Chapter 2: Lists

The Basics

In *Mathematica*, you can make a list of values:

```
list1 = {a, b, c, d, e}
{a, b, c, d, e}
```

Each element in your list is tagged by its order. The first element in list1 is *a* and the second element is *b*. To extract any element in your list, simply type the list name and the order in double square brackets:

```
list1[[1]]
list1[[2]]
a
b
```

If you have more than one list, you can add, subtract, multiply or divide one list by another. The use of the semicolon after the first input suppresses that output, which is useful if you are defining something and do not need *Mathematica* to repeat it, as seen above with the definition of *list1*.

```
list2 = {1, 2, 3, 4, 5};
addlist = list1 + list2

{1 + a, 2 + b, 3 + c, 4 + d, 5 + e}
```

The sum of the lists was named *addlist*. Since *addlist* is also a list, you can also call up individual elements:

```
addlist[[1]]
1 + a
```

You can also perform parallel mathematical operations using a list:

```
lpluslist1 = 1 + list1
{1 + a, 1 + b, 1 + c, 1 + d, 1 + e}
inverselist1 = 1 / list1
```

$$\left\{\frac{1}{a}, \frac{1}{b}, \frac{1}{c}, \frac{1}{d}, \frac{1}{e}\right\}$$

Being able to extract individual elements out of a list allows you to use the elements in mathematically operations without having to work with the entire list:

```
value1 =
 list1[[1]] + list2[[2]] * list1[[3]] / addlist[[4]] * addlist[[2]]
```

$$a + \frac{2\ (2 + b)\ c}{4 + d}$$

Expand and Simplify

Naming each equation becomes useful when the answer is not in a form that you want. There are times in which you want the answer to be written as a sum. Use *Expand* to multiply everything out into a sum of values.

```
value2 = Expand[value1]
```

$$a + \frac{4\ c}{4 + d} + \frac{2\ b\ c}{4 + d}$$

To expand exponential equations, use *PowerExpand* rather than *Expand*:

```
PowerExpand[(x y)^n]
```

$$x^n\ y^n$$

To simplify any output, use the command *Simplify*:

```
value3 = Simplify[value2]
```

$$\frac{2\ (2 + b)\ c + a\ (4 + d)}{4 + d}$$

Labeling Lists

To display your output with a text label, use the *Print* command:

```
Print["List 1   ", list1]
Print["List 2   ", list2]
Print["Sum of List 1 and List 2   ", addlist]
List 1   {a, b, c, d, e}
List 2   {1, 2, 3, 4, 5}
Sum of List 1 and List 2   {1 + a, 2 + b, 3 + c, 4 + d, 5 + e}
```

The *Print* command will only print text that is enclosed with quotation marks. *Print* can also print blank spaces, as shown above, which are useful to separate text and values.

You can also use *Print* to print text before and after your values:

```
Print["Sum of List 1 and List 2 = ", addlist[[1]],
 " (1st Element of List)"]
Sum of List 1 and List 2 = 1 + a (1st Element of List)
```

ListPlot

There are some real good uses for lists, especially when plots are needed. Define a list of x-values and a list of y-values:

```
xvalues = {1, 2, 3, 4, 5};
yvalues = {2, 4, 6, 8, 10};
```

Before you can plot, you need to have a list of {x, y} values for each point. Define your points as a new list:

```
totalList = {xvalues, yvalues}
{{1, 2, 3, 4, 5}, {2, 4, 6, 8, 10}}
```

The two lists are now defined as row vectors. To view the matrix, use the *TableForm* command:

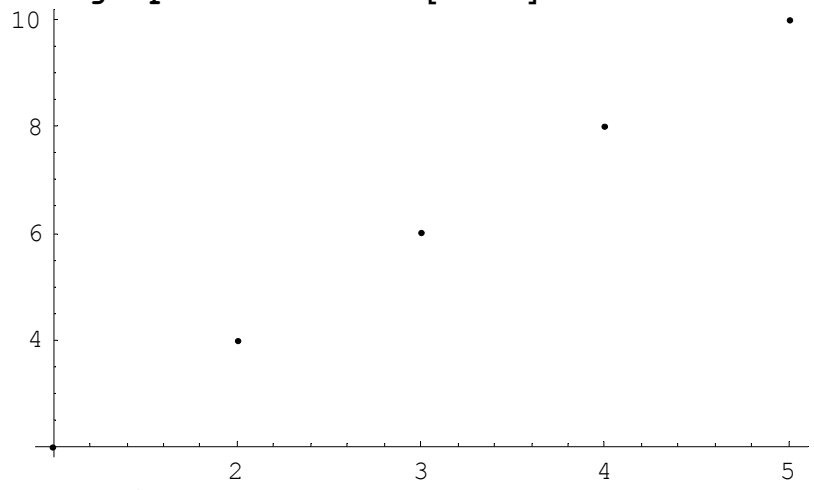
```
TableForm[totalList]
1       2       3       4       5
2       4       6       8       10
```

Remember that each set of values in the above list is a row (left to right) vector. We want the values called the column (up and down) vectors. The *Transpose* command will switch the row vectors into column vectors and vice versa:

```
data = Transpose[totalList]
{{1, 2}, {2, 4}, {3, 6}, {4, 8}, {5, 10}}
```

Our new list, called data, contains all our points for the graph. To plot these points, use the *ListPlot* command. *ListPlot* is used for points, while *Plot* is used for functions.

```
datagraph1 = ListPlot[data]
```

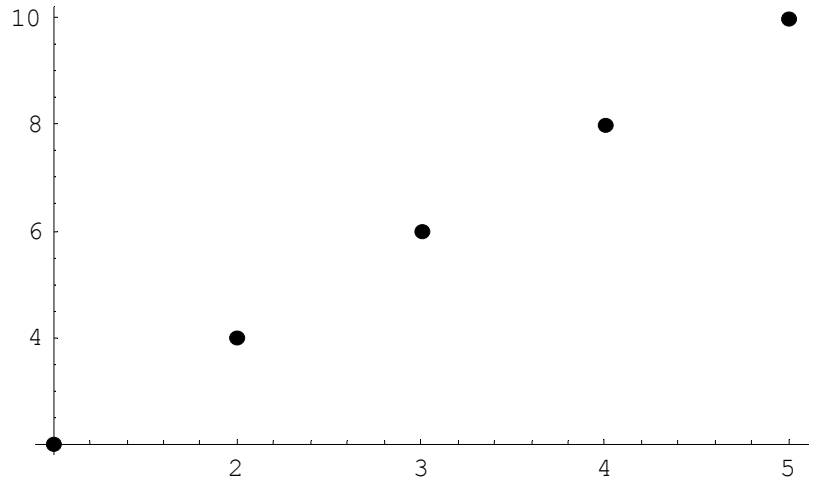



```
- Graphics -
```

PlotStyle

The pointsize is really small and hard to see. Also, the output "*Graphics*" is unnecessary. To increase the pointsize, use *PlotStyle→PointSize[0.02]*. Adjust the numerical value of the pointsize until it's satisfactory. To get rid of "*Graphics*," put a semicolon after the input. The semicolon will not suppress the graph.

```
datagraph2 = ListPlot[data, PlotStyle → PointSize[0.02]];
```



PlotLabel

Most instructors require that graphs be labeled. To give the graph a title, use:

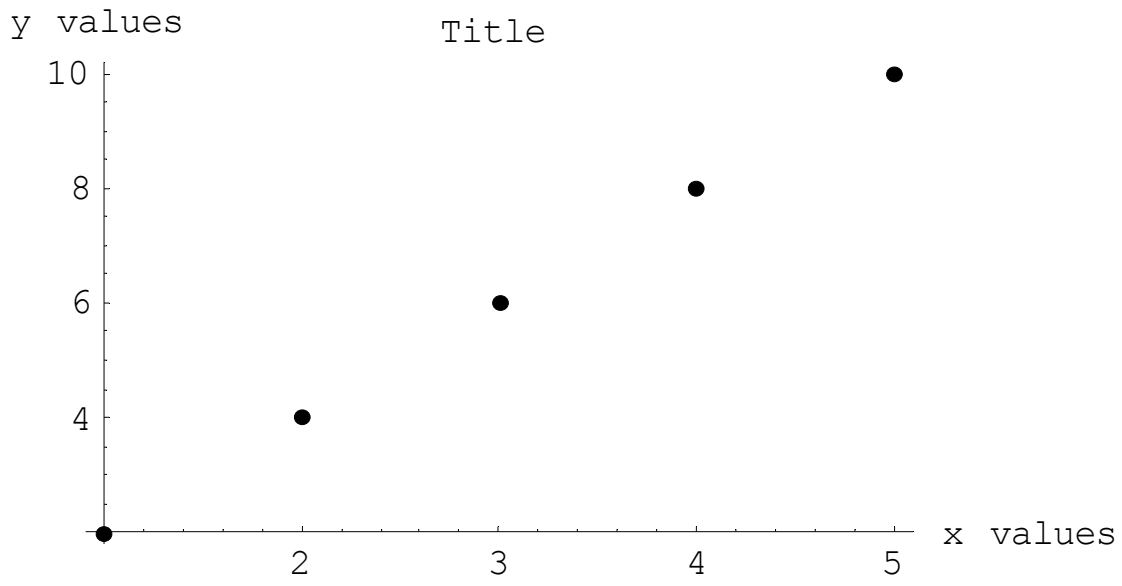
PlotLabel→"Title"

To name the axes, use:

AxesLabel→{"x values","y values"}

Note that all text must be enclosed by quotation marks, just as with the *Print* command.

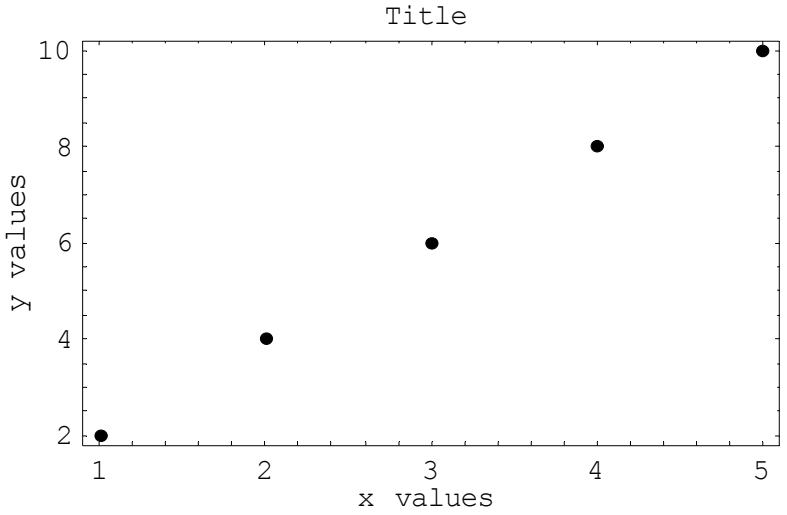
```
datagraph3 = ListPlot[data, PlotStyle → PointSize[0.02],
   PlotLabel → "Title",
   AxesLabel → {"x values", "y values"}];
```



Frames and Gridlines

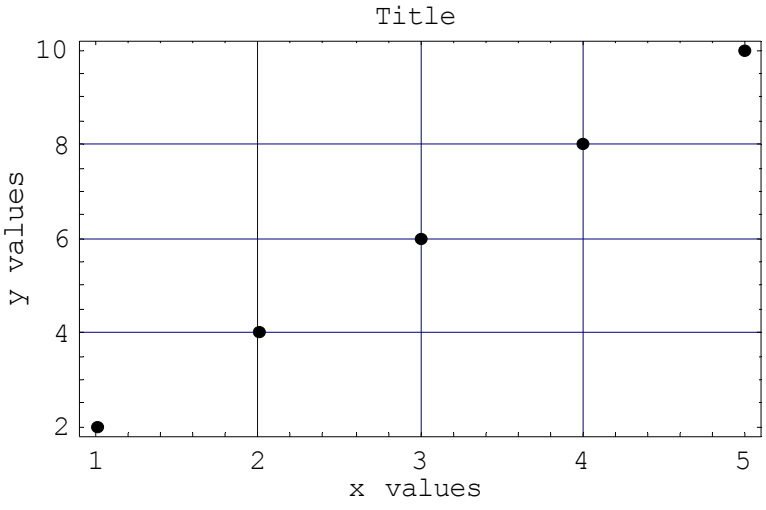
If you want to enclose your graph within a frame, use *Frame→True* and change the *AxesLabel* command to *FrameLabel* (since there are no axes within a frame):

```
datagraph4 = ListPlot[data, PlotStyle → PointSize[0.02],
    PlotLabel → "Title", Frame → True,
    FrameLabel → {"x values", "y values"}];
```



For gridlines, keep the *Frame* command and add *GridLines→Automatic*:

```
datagraph5 = ListPlot[data, PlotStyle → PointSize[0.02],
    PlotLabel → "Title", Frame → True,
    FrameLabel → {"x values", "y values"},
    GridLines → Automatic];
```



Fitting a Function

It is often necessary to fit a line or curve to a set of points. The *Fit* command can fit an equation to a set of points where the first argument is the data to fit, the second argument is a list of the functions to use in the fit and the last argument is the independent variable.

```
fiteqn1 = Fit[data, {1, x}, x]
3.66374 ×10^-15 + 2. x
```

If you feel that the fit might be of a higher order polynomial, add *x^2* (and above) to the list of functions used in the fit:

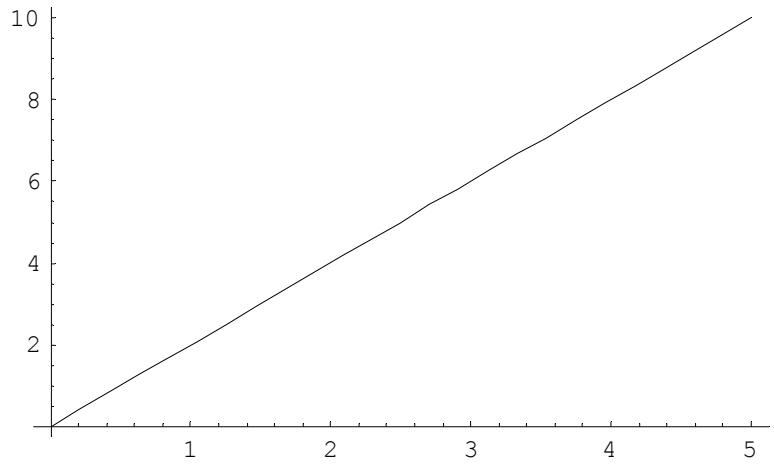
```
fiteqn2 = Fit[data, {1, x, x^2, x^3}, x]
-1.42109 ×10^-14 + 2. x - 1.79856 ×10^-14 x^2 + 2.19269 ×10^-15 x^3
```

Note: The *Fit* function is not suitable for the analysis of experimental data. A full statistical treatment is required for that purpose. You will learn about this in Integrated Lab.

Plotting a Function

The equation says that $y = 2x$ since all the other coefficients are close to zero and can be ignored. To show the plot of the fit line, use the *Plot* command. The first argument is the function to plot and the second is a list containing the independent variable, its min and max values:

```
fitplot = Plot[fiteqn1, {x, 0, 5}];
Print["The equation of the line is ", fiteqn1,
 ", which simplifies into y = 2x."]
```




```
The equation of the line is
 3.66374 ×10^-15 + 2. x, which simplifies into y = 2x.
```

From Physical Chemistry, 6th Edition by Peter Atkins:

Problem 11.2
The wavelength of the emission maximum from a small pinhole in an electrically heated container was determined at a series of temperatures, and the results are given below. Deduce a value of the Planck constant, h, in J s.

θ/°C	**1000**	**1500**	**2000**	**2500**	**3000**	**3500**
λ_{max}/nm	**2181**	**1600**	**1240**	**1035**	**878**	**763**

Equation 11.7 : T $\lambda_{max} = \frac{h\,c}{5\,k}$

k = Boltzman's constant (1.381 x 10^{-23} J / K)

c = speed of light (3 x 10^{8} m / s)

Equation 11.7 can be written as $\lambda_{max} = \frac{h\,c}{5\,k}\ \frac{1}{T}$ where λ_{max} is the dependent variable y, $\frac{1}{T}$ is the independent variable x and $\frac{h\,c}{5\,k}$ is the slope of the line.

The slope must have units of m K in order to compute h in J s. Convert λ_{max} from nm to m. Then convert °C to K and take the inverse of T.

Scatter plot a list of $\{\frac{1}{T}, \lambda_{max}\}$ values and fit the points with a line. From the slope, determine h and calculate the % error.

$$\% \text{ error} = \frac{\text{actual} - \text{calculated}}{\text{actual}} \text{ x } 100\%$$

The actual value of h = 6.626 x 10^{-34} J s

Answer: Problem 11.2

Make a list of λ_{max} values and a list of $\frac{1}{T}$ values:

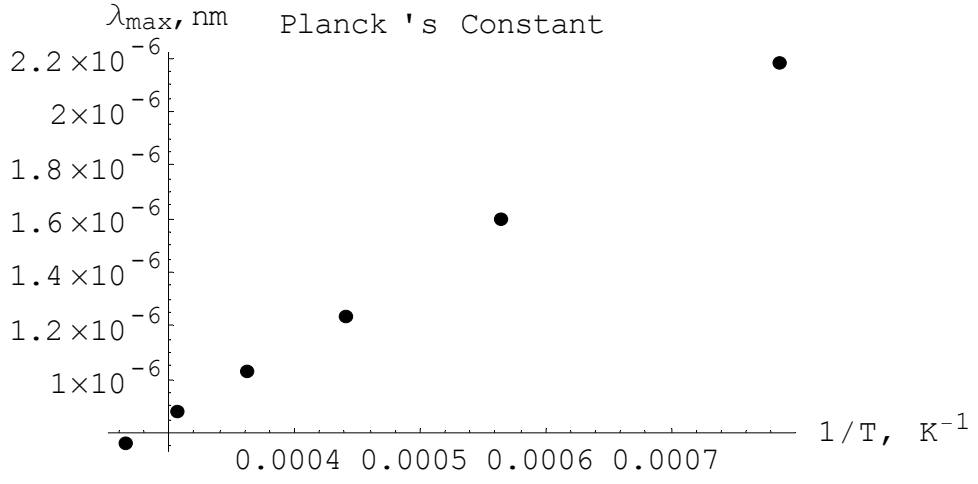
```
λmax = 10^-9 * {2181, 1600, 1240, 1035, 878, 763};
t = 273.15 + {1000, 1500, 2000, 2500, 3000, 3500};
inverseT = 1 / t;
data1 = {inverseT, λmax}
```

$$\left\{\{0.000785453,\ 0.000563968,\ 0.000439918,\ 0.000360601,\ 0.000305516,\ 0.000265031\},\ \left\{\frac{2181}{1000000000},\ \frac{1}{625000},\ \frac{31}{25000000},\ \frac{207}{200000000},\ \frac{439}{500000000},\ \frac{763}{1000000000}\right\}\right\}$$

```
data2 = Transpose[data1]
```

$$\left\{\left\{0.000785453,\ \frac{2181}{1000000000}\right\},\ \left\{0.000563968,\ \frac{1}{625000}\right\},\ \left\{0.000439918,\ \frac{31}{25000000}\right\},\ \left\{0.000360601,\ \frac{207}{200000000}\right\},\ \left\{0.000305516,\ \frac{439}{500000000}\right\},\ \left\{0.000265031,\ \frac{763}{1000000000}\right\}\right\}$$

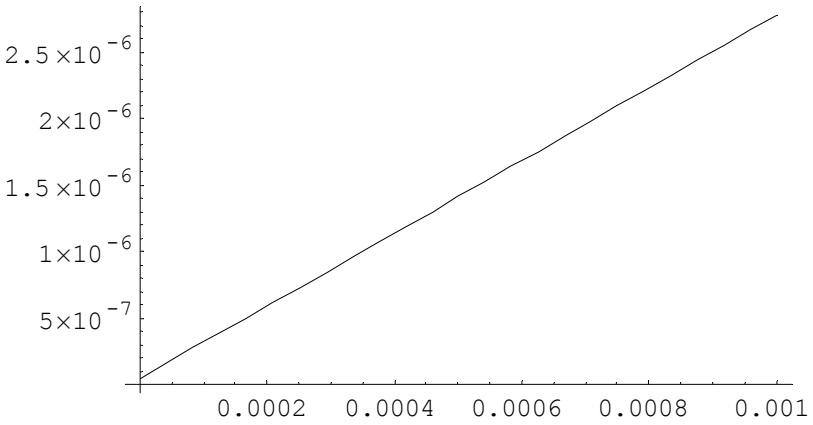
```
dataplot = ListPlot[data2, PlotStyle → PointSize[0.02],
    PlotLabel → "Planck's Constant",
    AxesLabel → {"1/T, K^-1", "λmax, nm"}];
```



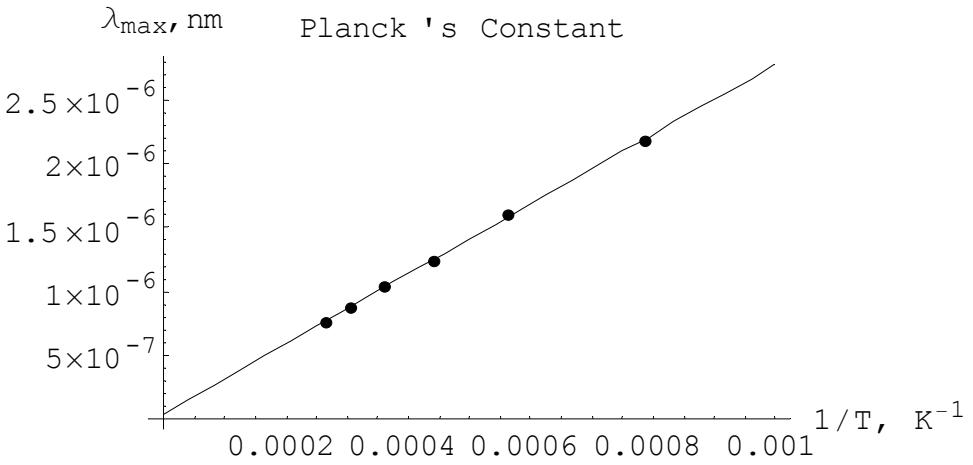

```
fiteqn = Fit[data2, {1, x}, x]
```

$$4.57072 \times 10^{-8} + 0.00272847\ x$$

```
fitplot = Plot[fiteqn, {x, 0, 0.001}];
```




```
planckPlot = Show[{dataplot, fitplot}];
```



The slope of the line is 0.00272847 m K, which is equal to $\frac{h\,c}{5\,k}$ so h = slope ($\frac{5\,k}{c}$):

```
k = 1.381 * 10^-23 J K^-1;
c = 3 * 10^8 m s^-1;
planck = 0.00272847 m K * (5 * k / c);
Print["Planck's constant = ", %]
Planck 's constant  = 6.28003 ×10^-34 J s

h = 6.626 * 10^-34 J s;
error = (h - planck) / h * 100;
Print["Error = ", error, "%"]
Error  = 5.22142 %
```