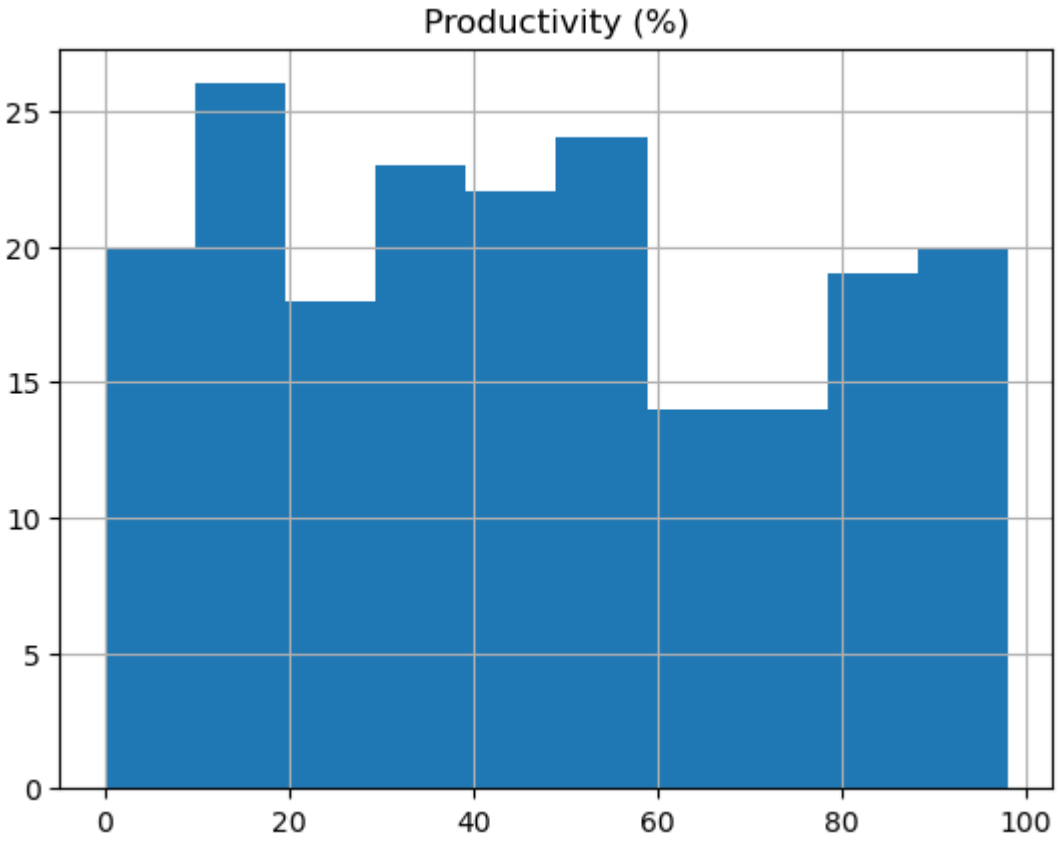
In [55]:
```
import matplotlib.pyplot as plt
from scipy.stats import poisson
import numpy as np
import pandas as pd
import thinkplot
import thinkstats2
hrData = pd.read_csv(r"C:\Users\bwych\Downloads\archive (1)\hr_dashboard_d
age = hrData['Age']
projCompleted = hrData['Projects Completed']
satisfaction = hrData['Satisfaction Rate (%)']
feedbackScore = hrData['Feedback Score']
salary = hrData['Salary']
```

In [ ]:
```
# Histograms
```

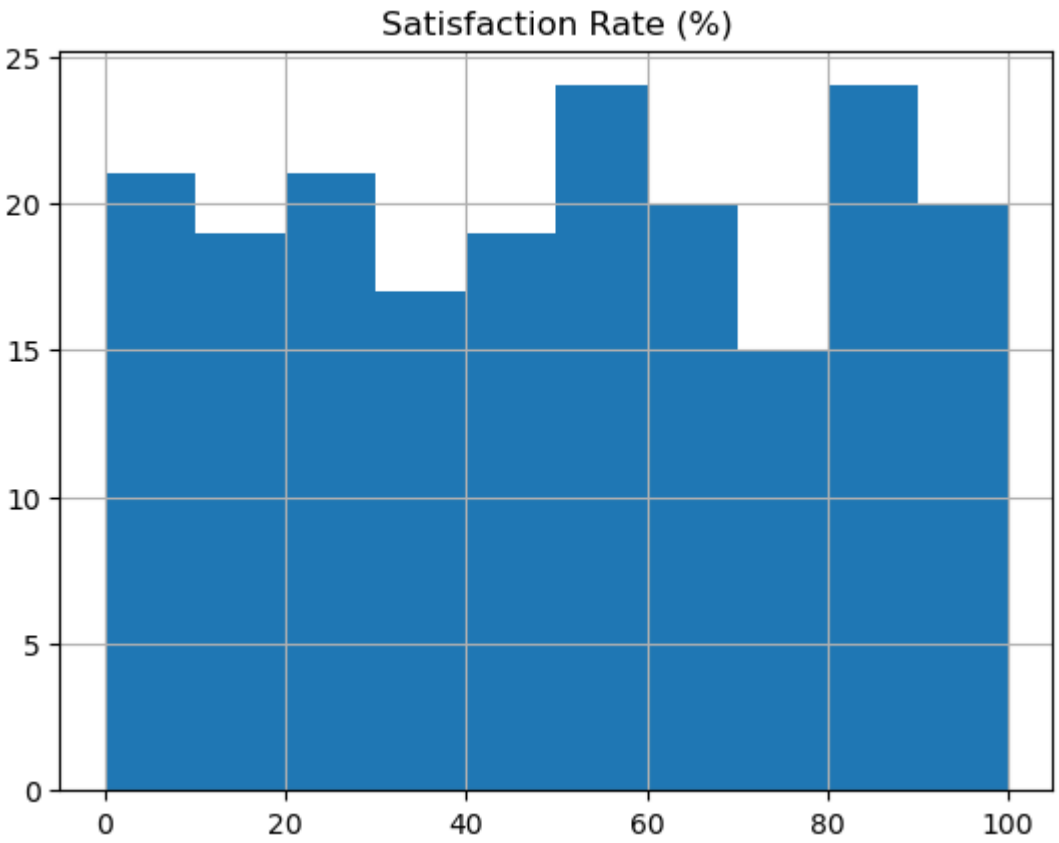
In [56]:
```
hrData.hist(column='Productivity (%)')
```

Out[56]:
```
array([[<Axes: title={'center': 'Productivity (%)'}>]], dtype=object)
```



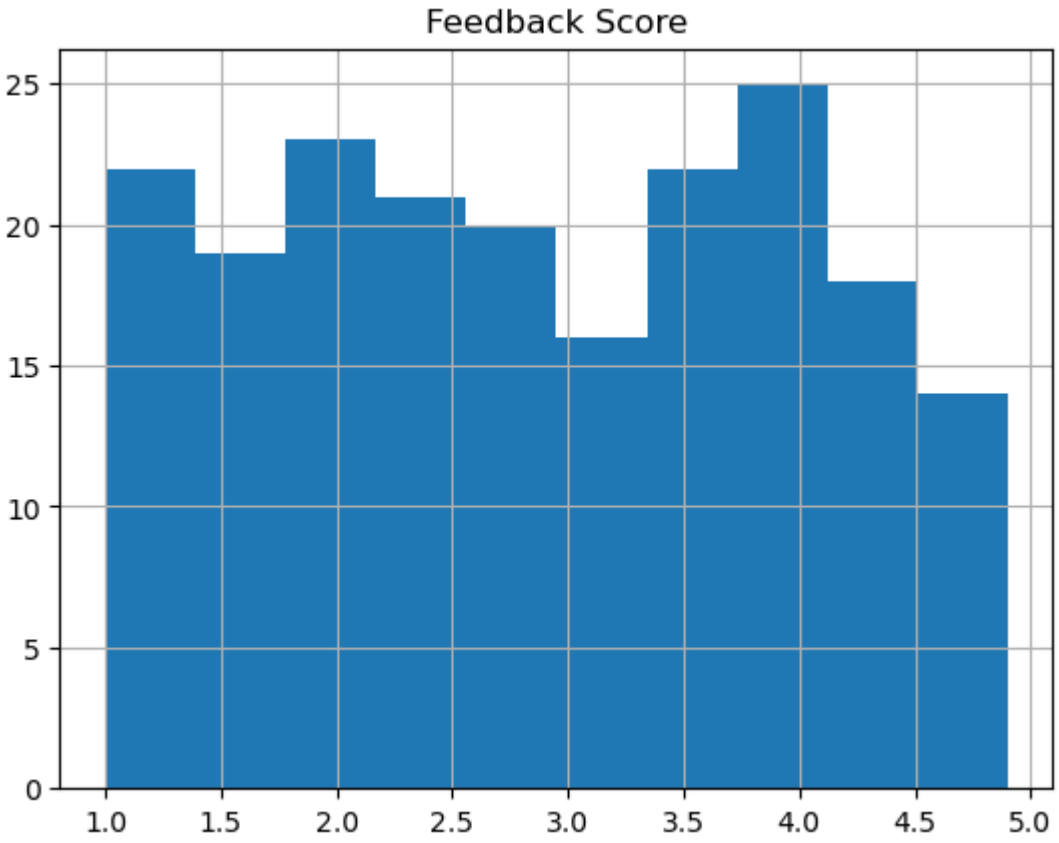
In [81]:
```
hrData.hist(column='Satisfaction Rate (%)')
```

Out[81]:
```
array([[<Axes: title={'center': 'Satisfaction Rate (%)'}>]], dtype=objec
t)
```



In [82]:
```
hrData.hist(column='Feedback Score')
```

Out[82]:
```
array([[<Axes: title={'center': 'Feedback Score'}>]], dtype=object)
```

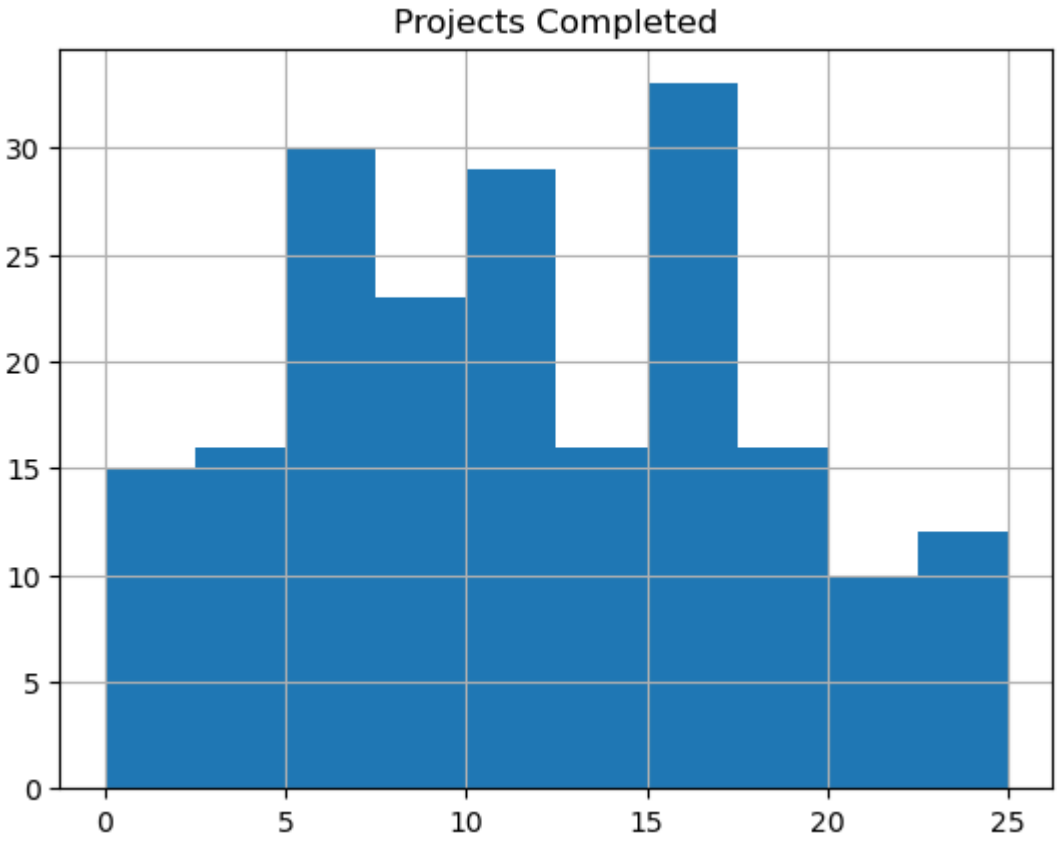


In [83]:
```
hrData.hist(column='Salary')
```

Out[83]:
```
array([[<Axes: title={'center': 'Salary'}>]], dtype=object)
```

In [84]:
```
hrData.hist(column='Projects Completed')
```

Out[84]:
```
array([[<Axes: title={'center': 'Projects Completed'}>]], dtype=object)
```



In [ ]:
```
# Descriptions
```

In [57]:
```
hrData.mean(axis=0)
```

```
C:\Users\bwych\AppData\Local\Temp\ipykernel_2868\2814385828.py:1: FutureW
arning: The default value of numeric_only in DataFrame.mean is deprecate
d. In a future version, it will default to False. In addition, specifying
'numeric_only=None' is deprecated. Select only valid columns or specify t
he value of numeric_only to silence this warning.
  hrData.mean(axis=0)
```

Out[57]:
```
Age                       34.650
Projects Completed        11.455
Productivity (%)          46.755
Satisfaction Rate (%)     49.935
Feedback Score             2.883
Salary                 76619.245
dtype: float64
```

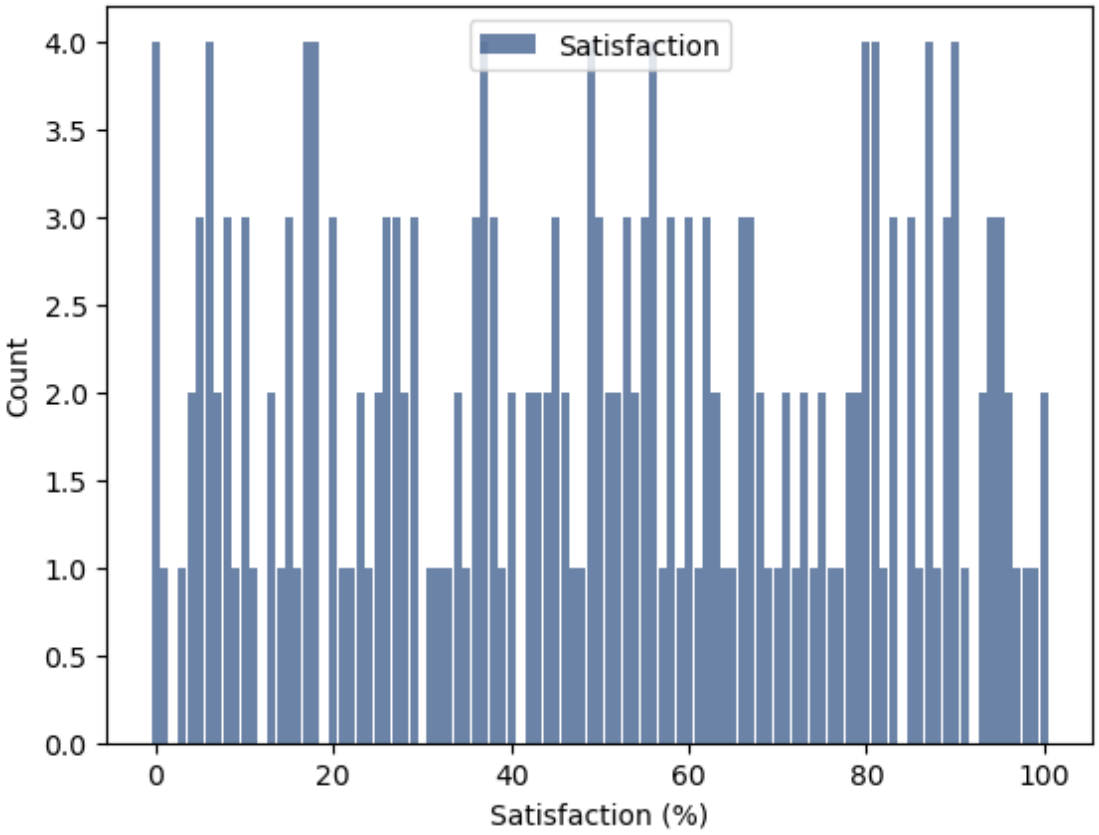
In [59]:
```
hrData.describe()
```

Out[59]:

	Age	Projects Completed	Productivity (%)	Satisfaction Rate (%)	Feedback Score	Salary
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000
mean	34.650000	11.455000	46.755000	49.935000	2.883000	76619.245000
std	9.797318	6.408849	28.530068	28.934353	1.123263	27082.299202
min	22.000000	0.000000	0.000000	0.000000	1.000000	30231.000000
25%	26.000000	6.000000	23.000000	25.750000	1.900000	53080.500000
50%	32.000000	11.000000	45.000000	50.500000	2.800000	80540.000000
75%	41.000000	17.000000	70.000000	75.250000	3.900000	101108.250000
max	60.000000	25.000000	98.000000	100.000000	4.900000	119895.000000

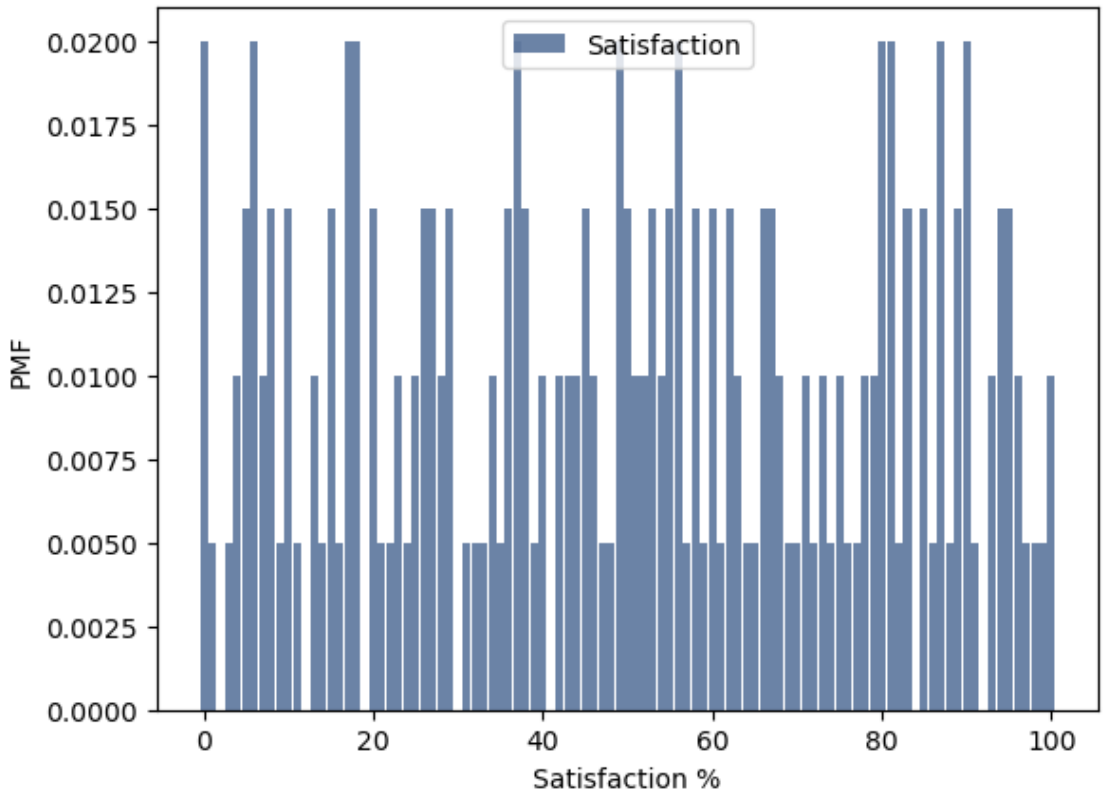
In [ ]:
```
# PMF
```

In [61]:
```
hist = thinkstats2.Hist(satisfaction, label='Satisfaction')
thinkplot.Hist(hist)
thinkplot.Config(xlabel='Satisfaction (%)', ylabel='Count')
```



In [62]:
```
n = hist.Total()
pmf = hist.Copy()
for x, freq in hist.Items():
    pmf[x] = freq / n
```

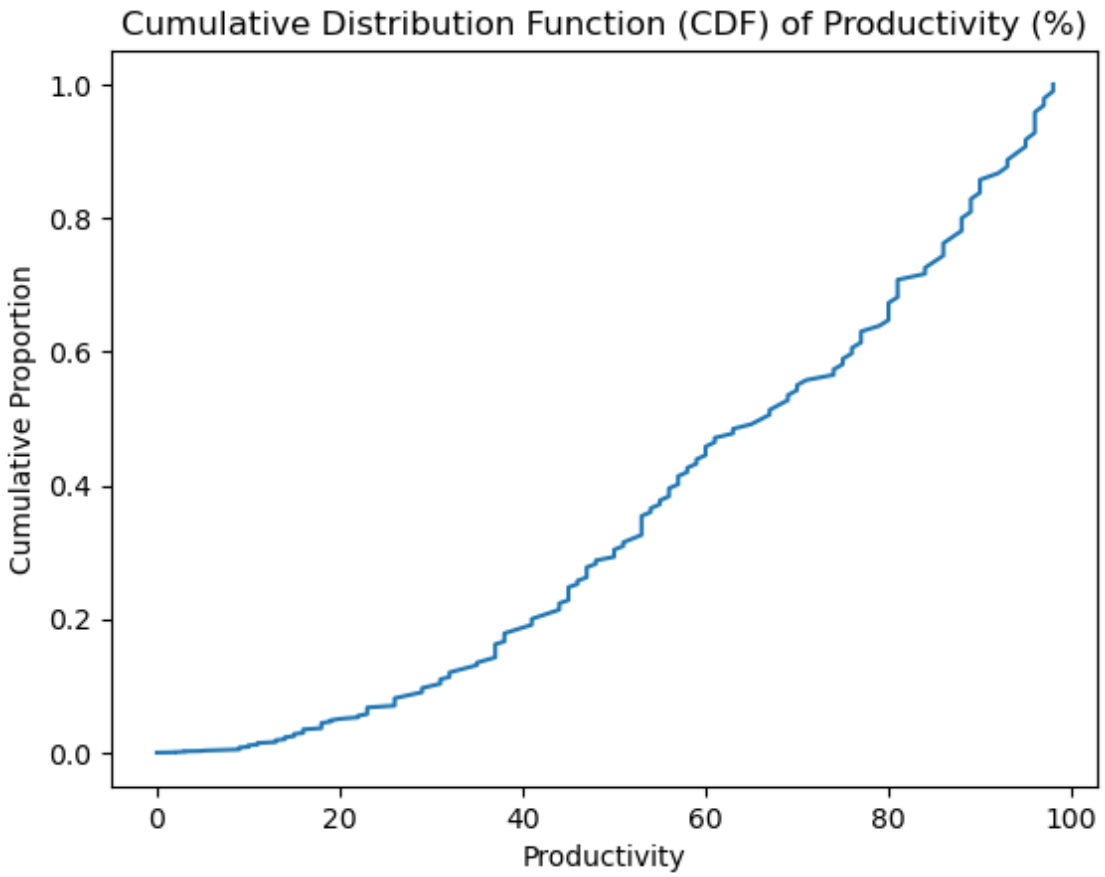
In [63]:
```
thinkplot.Hist(pmf)
thinkplot.Config(xlabel='Satisfaction %', ylabel='PMF')
```



In [ ]:
```
# CDF
```

In [54]:
```
productivity = hrData["Productivity (%)"]

cumulative = np.linspace(0, 1, len(productivity))
sorted_data = np.sort(productivity)
cumulative_data = np.cumsum(sorted_data) / np.sum(sorted_data)

plt.plot(sorted_data, cumulative_data)

plt.xlabel("Productivity")
plt.ylabel("Cumulative Proportion")
plt.title("Cumulative Distribution Function (CDF) of Productivity (%)")
```

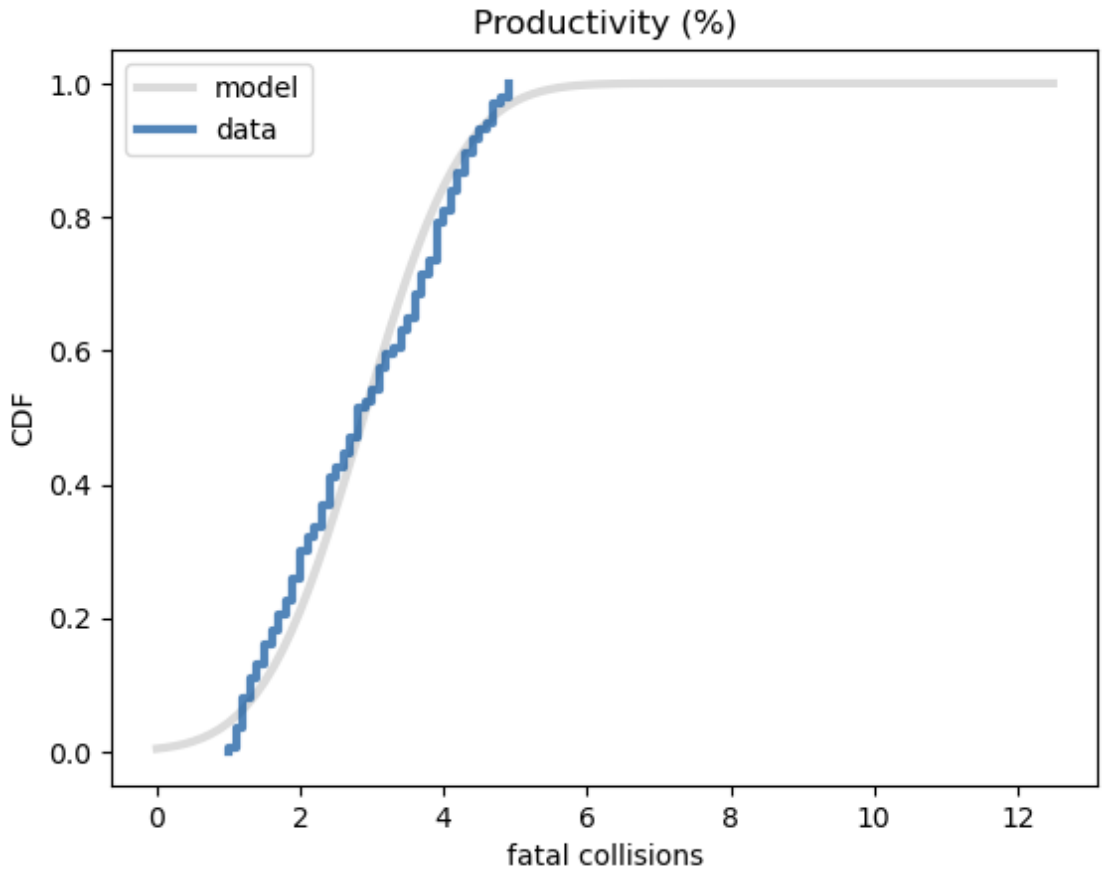
Out[54]:
```
Text(0.5, 1.0, 'Cumulative Distribution Function (CDF) of Productivity
(%)')
```



In [ ]:
```
# Analytical Distribution
```

In [68]:
```
mu, var = thinkstats2.TrimmedMeanVar(feedbackScore, p=0.01)
print('Mean, Var', mu, var)

sigma = np.sqrt(var)
print('Sigma', sigma)
xs, ps = thinkstats2.RenderNormalCdf(mu, sigma, low=0, high=12.5)

thinkplot.Plot(xs, ps, label='model', color='0.8')
cdf = thinkstats2.Cdf(feedbackScore, label='data')

thinkplot.PrePlot(1)
thinkplot.Cdf(cdf)
thinkplot.Config(title='Feedback Score',
                 xlabel='score',
                 ylabel='CDF')
```

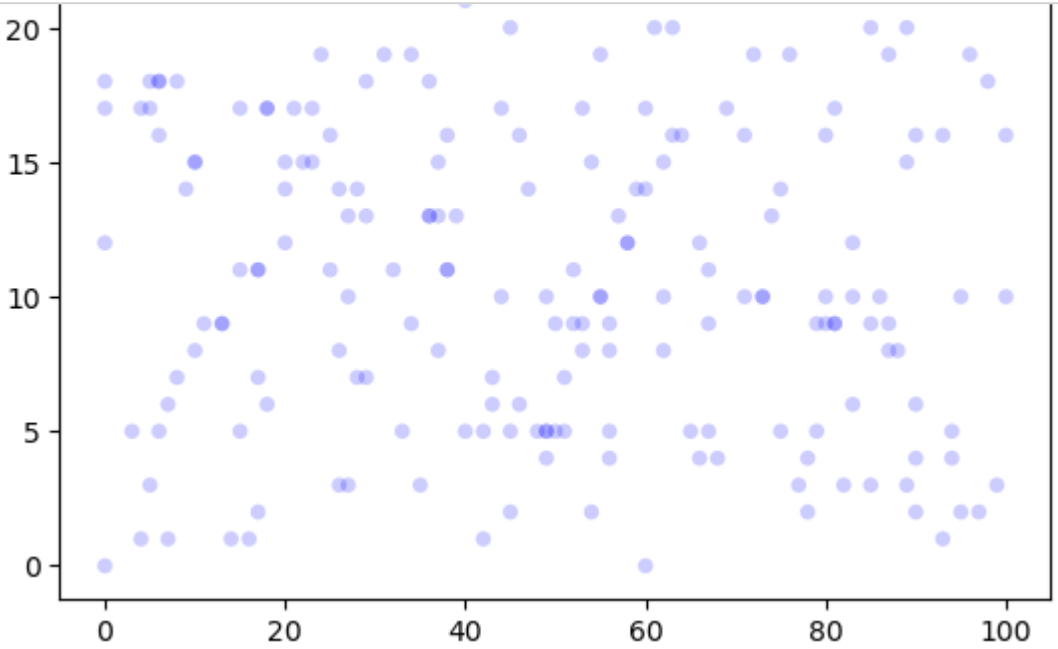
```
Mean, Var 2.8811224489795917 1.2052048625572678
Sigma 1.0978182283772062
```



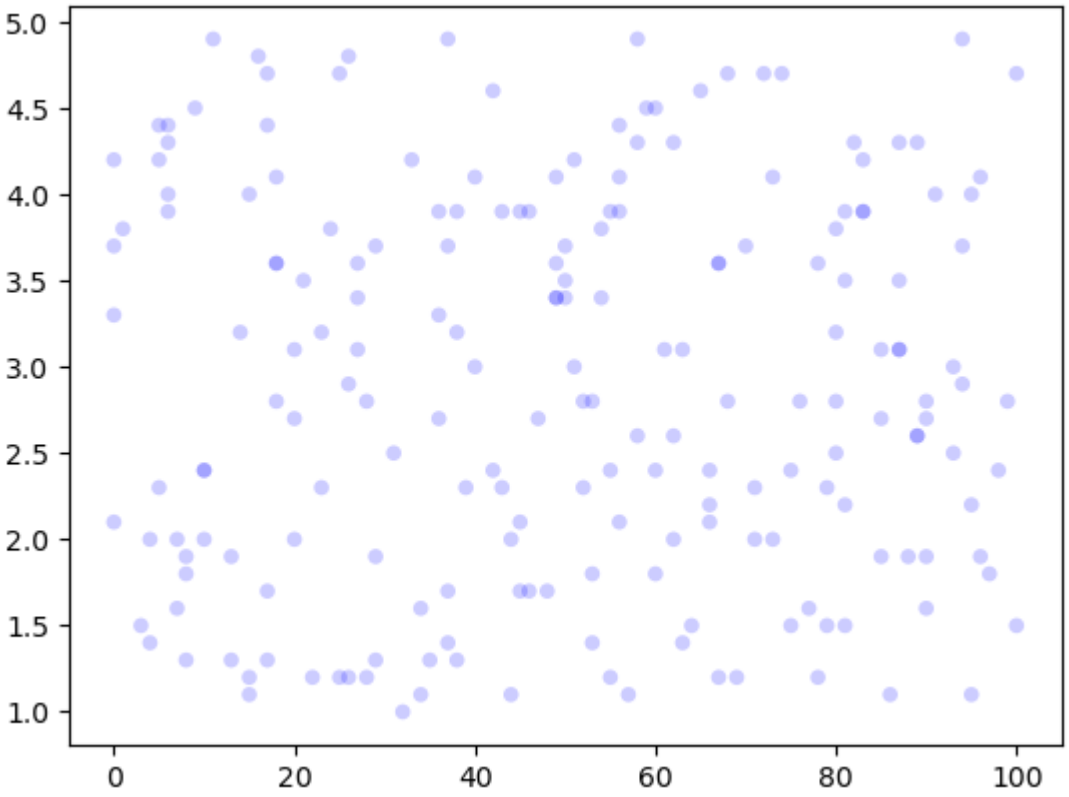
In [ ]:
```
# Scatter Plots
```

In [71]:
```
def Corr(xs, ys):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    meanx, varx = thinkstats2.MeanVar(xs)
    meany, vary = thinkstats2.MeanVar(ys)

    corr = Cov(xs, ys, meanx, meany) / np.sqrt(varx * vary)
    return corr
```

In [85]:
```
thinkplot.Scatter(satisfaction, projCompleted)
```



In [86]:
```
thinkplot.Scatter(satisfaction, feedbackScore)
```



In [76]:
```
Corr(satisfaction, projCompleted)
```

Out[76]: -0.01081480118624146

In [87]:
```
Corr(satisfaction, feedbackScore)
```

Out[87]: 0.008067658039439084

In [ ]:
```
# Test on Hypothesis
```

In [88]:
```
class DiffMeansPermute(thinkstats2.HypothesisTest):

    def TestStatistic(self, data):
        group1, group2 = data
        test_stat = abs(group1.mean() - group2.mean())
        return test_stat

    def MakeModel(self):
        group1, group2 = self.data
        self.n, self.m = len(group1), len(group2)
        self.pool = np.hstack((group1, group2))

    def RunModel(self):
        np.random.shuffle(self.pool)
        data = self.pool[:self.n], self.pool[self.n:]
        return data

data = satisfaction, productivity

ht = DiffMeansPermute(data)
pvalue = ht.PValue()
pvalue
```

Out[88]:
```
0.281
```

In [ ]:
```
# Regression Analysis
```

In [78]:
```
import statsmodels.formula.api as smf
formula = 'satisfaction ~ projCompleted'
model = smf.ols(formula, data=hrData)
results = model.fit()
results.summary()
```

Out[78]:

OLS Regression Results

Dep. Variable:	satisfaction	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.005
Method:	Least Squares	**F-statistic:**	0.02316
Date:	Fri, 25 Aug 2023	**Prob (F-statistic):**	0.879
Time:	13:43:50	**Log-Likelihood:**	-956.28
No. Observations:	200	**AIC:**	1917.
Df Residuals:	198	**BIC:**	1923.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	50.4943	4.209	11.998	0.000	42.195	58.794
projCompleted	-0.0488	0.321	-0.152	0.879	-0.682	0.584

Omnibus:	81.084	**Durbin-Watson:**	1.976
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	11.553
Skew:	-0.023	**Prob(JB):**	0.00310
Kurtosis:	1.823	**Cond. No.**	27.0

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.