-E37D-8B10-1D20AFED50B1	92.733946	2015-11-02 06:55:34.282000000	83.030552	2 0.
1	98A0FAE0-A19A-13D2-4BB5-CFBFD94031D1	86.790821	2015-11-29 14:57:44.429000000	86.290821	2 1
2	D0F62040-CEB0-904C-F563-2F8620916C4E	85.512541	2016-01-09 05:36:02.389000000	85.512541	2 0
3	FFDF2B2C-F514-EF7F-6538-A6A53518E9DC	86.030665	2016-04-30 06:50:39.801000000	68.824532	2 1
4	5ECBEEB6-F1CE-80AE-3164-E45E99473FB4	64.813800	2015-12-13 17:06:10.750000000	51.491040	2 1.

In [23]:
```
len(df)
```

Out[23]: 2315

In [24]:
```
early = df[df['assignment1_submission'] <= '2015-12-31']
late = df[df['assignment1_submission'] > '2015-12-31']
```

In [25]:
```
early.mean()
```

Out[25]:
```
assignment1_grade    74.972741
assignment2_grade    67.252190
assignment3_grade    61.129050
assignment4_grade    54.157620
assignment5_grade    48.634643
assignment6_grade    43.838980
dtype: float64
```

In [26]:
```
late.mean()
```

Out[26]:
```
assignment1_grade    74.017429
assignment2_grade    66.370822
assignment3_grade    60.023244
assignment4_grade    54.058138
assignment5_grade    48.599402
assignment6_grade    43.844384
dtype: float64
```

In [27]:
```
from scipy import stats
stats.ttest_ind?
```

In [28]:
```
stats.ttest_ind(early['assignment1_grade'], late['assignment1_grade'])
```

Out[28]:
```
Ttest_indResult(statistic=1.400549944897566, pvalue=0.16148283016060577)
```

In [29]:
```
stats.ttest_ind(early['assignment2_grade'], late['assignment2_grade'])
```

Out[29]:
```
Ttest_indResult(statistic=1.3239868220912567, pvalue=0.18563824610067967)
```

In [30]:
```
stats.ttest_ind(early['assignment3_grade'], late['assignment3_grade'])
```

Out[30]:
```
Ttest_indResult(statistic=1.7116160037010733, pvalue=0.08710151634155667 6)
```