

Code for Term Project DSC 530-T301

By: Samuel Castro

03/04/2023

Referencing: thinkStats Exploratory Data Analysis by Allen B. Downey 2nd Edition

```
In [1]: import pandas as pd
from thinkstats2 import Mean, MeanVar, Var, Std, Cov
```

```
In [2]: dataset = pd.read_csv("~/desktop/DSC528/Sleep_Efficiency.csv")
```

```
In [3]: dataset.head()
```

dataset.head()															
ID	Age	Gender	Bedtime	Wakeup time	Sleep duration	Sleep efficiency	REM sleep percentage	Deep sleep percentage	Light sleep percentage	Awakenings	Caffeine consumption	Alcohol consumption	Smoking status	Exercise frequency	
0	1	65	Female	2021-03-06 01:00:00	2021-03-06 07:00:00	6.0	0.88	18	70	12	0.0	0.0	0.0	Yes	3.0
1	2	69	Male	2021-12-05 02:00:00	2021-12-05 09:00:00	7.0	0.66	19	28	53	3.0	0.0	3.0	Yes	3.0
2	3	40	Female	2021-05-25 21:30:00	2021-05-25 05:30:00	8.0	0.89	20	70	10	1.0	0.0	0.0	No	3.0
3	4	40	Female	2021-11-03 02:30:00	2021-11-03 08:30:00	6.0	0.51	23	25	52	3.0	50.0	5.0	Yes	1.0
4	5	57	Male	2021-03-13 01:00:00	2021-03-13 09:00:00	8.0	0.76	27	55	18	3.0	0.0	3.0	No	3.0
dataset.columns															
Index(['ID', 'Age', 'Gender', 'Bedtime', 'Wakeup time', 'Sleep duration', 'Sleep efficiency', 'REM sleep percentage', 'Deep sleep percentage', 'Light sleep percentage', 'Awakenings', 'Caffeine consumption', 'Alcohol consumption', 'Smoking status', 'Exercise frequency'], dtype='object')															
import pickle															

```
In [4]: dataset.columns
```

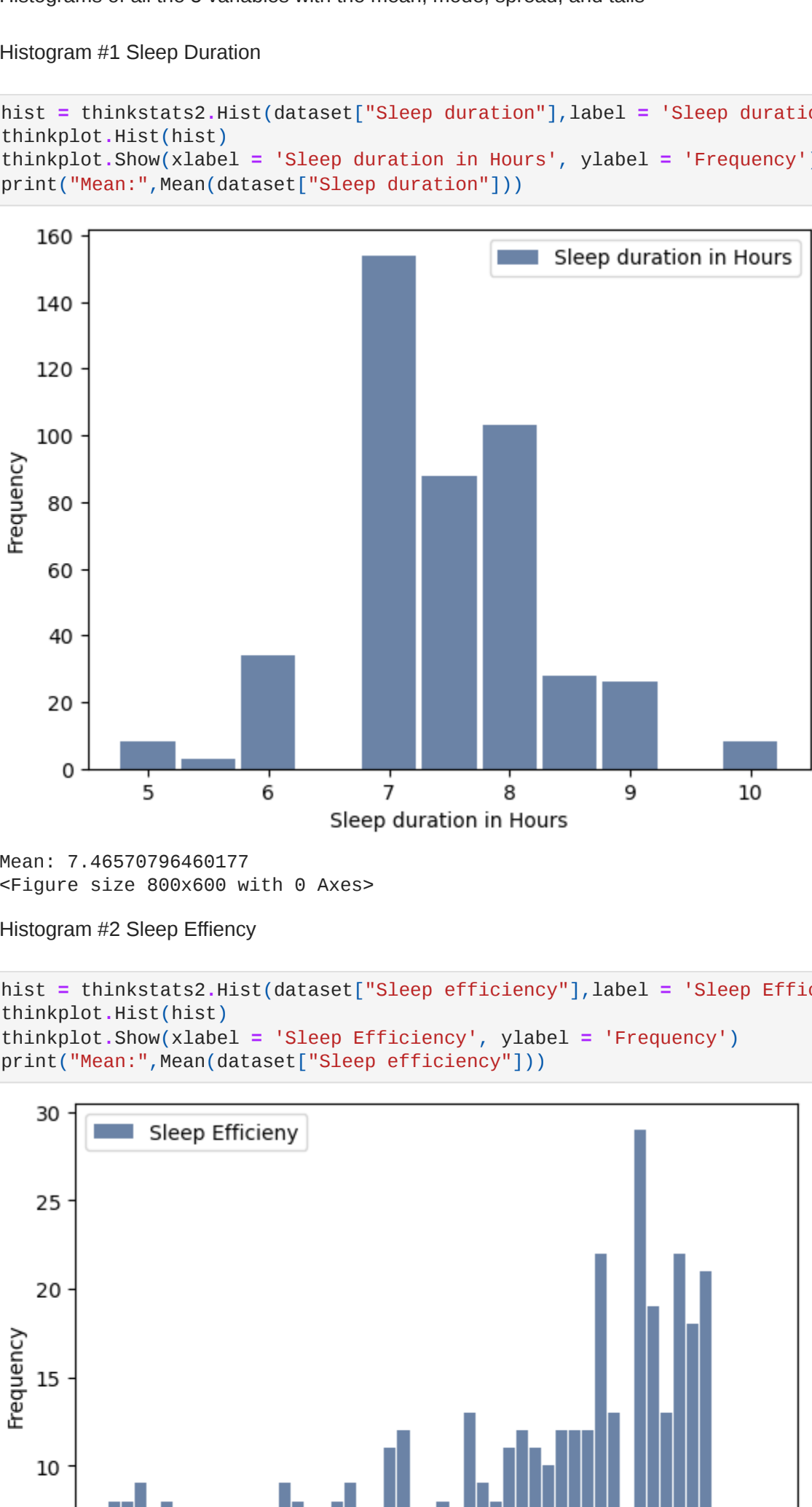
```
Out[4]: Index(['ID', 'Age', 'Gender', 'Bedtime', 'Wakeup time', 'Sleep duration', 'Sleep efficiency', 'REM sleep percentage', 'Deep sleep percentage', 'Light sleep percentage', 'Awakenings', 'Caffeine consumption', 'Alcohol consumption', 'Smoking status', 'Exercise frequency'], dtype='object')
```

```
In [5]: import thinkplot
import thinkstats2
import numpy as np
```

Histograms of all the 5 variables with the mean, mode, spread, and tails

Histogram #1 Sleep Duration

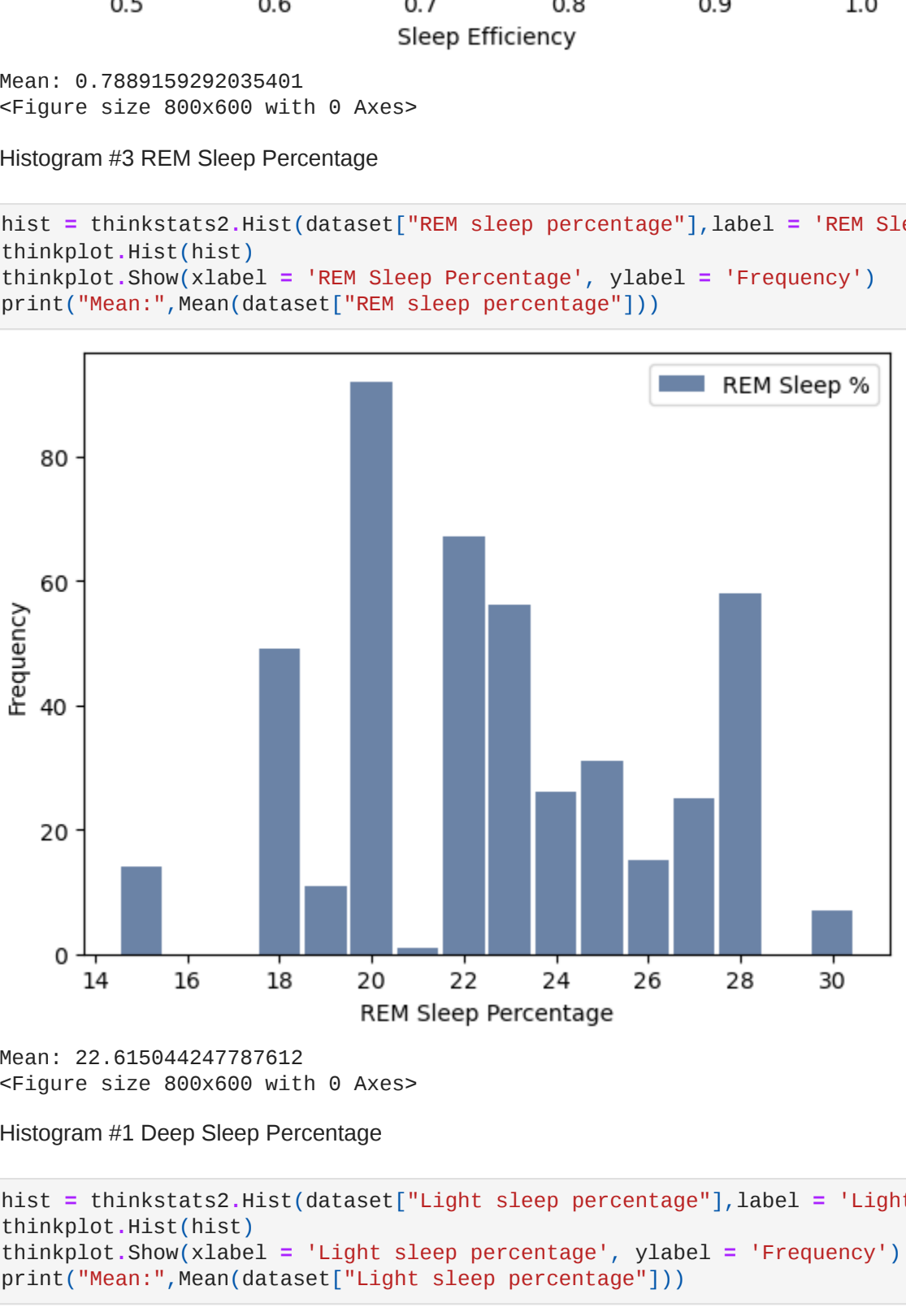
```
In [6]: hist = thinkstats2.Hist(dataset[["Sleep duration"]],label = 'Sleep duration in Hours')
thinkplot.Hist(hist)
thinkplot.Show(xlabel = 'Sleep duration in Hours', ylabel = 'Frequency')
print("Mean:",Mean(dataset[["Sleep duration"]]))
```



Mean: 7.46579796469177
<Figure size 800x600 with 0 Axes>

Histogram #2 Sleep Efficiency

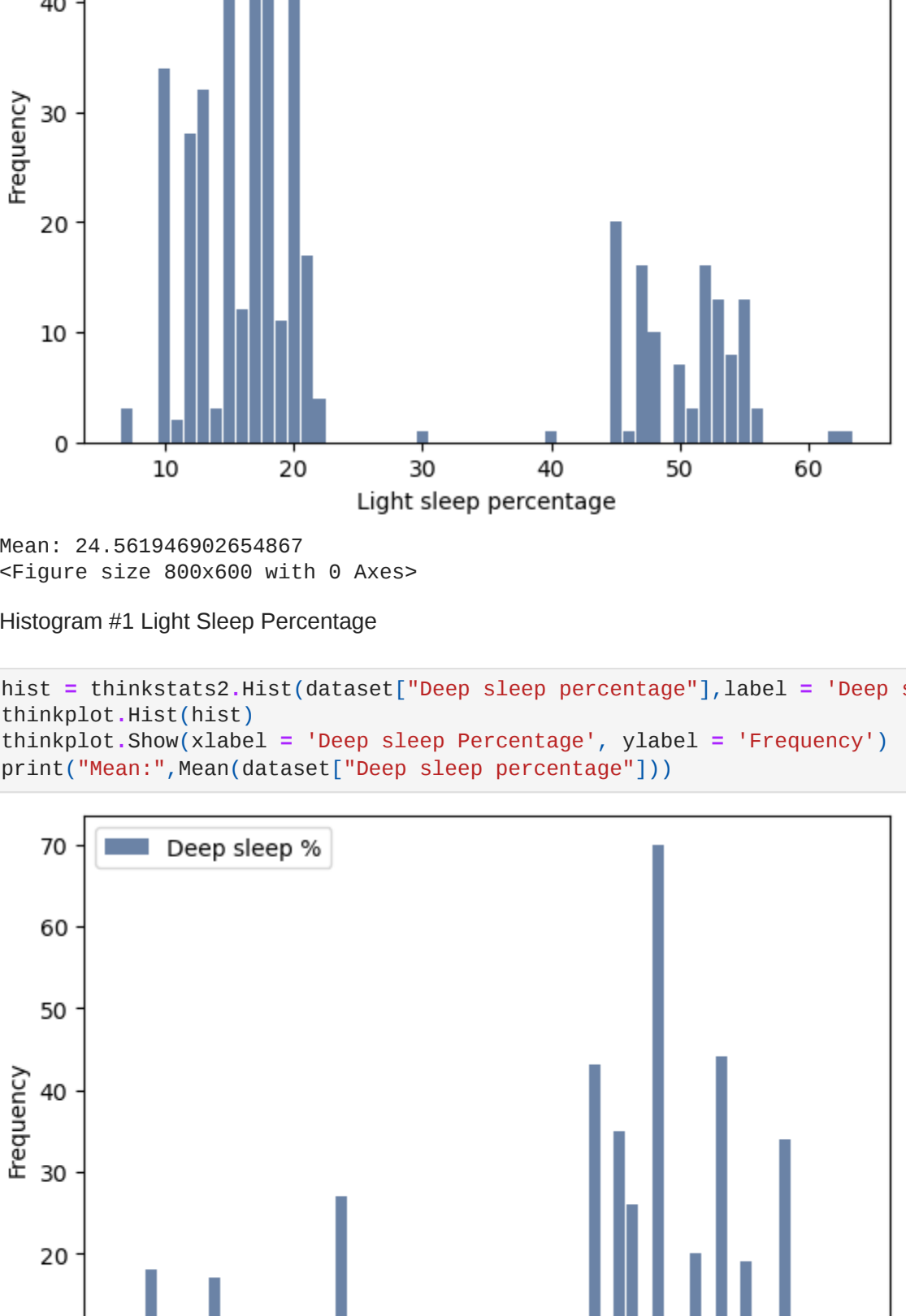
```
In [7]: hist = thinkstats2.Hist(dataset[["Sleep efficiency"]],label = 'Sleep Efficiency')
thinkplot.Hist(hist)
thinkplot.Show(xlabel = 'Sleep Efficiency', ylabel = 'Frequency')
print("Mean:",Mean(dataset[["Sleep efficiency"]]))
```



Mean: 0.7899159292935491
<Figure size 800x600 with 0 Axes>

Histogram #3 REM Sleep Percentage

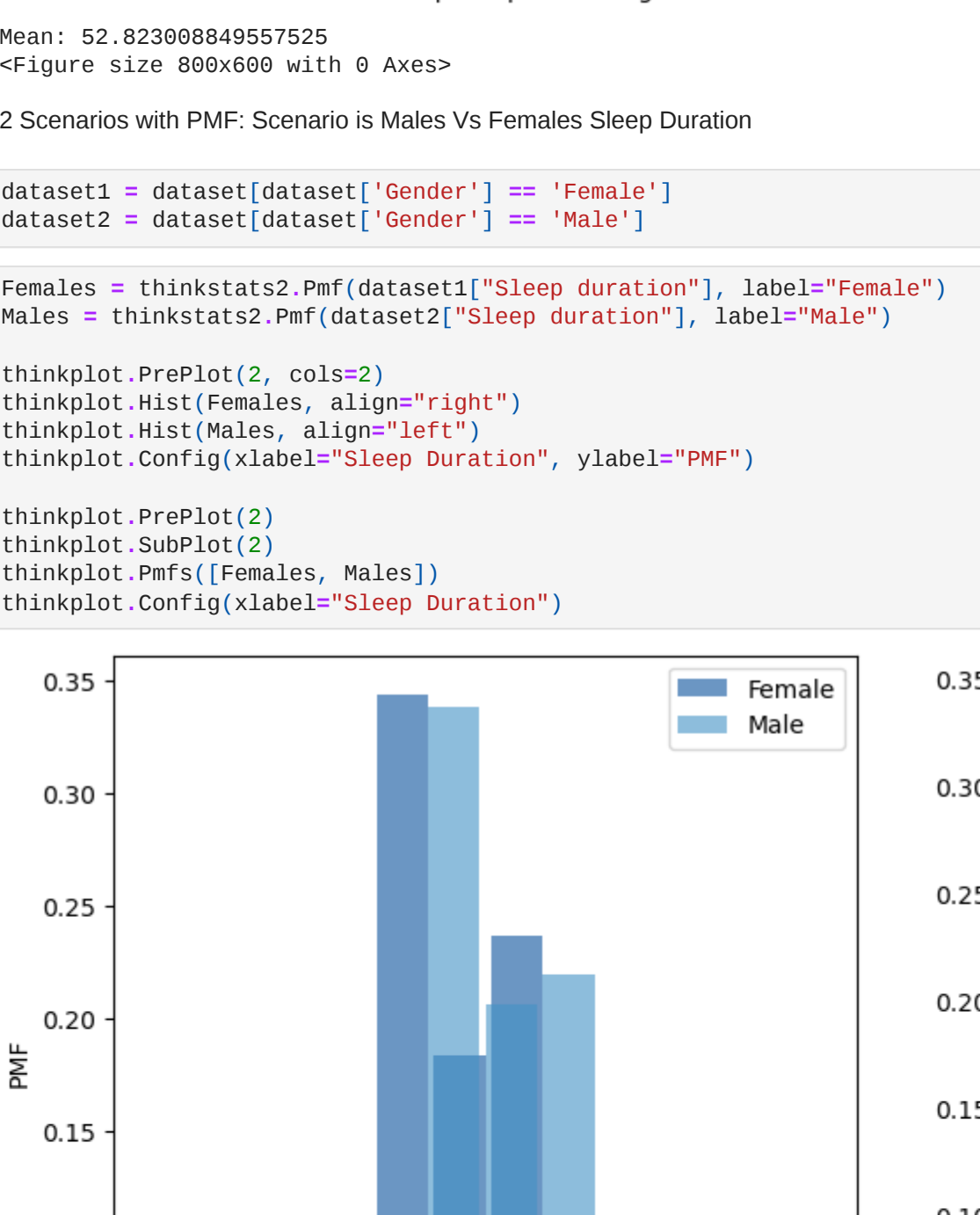
```
In [8]: hist = thinkstats2.Hist(dataset[["REM sleep percentage"]],label = 'REM Sleep %')
thinkplot.Hist(hist)
thinkplot.Show(xlabel = 'REM Sleep Percentage', ylabel = 'Frequency')
print("Mean:",Mean(dataset[["REM sleep percentage"]]))
```



Mean: 22.615844247787612
<Figure size 800x600 with 0 Axes>

Histogram #1 Light Sleep Percentage

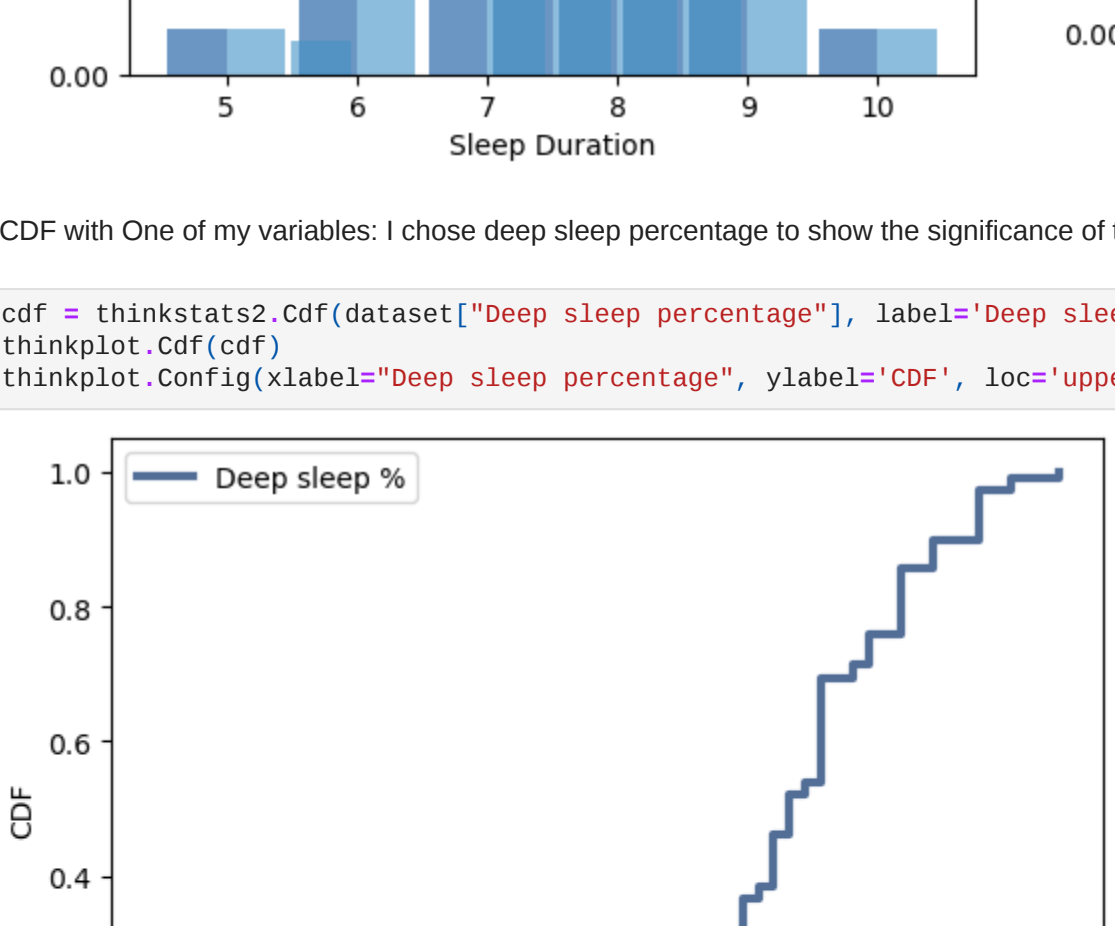
```
In [9]: hist = thinkstats2.Hist(dataset[["Light sleep percentage"]],label = 'Light sleep %')
thinkplot.Hist(hist)
thinkplot.Show(xlabel = 'Light sleep percentage', ylabel = 'Frequency')
print("Mean:",Mean(dataset[["Light sleep percentage"]]))
```



Mean: 24.561946902654867
<Figure size 800x600 with 0 Axes>

Histogram #1 Deep Sleep Percentage

```
In [10]: hist = thinkstats2.Hist(dataset[["Deep sleep percentage"]],label = 'Deep sleep %')
thinkplot.Hist(hist)
thinkplot.Show(xlabel = 'Deep sleep Percentage', ylabel = 'Frequency')
print("Mean:",Mean(dataset[["Deep sleep percentage"]]))
```

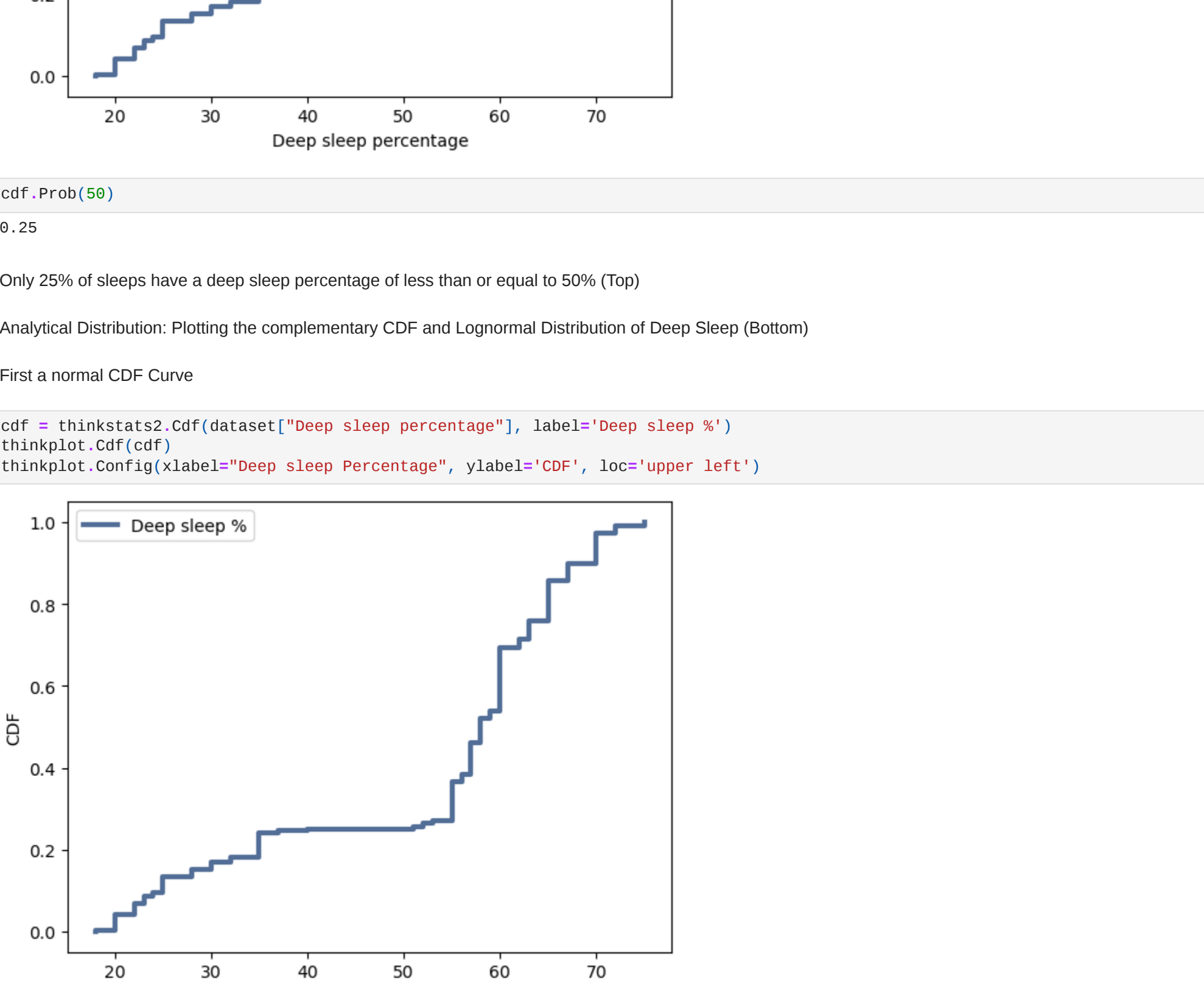


2 Scenarios with PMF: Scenario is Males Vs Females Sleep Duration

```
In [11]: dataset1 = dataset[dataset['Gender'] == 'Female']
dataset2 = dataset[dataset['Gender'] == 'Male']
```

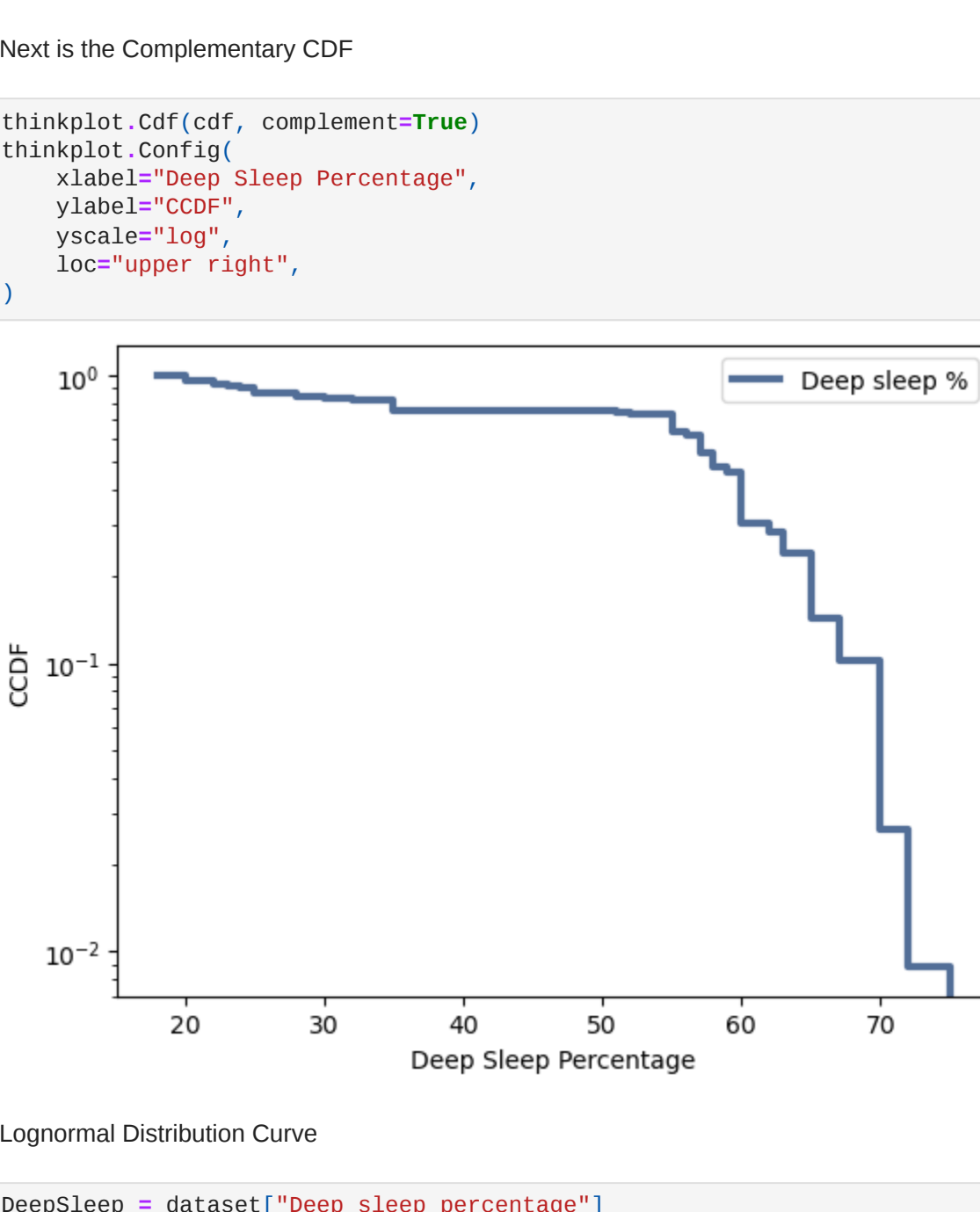
```
In [12]: Females = thinkstats2.Pmf(dataset1[["Sleep duration"]],label="Female")
Males = thinkstats2.Pmf(dataset2[["Sleep duration"]],label="Male")
```

```
thinkplot.PrePlot(2,cols=2)
thinkplot.Hist(Females,align="right")
thinkplot.Hist(Males,align="left")
thinkplot.Config(xlabel="Sleep Duration",ylabel="PMF")
thinkplot.SubPlot(2)
thinkplot.Pmf(Females,Males)
thinkplot.Config(xlabel="Sleep Duration")
```



CDF with One of my variables: I chose deep sleep percentage to show the significance of this variable.

```
In [13]: cdf = thinkstats2.Cdf(dataset[["Deep sleep percentage"]],label="Deep sleep %")
thinkplot.Cdf(cdf)
thinkplot.Config(xlabel="Deep sleep percentage",ylabel="CDF",loc="upper left")
```



cdf.Prob(50)

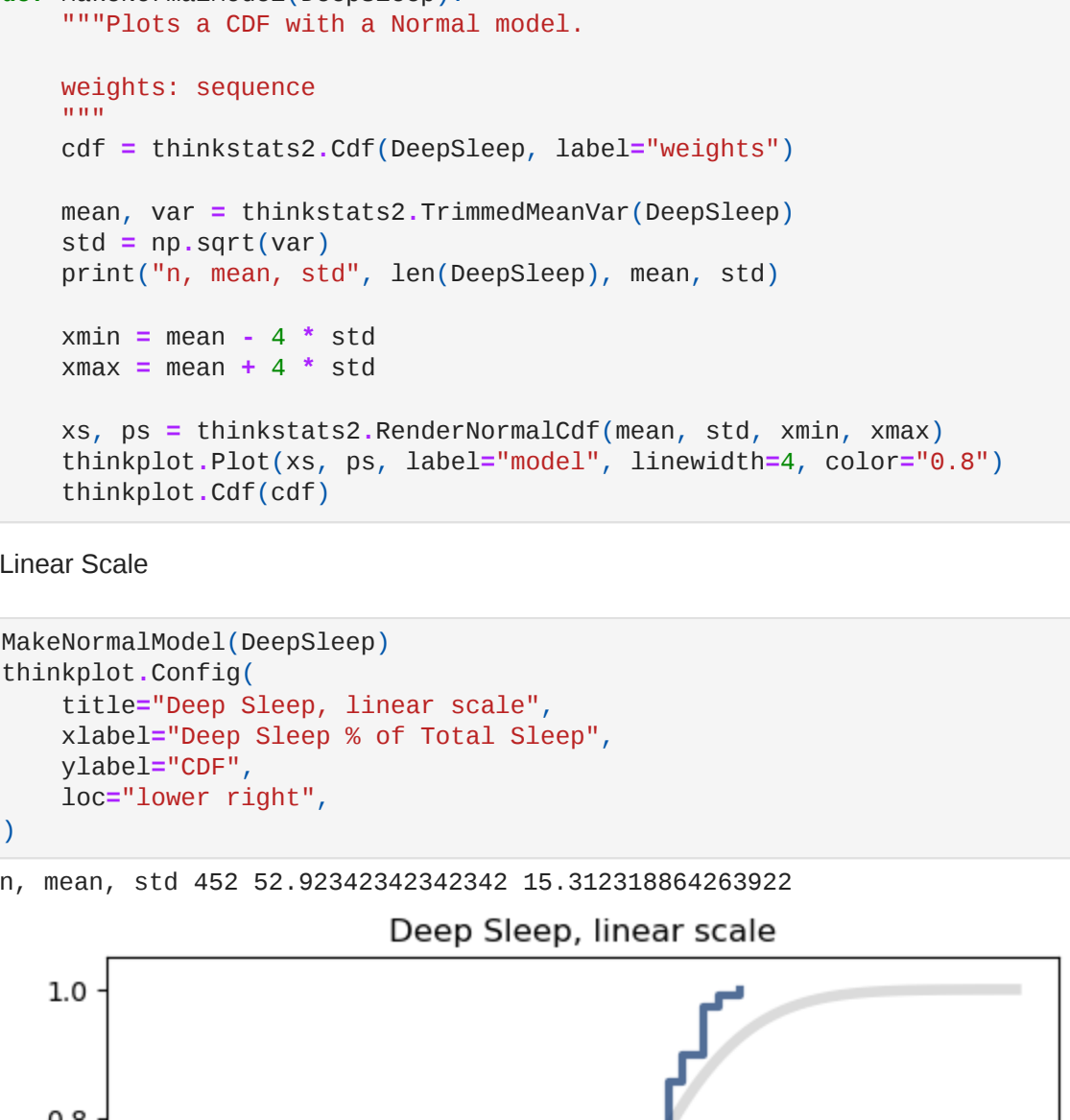
0.25

Only 25% of sleeps have a deep sleep percentage of less than or equal to 50% (Top)

Analytical Distribution: Plotting the complementary CDF and Lognormal Distribution of Deep Sleep (Bottom)

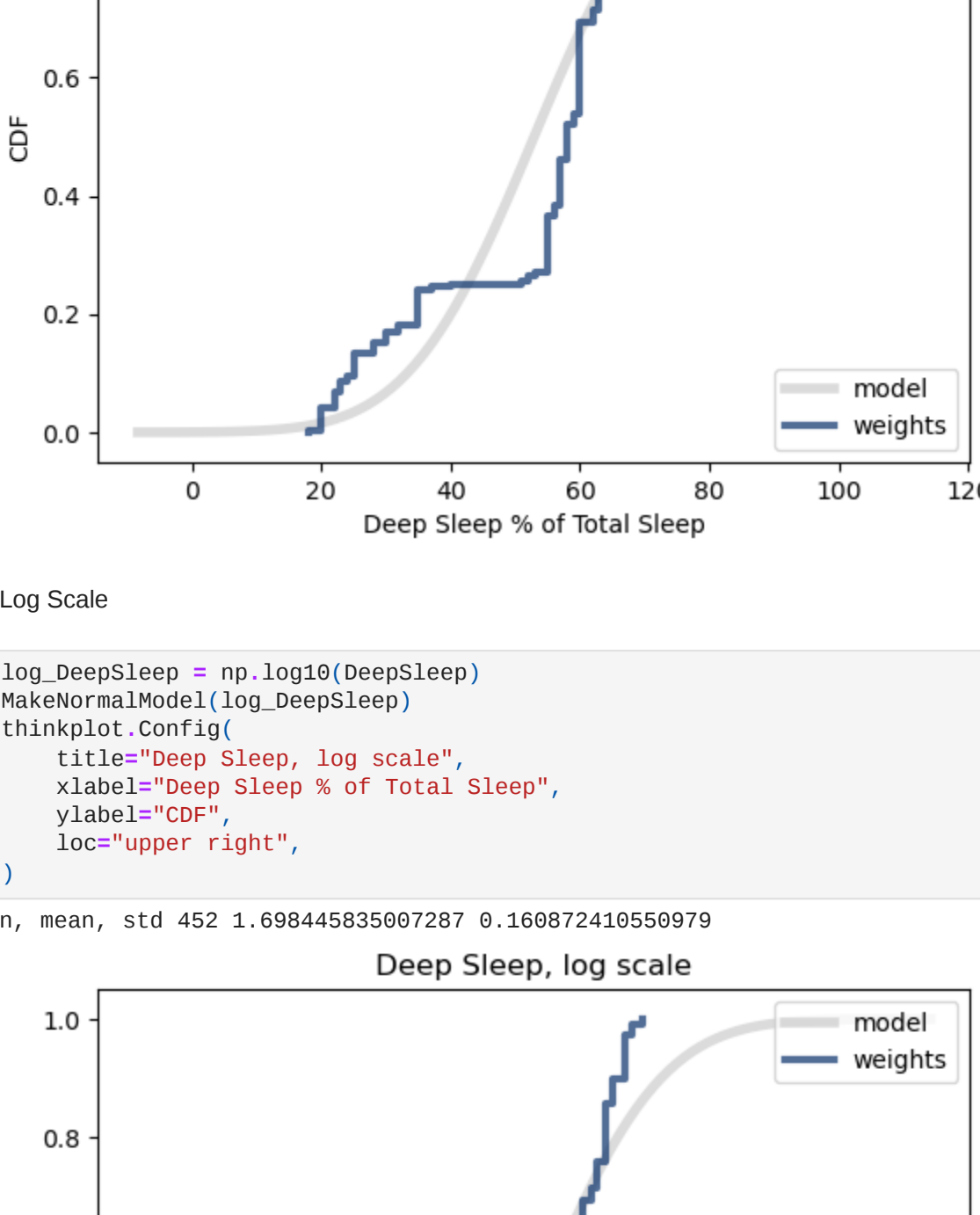
First a normal CDF Curve

```
In [15]: cdf = thinkstats2.Cdf(dataset[["Deep sleep percentage"]],label="Deep sleep %")
thinkplot.Cdf(cdf)
thinkplot.Config(xlabel="Deep sleep Percentage",ylabel="CDF",loc="upper left")
```



Next is the Complementary CDF

```
In [16]: MakeNormalModel(cdf,complement=True)
thinkplot.Config(
    xlabel="Deep Sleep Percentage",
    ylabel="CCDF",
    yscale="log",
    loc="upper right",
)
```



Lognormal Distribution Curve

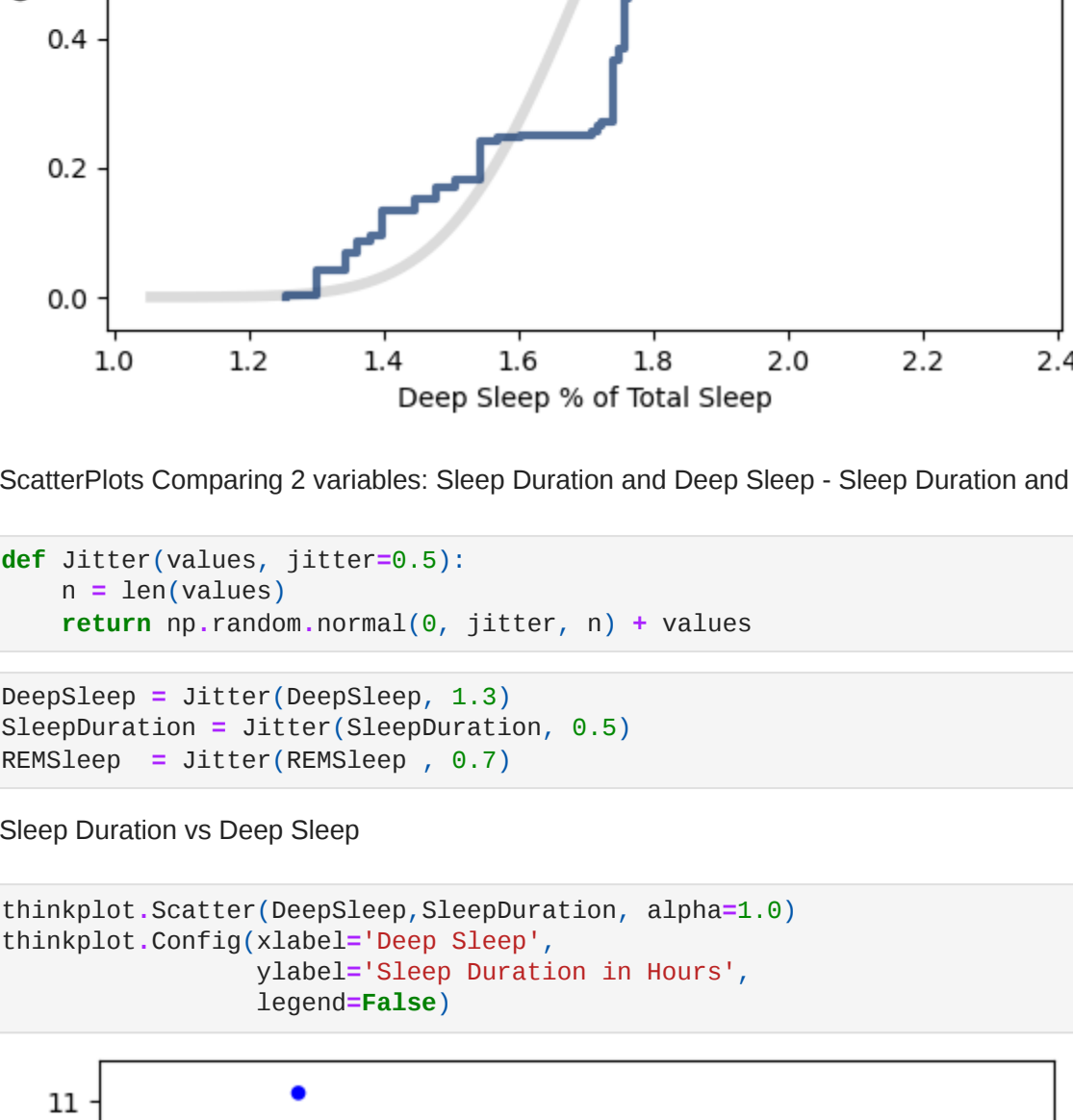
```
In [17]: DeepSleep = dataset[["Deep sleep percentage"]]
SleepDuration = dataset[["Sleep duration"]]
REMSleep = dataset[["REM sleep percentage"]]
```

```
In [18]: def MakeNormalModel(DeepSleep):
    """Plots a CDF with a Normal model.
    weights: sequence
    """
    cdf = thinkstats2.Cdf(DeepSleep, label="weights")
    mean, var = thinkstats2.TrimmedMeanVar(DeepSleep)
    std = np.sqrt(var)
    print("n, mean, std", len(DeepSleep), mean, std)
    xmin = mean - 4 * std
    xmax = mean + 4 * std
    xs, ps = thinkstats2.RenderNormalCdf(mean, std, xmin, xmax)
    thinkplot.Plot(xs, ps, label="model", linewidth=4, color="0.8")
    thinkplot.Cdf(cdf)
```

Linear Scale

```
In [19]: MakeNormalModel(DeepSleep)
thinkplot.Config(
    title="Deep Sleep, linear scale",
    xlabel="Deep Sleep % of Total Sleep",
    ylabel="CDF",
    loc="lower right",
)
```

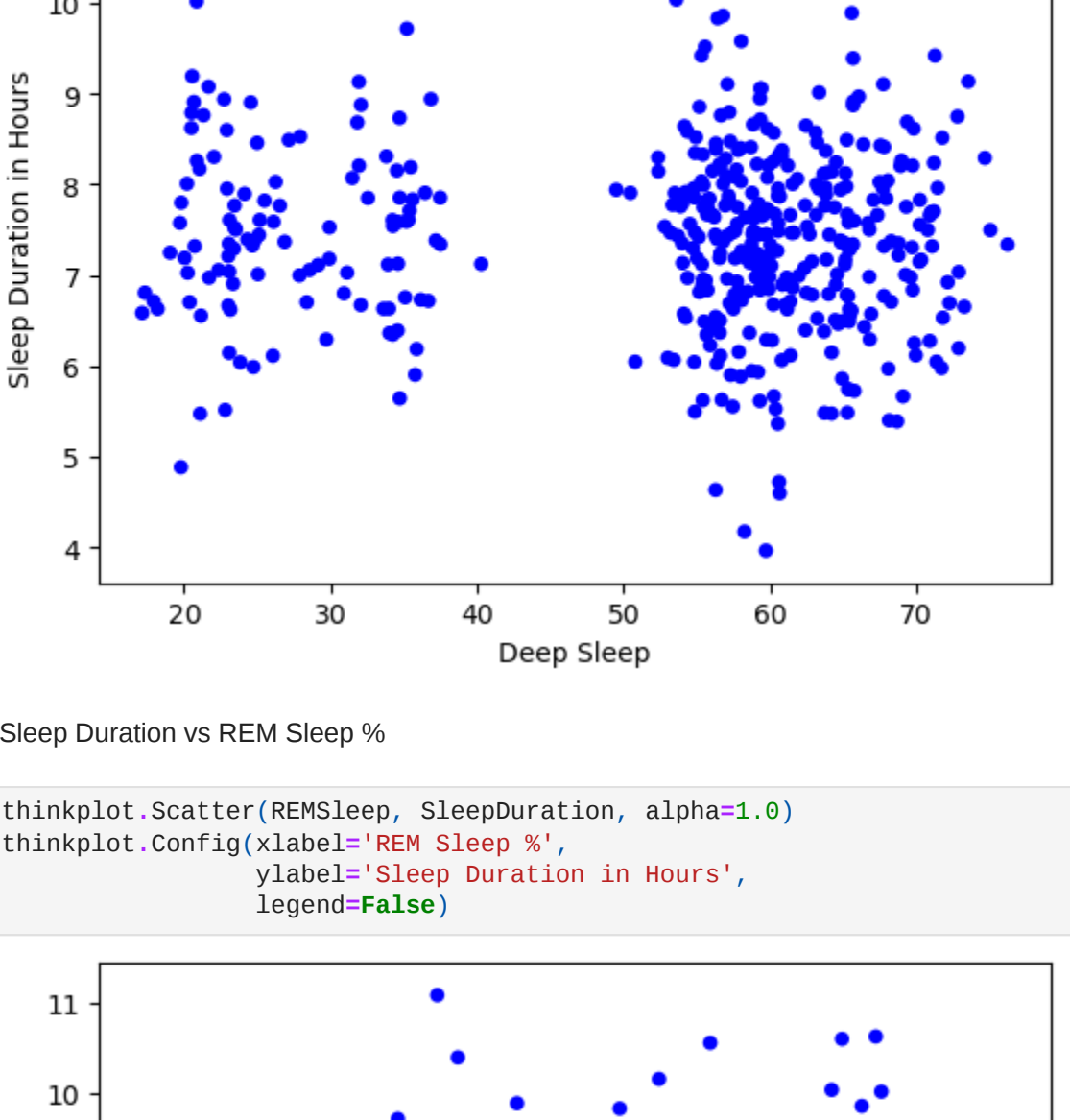
n, mean, std 452 52.92342342342342 15.312318864263922



Log Scale

```
In [20]: log_DeepSleep = np.log10(DeepSleep)
MakeNormalModel(log_DeepSleep)
thinkplot.Config(
    title="Deep Sleep, log scale",
    xlabel="Deep Sleep % of Total Sleep",
    ylabel="CDF",
    loc="upper right",
)
```

n, mean, std 452 1.698445835697287 0.166872418559979



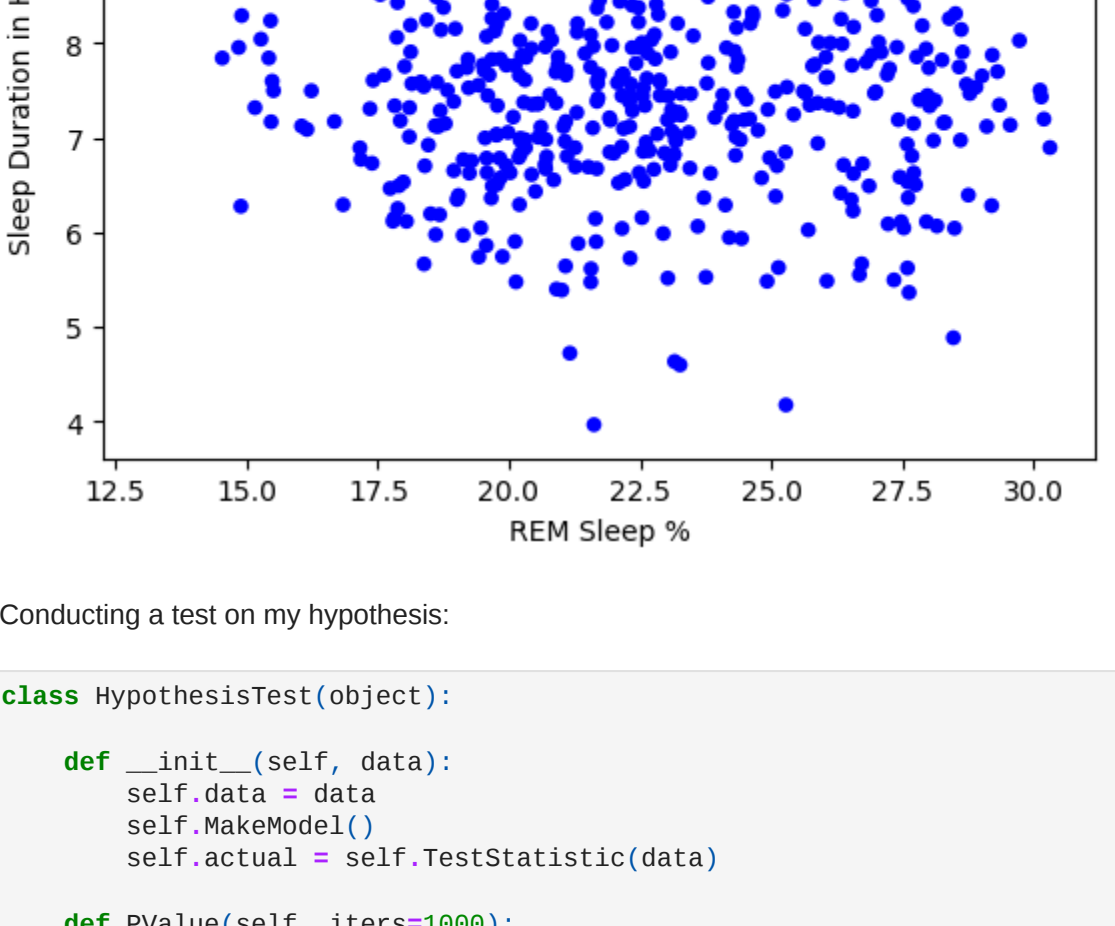
ScatterPlots Comparing 2 variables: Sleep Duration and Deep Sleep - Sleep Duration and REM Sleep %

```
In [21]: def Jitter(values, jitter=0.5):
    n = len(values)
    return np.random.normal(0, jitter, n) + values
```

```
In [22]: DeepSleep = Jitter(DeepSleep, 1.3)
SleepDuration = Jitter(SleepDuration, 0.5)
REMSleep = Jitter(REMSleep, 0.7)
```

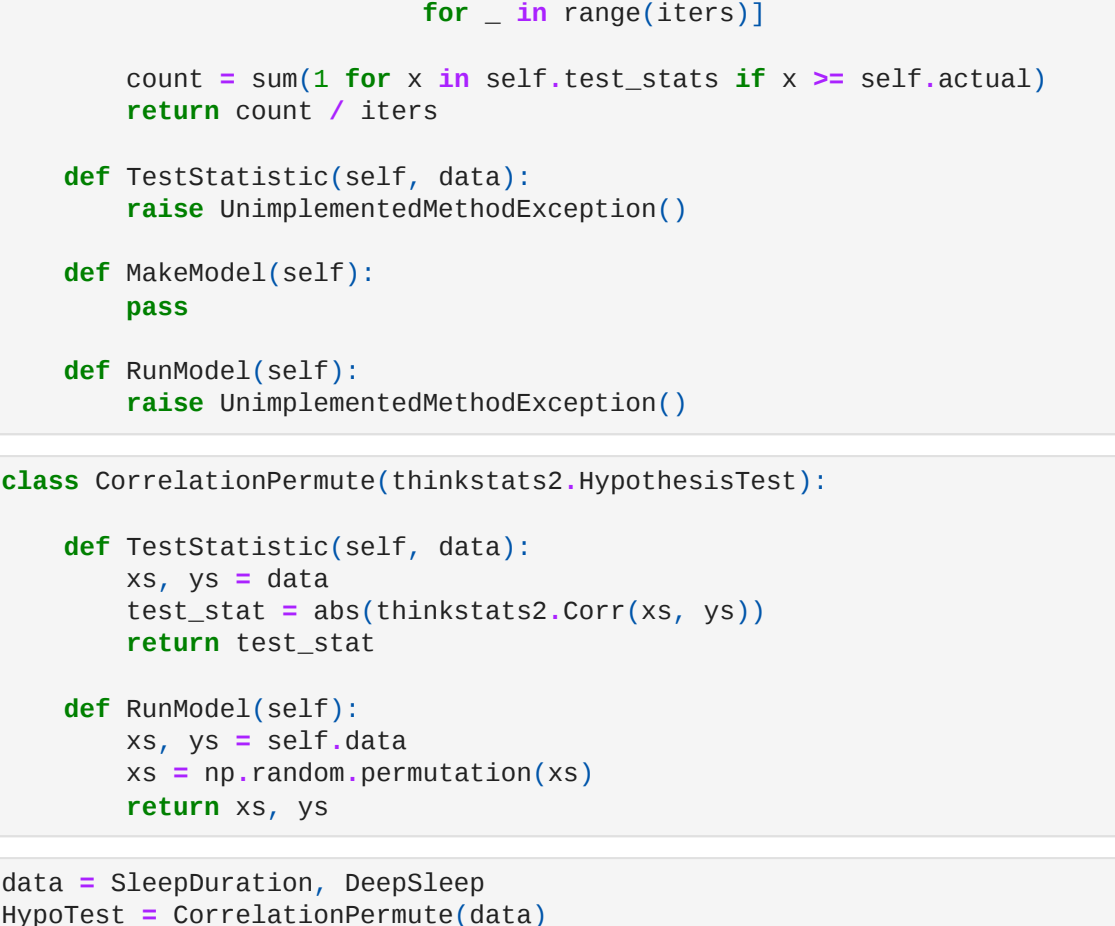
Sleep Duration vs Deep Sleep

```
In [23]: thinkplot.Scatter(DeepSleep,SleepDuration,alpha=1.0)
thinkplot.Config(xlabel="Deep Sleep",
    ylabel="Sleep Duration in Hours",
    legend=False)
```



Sleep Duration vs REM Sleep %

```
In [24]: thinkplot.Scatter(REMSleep,SleepDuration,alpha=1.0)
thinkplot.Config(xlabel="REM Sleep %",
    ylabel="Sleep Duration in Hours",
    legend=False)
```



Conducting a test on my hypothesis:

```
In [25]: class HypothesisTest(object):
    def __init__(self, data):
        self.data = data
        self.MakeModel()
        self.actual = self.TestStatistic(data)

    def PValue(self, s=1000):
        self.test_stats = [self.TestStatistic(self.RunModel())
            for _ in range(1000)]
        count = sum(1 for x in self.test_stats if x >= self.actual)
        return count / 1000

    def TestStatistic(self, data):
        raise NotImplementedError()

    def MakeModel(self):
        pass

    def RunModel(self):
        raise NotImplementedError()
```

```
In [26]: class CorrelationPermute(thinkstats2.HypothesisTest):
    def TestStatistic(self, data):
        xs, ys = data
        test_stat = abs(thinkstats2.Corr(xs, ys))
        return test_stat

    def RunModel(self):
        xs, ys = self.data
        xs = np.random.permutation(xs)
        return xs, ys
```

```
In [27]: data = SleepDuration, DeepSleep
HypoTest = CorrelationPermute(data)
pvalue = HypoTest.PValue()
```

0.108

```
Out[27]: 0.108
```

```
In [28]: def LeastSquares(xs, ys):
    meanx, varx = MeanVar(xs)
    meany = Mean(ys)
    slope = Cov(xs, ys, meanx, meany) / varx
    inter = meany - slope * meanx
```

LeastSquares(SleepDuration, DeepSleep)

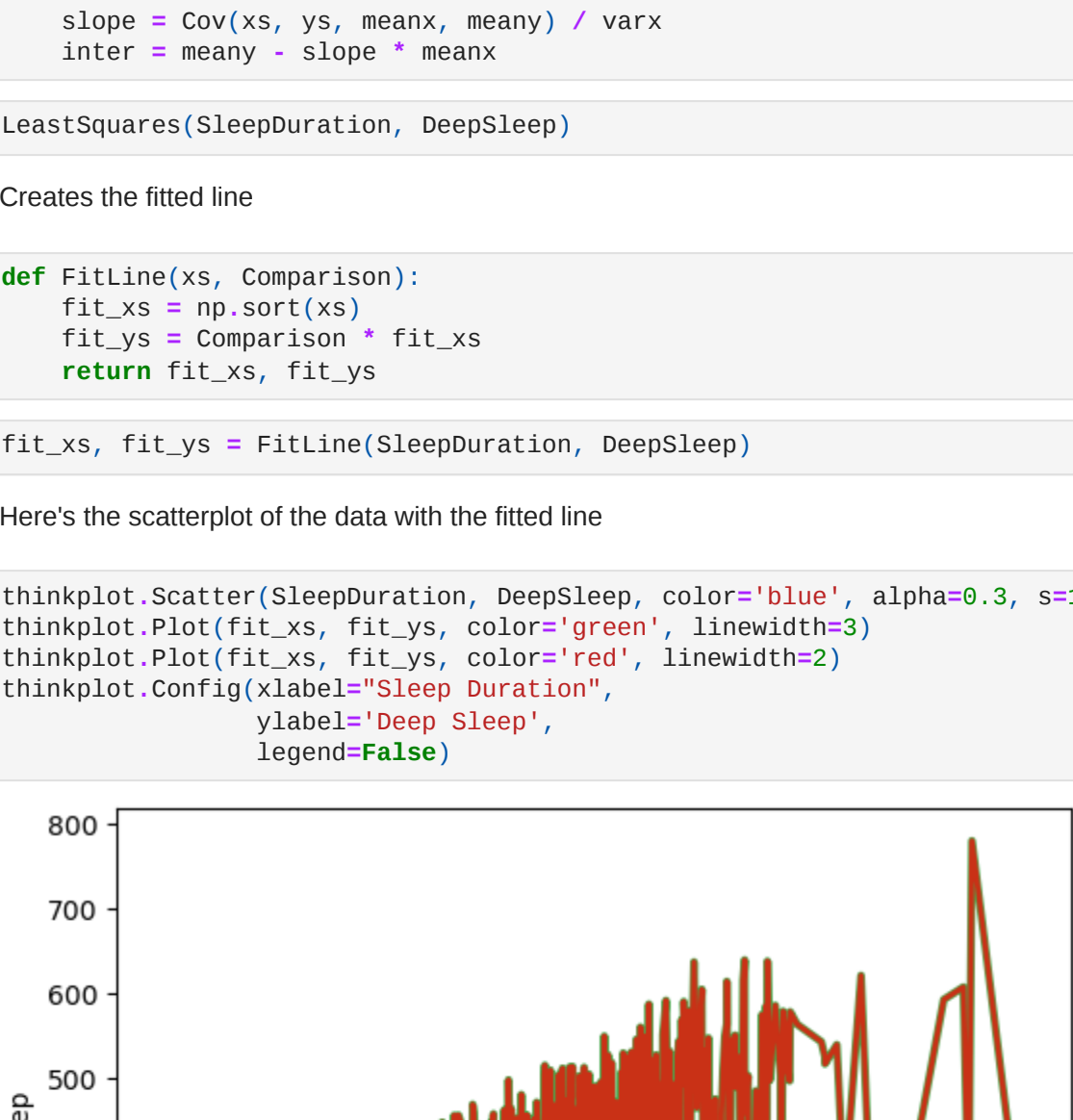
Creates the fitted line

```
In [30]: def FitLine(xs, Comparison):
    fit_xs = np.sort(xs)
    fit_ys = Comparison * fit_xs
    return fit_xs, fit_ys
```

```
In [31]: fit_xs, fit_ys = FitLine(SleepDuration, DeepSleep)
```

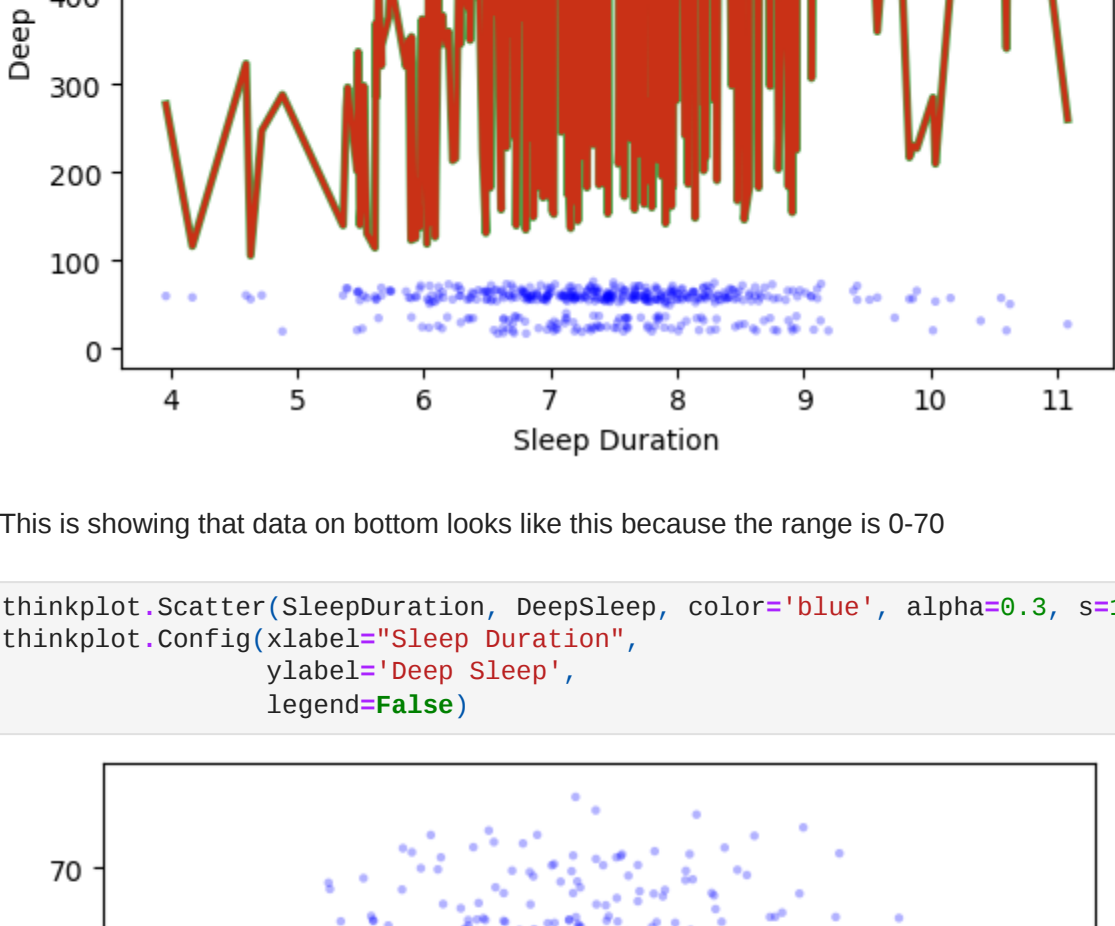
Here's the scatterplot of the data with the fitted line

```
In [32]: thinkplot.Scatter(SleepDuration, DeepSleep, color="blue", alpha=0.3, s=10)
thinkplot.Plot(fit_xs, fit_ys, color="green", linewidth=3)
thinkplot.Plot(fit_xs, fit_ys, color="red", linewidth=2)
thinkplot.Config(xlabel="Sleep Duration",
    ylabel="Deep Sleep",
    legend=False)
```



This is showing that data on bottom looks like this because the range is 0-70

```
In [33]: thinkplot.Scatter(SleepDuration, DeepSleep, color="blue", alpha=0.3, s=18)
thinkplot.Config(xlabel="Sleep Duration",
    ylabel="Deep Sleep",
    legend=False)
```



```
In [ ]:
```