Python Matplotlib Data Visualization Practice Exercises

Create a simple line chart in Python Matplotlib

In [9]:
```
import matplotlib.pyplot as plt
```

In [10]:
```
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [300.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3]
```

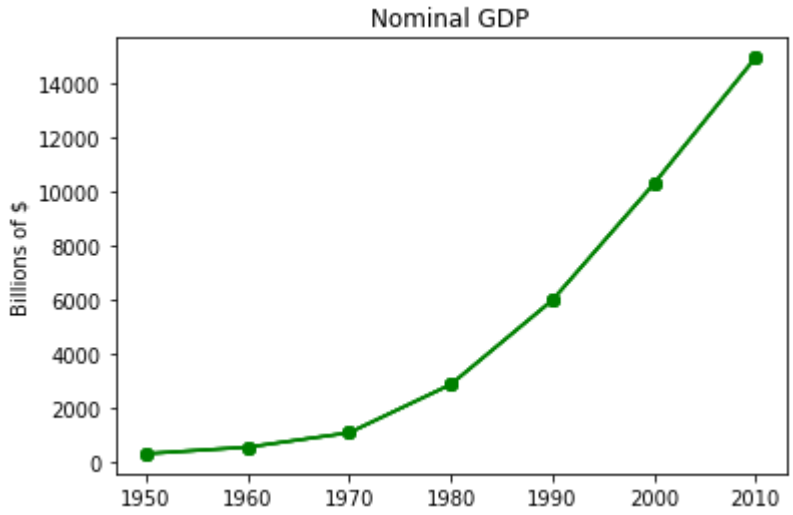
In [11]:
```
# create a line chart, years on x-axis, gdp on y-axis
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x2188e30a748>]
```

In [12]:
```
plt.title("Nominal GDP")
```

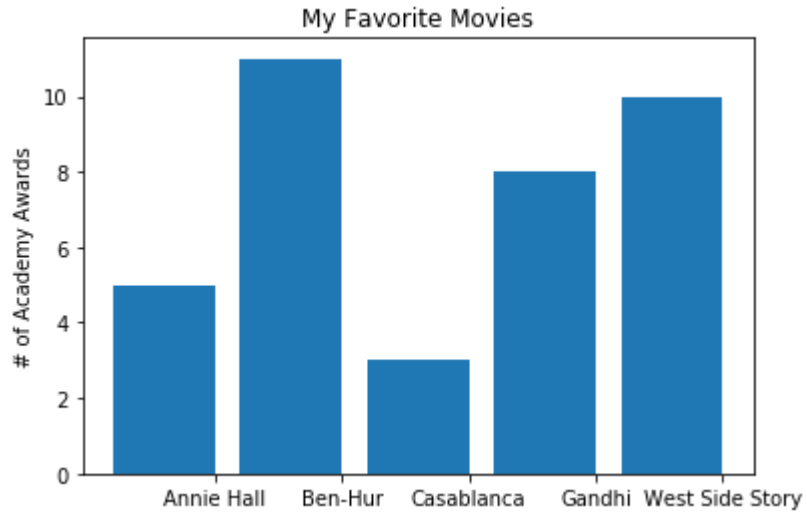
Out[12]:
```
Text(0.5,1,'Nominal GDP')
```

In [14]:
```
plt.ylabel("Billions of $")
plt.show()
```



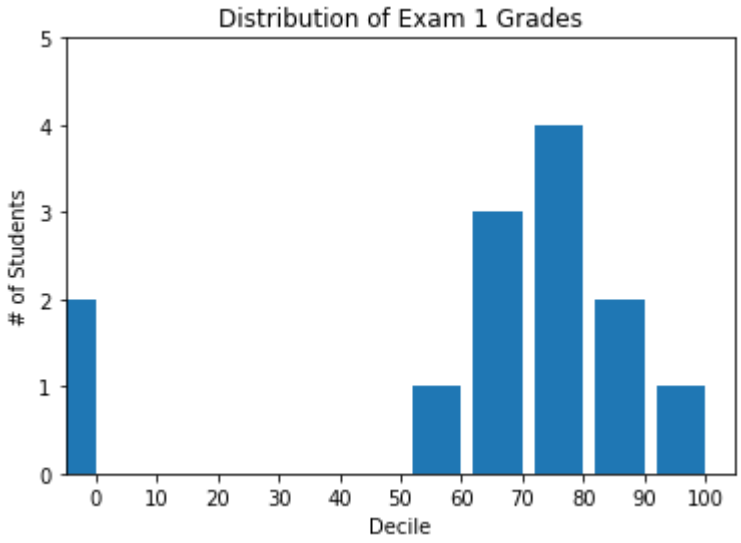
Create a simple bar chart.

In [17]:
```
movies = ["Annie Hall", "Ben-Hur", "Casablanca", "Gandhi", "West Side Story"
]
num_oscars = [5, 11, 3, 8, 10]

# bars are by default width 0.8, so we'll add 0.1 to the left coordinates
# so that each bar is centered
xs = [i + 0.1 for i, _ in enumerate(movies)]

# plot bars with left x-coordinates [xs], heights [num_oscars]
plt.bar(xs, num_oscars)
plt.ylabel("# of Academy Awards")
plt.title("My Favorite Movies")

# label x-axis with movie names at bar centers
plt.xticks([i + 0.5 for i, _ in enumerate(movies)], movies)

plt.show()
```



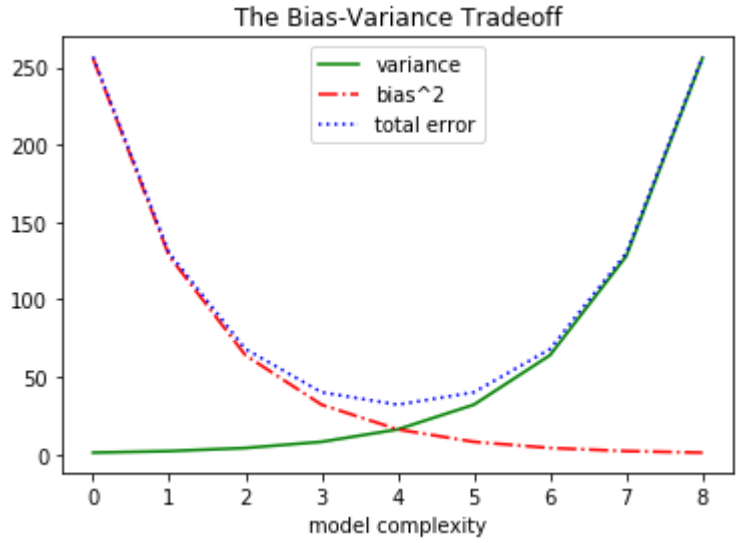
Create a simple histogram.

In [19]:
```
grades = [83,95,91,87,70,0,85,82,100,67,73,77,0]
decile = lambda grade: grade // 10 * 10
histogram = Counter(decile(grade) for grade in grades)

plt.bar([x - 4 for x in histogram.keys()], # shift each bar to the left by 4
            histogram.values(),                # give each bar its correct hei
ght
            8)                                     # give each bar a width of 8
plt.axis([-5, 105, 0, 5])                  # x-axis from -5 to 105,
                                               # y-axis from 0 to 5
plt.xticks([10 * i for i in range(11)])    # x-axis labels at 0, 10, ..., 100
plt.xlabel("Decile")
plt.ylabel("# of Students")
plt.title("Distribution of Exam 1 Grades")
plt.show()
```



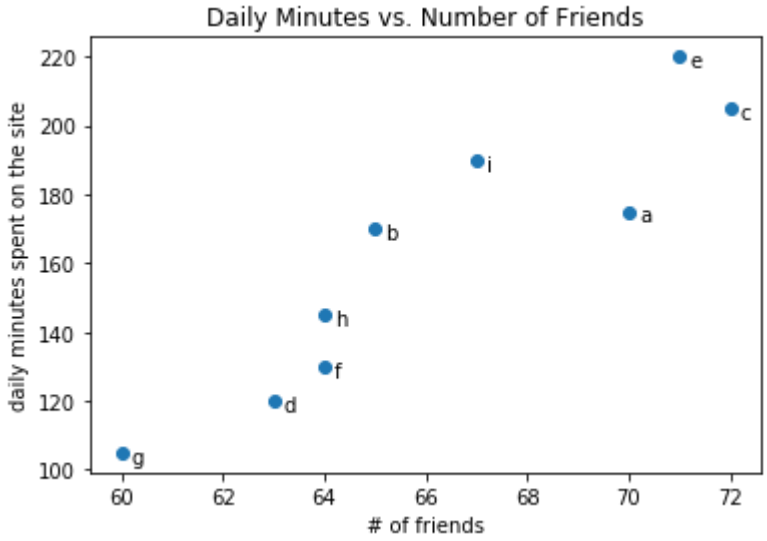
Create a line chart.

In [20]:
```
variance     = [1,2,4,8,16,32,64,128,256]
bias_squared = [256,128,64,32,16,8,4,2,1]
total_error  = [x + y for x, y in zip(variance, bias_squared)]

xs = range(len(variance))

# we can make multiple calls to plt.plot
# to show multiple series on the same chart
plt.plot(xs, variance,     'g-',  label='variance')    # green solid line
plt.plot(xs, bias_squared, 'r-.', label='bias^2')      # red dot-dashed line
plt.plot(xs, total_error,  'b:',  label='total error') # blue dotted line

# because we've assigned labels to each series
# we can get a legend for free
# loc=9 means "top center"
plt.legend(loc=9)
plt.xlabel("model complexity")
plt.title("The Bias-Variance Tradeoff")
plt.show()
```



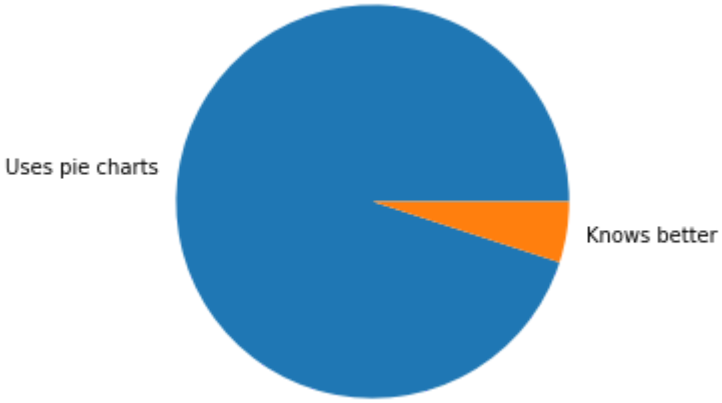
Create a scatterplot.

In [21]:
```
friends = [ 70, 65, 72, 63, 71, 64, 60, 64, 67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

plt.scatter(friends, minutes)

    # label each point
for label, friend_count, minute_count in zip(labels, friends, minutes):
    plt.annotate(label,
                 xy=(friend_count, minute_count), # put the label with its poin
t
                 xytext=(5, -5), # but slightly offset
                 textcoords='offset points')

plt.title("Daily Minutes vs. Number of Friends")
plt.xlabel("# of friends")
plt.ylabel("daily minutes spent on the site")
plt.show()
```



Create a pie chart.

In [22]:
```
plt.pie([0.95, 0.05], labels=["Uses pie charts", "Knows better"])

# make sure pie is a circle and not an oval
plt.axis("equal")
plt.show()
```



Save a chart in a function and recall the function.

In [23]:
```
def make_chart_pie_chart(plt):

    plt.pie([0.95, 0.05], labels=["Uses pie charts", "Knows better"])

    # make sure pie is a circle and not an oval
    plt.axis("equal")
    plt.show()
```

In [24]:
```
make_chart_pie_chart(plt)
```