Does demographical factors (age, gender, health score, test preparedness, and family relationship) effect overall test scores?

In [149]:
```
import os
import numpy as np
import pandas as pd
import thinkplot
import thinkstats2
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
```

In [153]:
```
# Utilize to import csv file

student_data = pd.read_csv(r"C:\Users\qvant\Desktop\JNotebook\ThinkStats2-master\
```

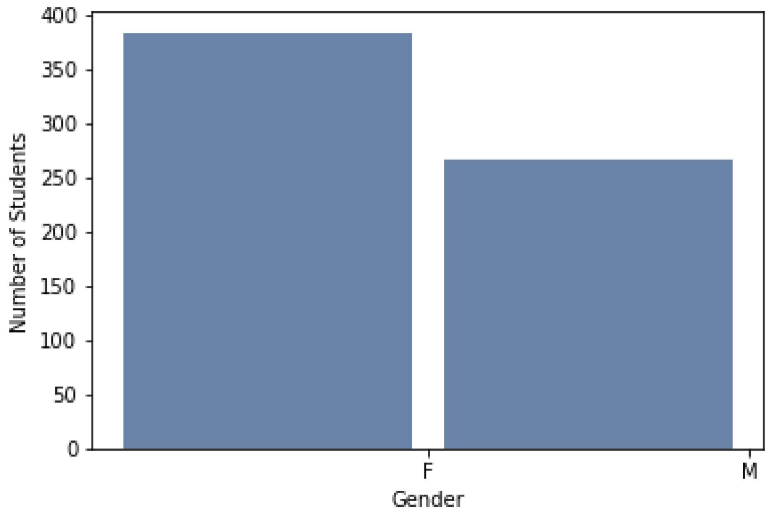
In [154]:
```
# Utilize to view column names

student_data.head()
```

Out[154]:

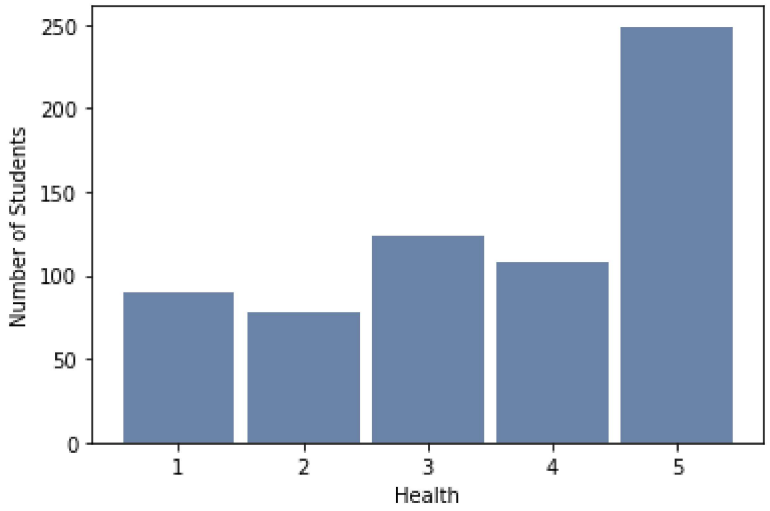
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	freetime	goo
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	3	
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	
4	GP	F	16	U	GT3	T	3	3	other	other	...	3	

5 rows × 34 columns

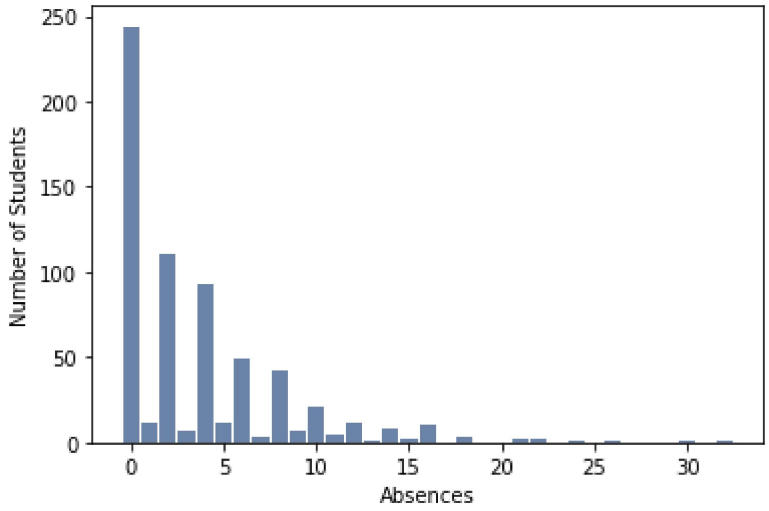
In [155]:
```
# Gender [No Outliers Found + Did not need to Omit any Data]

histogram_gender = thinkstats2.Hist(student_data.sex)
thinkplot.Hist(histogram_gender)
thinkplot.Config(xlabel='Gender', ylabel='Number of Students')
```



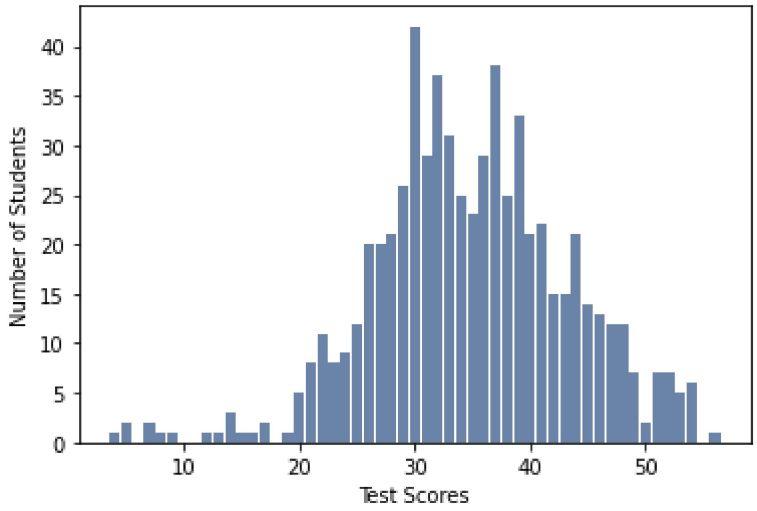
In [157]:
```
# Health [No Outliers Found + Did not need to Omit any Data]

histogram_health = thinkstats2.Hist(student_data.health)
thinkplot.Hist(histogram_health)
thinkplot.Config(xlabel='Health', ylabel='Number of Students')
```



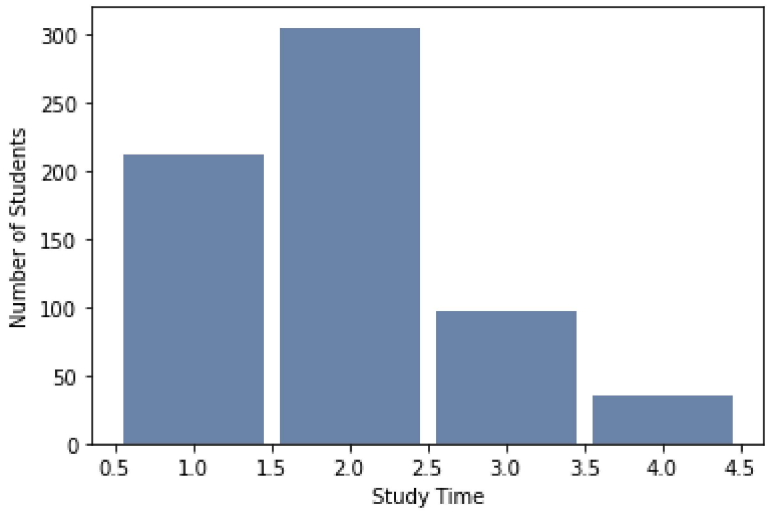
In [252]:
```
# Absences - There are outliers where students have more than 20 absences but I c
# if we can identify if its a specific variable that are driving this.

histogram_absences = thinkstats2.Hist(student_data.absences)
thinkplot.Hist(histogram_absences)
thinkplot.Config(xlabel='Absences', ylabel='Number of Students')
```



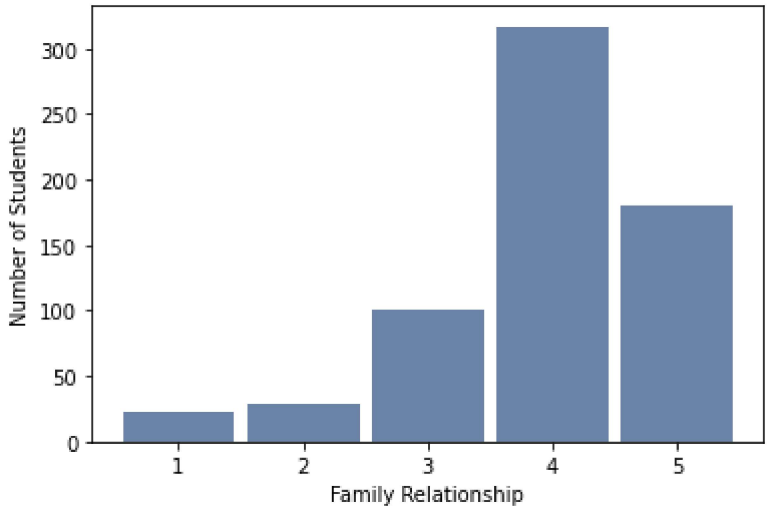
In [159]:
```
# Test Scores [A bulk of the data is bell-shaped with semi normal distribution. T
# test scores lower than 10. I choose to include this as it can show a significar

histogram_score = thinkstats2.Hist(student_data.total_score)
thinkplot.Hist(histogram_score)
thinkplot.Config(xlabel='Test Scores', ylabel='Number of Students')
```



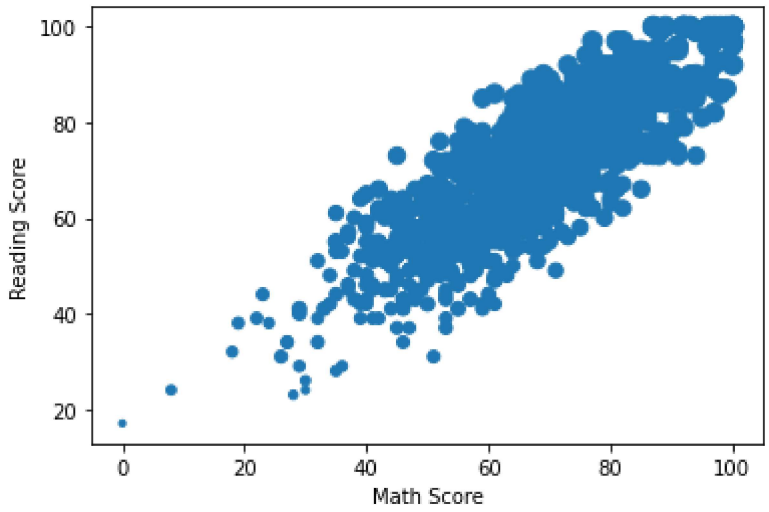
In [160]:
```
# Test Studytime [No Outliers Found + Did not need to Omit any Data]

histogram_studytime = thinkstats2.Hist(student_data.studytime)
thinkplot.Hist(histogram_studytime)
thinkplot.Config(xlabel='Study Time', ylabel='Number of Students')
```



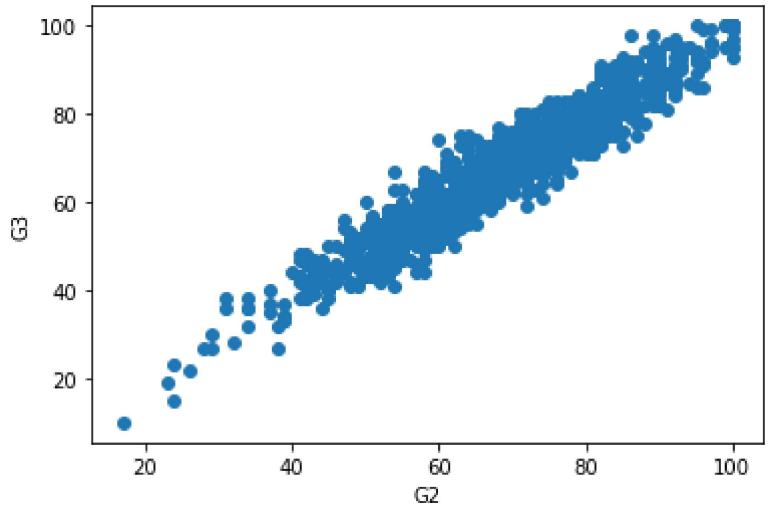
In [161]:
```
# Family Relationship [No Outliers Found + Did not need to Omit any Data]

histogram_famrel = thinkstats2.Hist(student_data.famrel)
thinkplot.Hist(histogram_famrel)
thinkplot.Config(xlabel='Family Relationship', ylabel='Number of Students')
```



In [162]:
```
# Math and Reading Scores [Outliers Found but they are significant + Did not need

G1 = student_data.G1
G2 = student_data.G2

plt.scatter(math, reading, writing)
plt.xlabel("G1")
plt.ylabel("G2");
```



In [163]:
```
# Reading and Writing Scores [Outliers Found but they are significant + Did not r

G2 = student_data.G2
G3 = student_data.G3

plt.scatter(reading, writing)
plt.xlabel("G2")
plt.ylabel("G3");
```



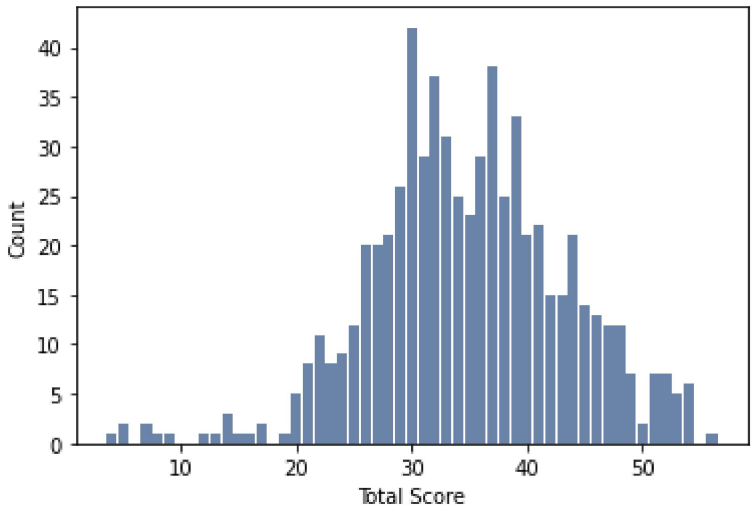
In [164]:
```
# There are outliers found but they are significant to the data. I can remove the
# provide a very biased result that no student can score a 0, or have a total sc
# variables chosen, lack of a few can mean a very low scoring total.
```

In [165]:
```
student_data.describe()
```

Out[165]:

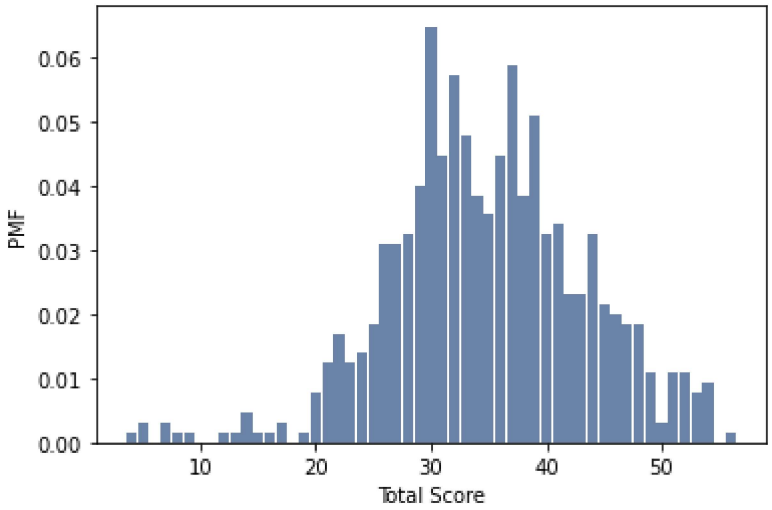
	age	Medu	Fedu	traveltime	studytime	failures	famrel	free
count	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.00
mean	16.744222	2.514638	2.306626	1.568567	1.930663	0.221880	3.930663	3.18
std	1.218138	1.134552	1.099931	0.748660	0.829510	0.593235	0.955717	1.05
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.00
25%	16.000000	2.000000	1.000000	1.000000	1.000000	0.000000	4.000000	3.00
50%	17.000000	2.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.00
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.00
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.00

In [166]:
```
#PMF of Total Score

hist = thinkstats2.Hist(student_data.total_score, label='Total Score')
thinkplot.Hist(hist)
thinkplot.Config(xlabel='Total Score', ylabel='Count')
```

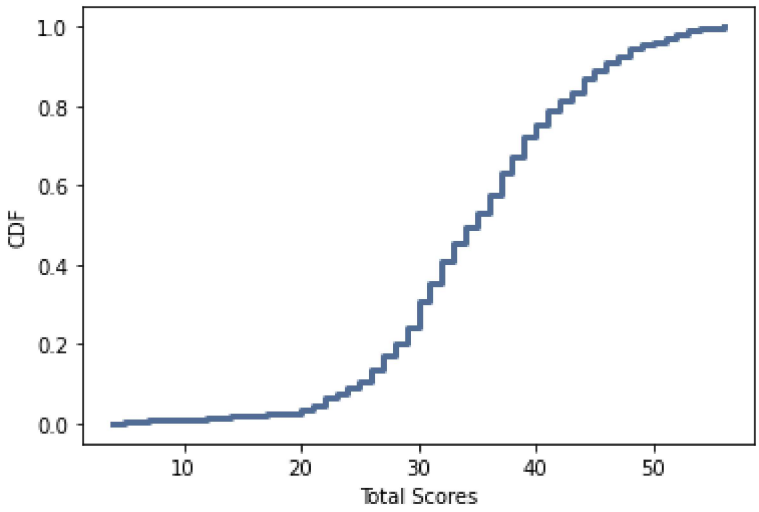


In [168]:
```
n = hist.Total()
pmf = hist.Copy()
for x, freq in hist.Items():
    pmf[x] = freq / n
```

In [169]:
```
thinkplot.Hist(pmf)
thinkplot.Config(xlabel='Total Score', ylabel='PMF')
```

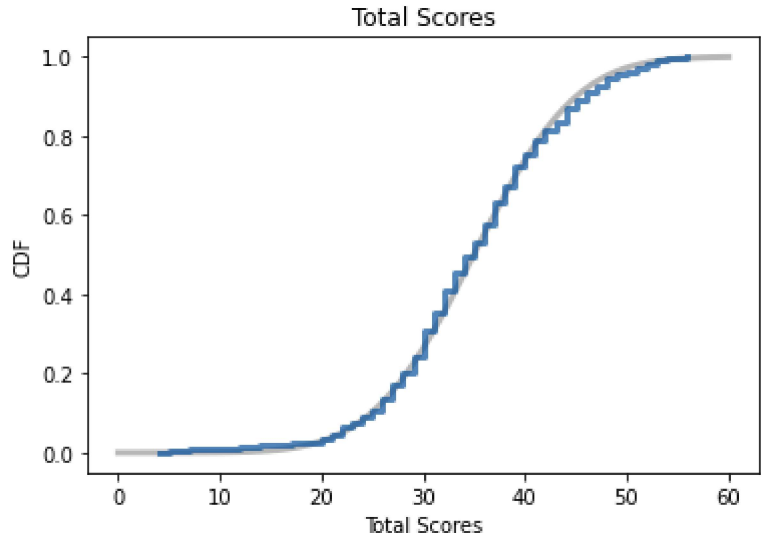


In [170]:
```
cdf = thinkstats2.Cdf(student_data.total_score, label='Test Scores')
thinkplot.Cdf(cdf)
thinkplot.Config(xlabel='Total Scores', ylabel='CDF', loc='upper left')
```



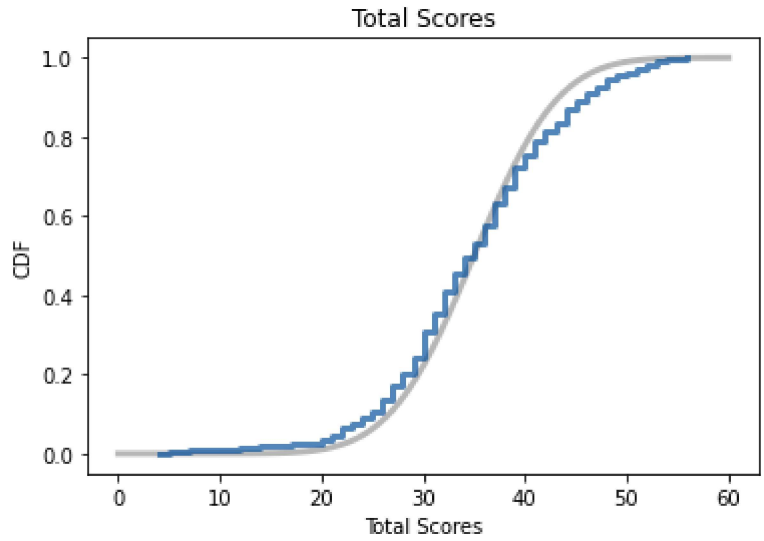
In [174]:
```
# Normal Distribution w/ p = 0.01

mu, var = thinkstats2.TrimmedMeanVar(student_data.total_score, p=0.01)
print('Mean, Var', mu, var)

# plot the model
sigma = np.sqrt(var)
print('Sigma', sigma)
xs, ps = thinkstats2.RenderNormalCdf(mu, sigma, low=0, high=60)

thinkplot.Plot(xs, ps, label='model', color='0.6')

# plot the data
cdf = thinkstats2.Cdf(student_data.total_score, label='data')

thinkplot.PrePlot(1)
thinkplot.Cdf(cdf)
thinkplot.Config(title='Total Scores',
                 xlabel='Total Scores',
                 ylabel='CDF')
```

```
Mean, Var 34.96389324960754 62.04736192267026
Sigma 7.877014784972176
```



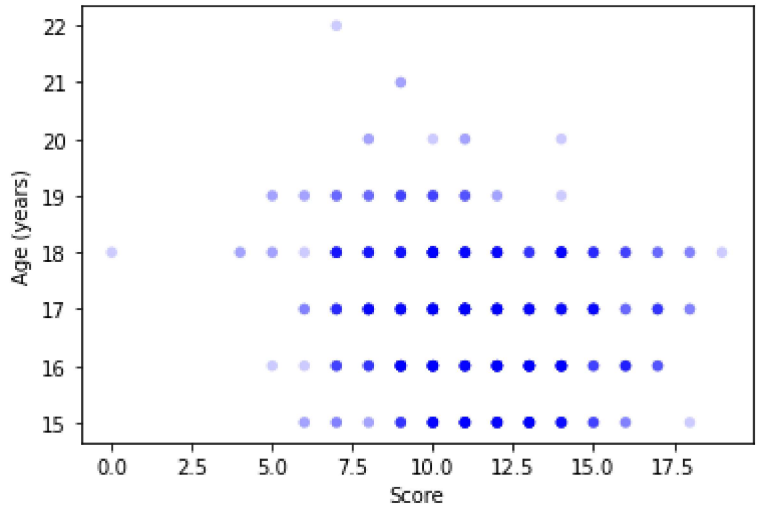
In [175]:
```
# Normal Distribution w/ p = 0.05

mu, var = thinkstats2.TrimmedMeanVar(student_data.total_score, p=0.05)
print('Mean, Var', mu, var)

# plot the model
sigma = np.sqrt(var)
print('Sigma', sigma)
xs, ps = thinkstats2.RenderNormalCdf(mu, sigma, low=0, high=60)

thinkplot.Plot(xs, ps, label='model', color='0.6')

# plot the data
cdf = thinkstats2.Cdf(student_data.total_score, label='data')

thinkplot.PrePlot(1)
thinkplot.Cdf(cdf)
thinkplot.Config(title='Total Scores',
                 xlabel='Total Scores',
                 ylabel='CDF')
```

```
Mean, Var 34.98119658119658 42.37058660238148
Sigma 6.509269283289905
```



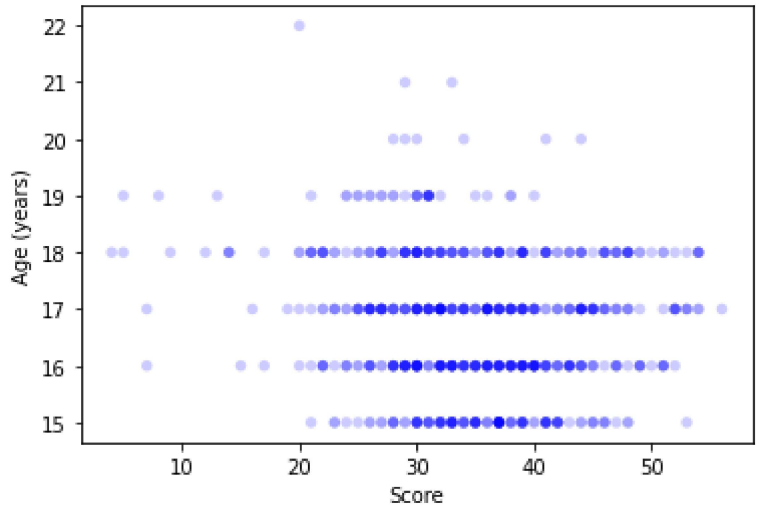
In [183]:
```
# ScatterPlot #1 G1 Scores

G1 = student_data.G1
age = student_data.age
```

In [184]:
```
thinkplot.Scatter(G1, age)
thinkplot.Config(xlabel='Score',
                 ylabel='Age (years)',
                 legend=False)
```



In [186]:
```
# ScatterPlot #2 Total Scores (G1, G2 and G3)

score = student_data.total_score
age2 = student_data.age
```

In [187]:
```
thinkplot.Scatter(score, age2)
thinkplot.Config(xlabel='Score',
                 ylabel='Age (years)',
                 legend=False)
```



In [188]:
```
def Cov(xs, ys, meanx=None, meany=None):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    if meanx is None:
        meanx = np.mean(xs)
    if meany is None:
        meany = np.mean(ys)

    cov = np.dot(xs-meanx, ys-meany) / len(xs)
    return cov
```

In [191]:
```
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 [192]:
```
Corr(age2, score)
```

Out[192]:
```
-0.13349865498491353
```

In [193]:
```
np.corrcoef(age2, score)
```

Out[193]:
```
array([[ 1.        , -0.13349865],
       [-0.13349865,  1.        ]])
```

In [194]:
```
def SpearmanCorr(xs, ys):
    xranks = pd.Series(xs).rank()
    yranks = pd.Series(ys).rank()
    return Corr(xranks, yranks)
```

In [195]:
```
SpearmanCorr(age2, score)
```

Out[195]:
```
-0.12074693513128194
```

In [196]:
```
def SpearmanCorr(xs, ys):
    xs = pd.Series(xs)
    ys = pd.Series(ys)
    return xs.corr(ys, method='spearman')
```

In [197]:
```
SpearmanCorr(age2, score)
```

Out[197]:
```
-0.12074693513128194
```

In [238]:
```
from thinkstats2 import Mean, MeanVar, Var, Std, Cov

def LeastSquares(xs, ys):
    meanx, varx = MeanVar(xs)
    meany = Mean(ys)

    slope = Cov(xs, ys, meanx, meany) / varx
    inter = meany - slope * meanx

    return inter, slope
```

In [239]:
```
inter, slope = LeastSquares(age, score)
inter, slope
```

Out[239]: (50.47317960387155, -0.9315444522787001)

In [240]:
```
inter + slope * 25
```

Out[240]: 27.184568296904047

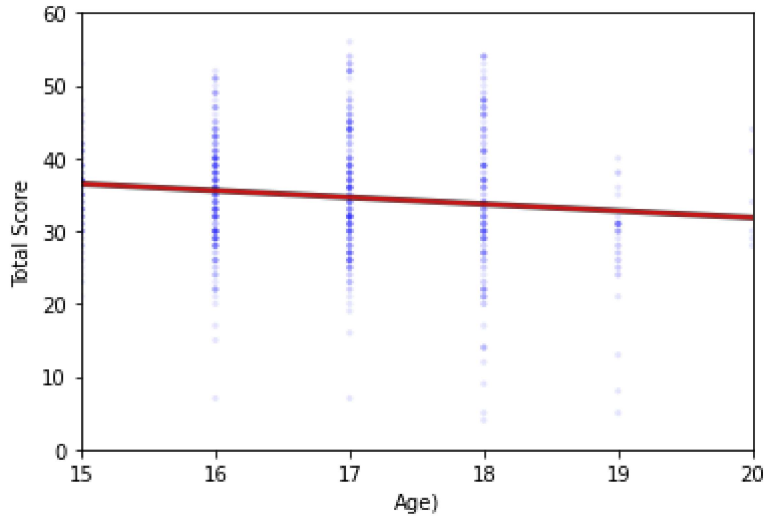
In [241]:
```
slope * 10
```

Out[241]: -9.315444522787

In [242]:
```
def FitLine(xs, inter, slope):
    fit_xs = np.sort(xs)
    fit_ys = inter + slope * fit_xs
    return fit_xs, fit_ys
```

In [244]:
```
fit_xs, fit_ys = FitLine(age, inter, slope)
```

In [251]:
```
thinkplot.Scatter(age, score, color='blue', alpha=0.1, s=10)
thinkplot.Plot(fit_xs, fit_ys, color='black', linewidth=3)
thinkplot.Plot(fit_xs, fit_ys, color='red', linewidth=2)
thinkplot.Config(xlabel="Age)",
                 ylabel='Total Score',
                 axis=[15, 20, 0, 60],
                 legend=False)
```



In [ ]: