In[1]:=
```
(* - Lists*
 1. How to generate:  Table command; nested lists
 2. Length and Dimensions
 3. REARRANGING LISTS: Sort, Rotate, etc.
  4. Restructuring Lists: Flatten, Drop, etc.
  5. Combining Lists: Join, Union, Intersection
 6. Operating on lists: the Map command
 7. Lists as vectors and matrices

*)

(* 1. How to generate:

   Table[expr, {imax}] generates a list of imax copies of
    expr. Table[expr, {i, imax}] generates a list of the values of expr when
    i runs from 1 to imax. Table[expr, {i, imin, imax}] starts with i =
 imin. Table[expr, {i, imin, imax, di}] uses steps
  di. Table[expr, {i, imin, imax}, {j, jmin, jmax}, ... ] gives a nested list.
    *)
```

Table[RandomInteger[], {10}]

Out[1]= {0, 0, 1, 1, 1, 1, 0, 1, 1, 1}

In[2]:=
```
Table[i, {i, 10}]
```

Out[2]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In[3]:=
```
Table[i, {i, 10, 21, 3}]
```

Out[3]= {10, 13, 16, 19}

In[4]:=
```
Length[%]
```

Out[4]= 4

In[5]:=
```
ColumnForm[%%]
```

Out[5]=
10
13
16
19

In[6]:=
```
Table[Sin[n x], {n, 1, 7, 2}]
```
Out[6]=
$$\{\text{Sin}[x], \text{Sin}[3\,x], \text{Sin}[5\,x], \text{Sin}[7\,x]\}$$

In[7]:=

In[8]:=

In[9]:=

In[10]:=

In[11]:=

In[12]:=

In[13]:=

In[14]:=

In[15]:=

In[16]:=

In[17]:=
```
Table[x^i + y^j, {i, 3}, {j, 4}]
```
Out[17]=
$$\left\{\left\{x + y,\ x + y^2,\ x + y^3,\ x + y^4\right\},\ \left\{x^2 + y,\ x^2 + y^2,\ x^2 + y^3,\ x^2 + y^4\right\},\ \left\{x^3 + y,\ x^3 + y^2,\ x^3 + y^3,\ x^3 + y^4\right\}\right\}$$

In[18]:=
```
Dimensions[%]
```
Out[18]=
$$\{3, 4\}$$

In[19]:=
```
TableForm[%%](*for a human*)
```
Out[19]//TableForm=

$x + y$	$x + y^2$	$x + y^3$	$x + y^4$
$x^2 + y$	$x^2 + y^2$	$x^2 + y^3$	$x^2 + y^4$
$x^3 + y$	$x^3 + y^2$	$x^3 + y^3$	$x^3 + y^4$

In[20]:=
```
MatrixForm[%%%]
```
Out[20]//MatrixForm=
$$\begin{pmatrix} x + y & x + y^2 & x + y^3 & x + y^4 \\ x^2 + y & x^2 + y^2 & x^2 + y^3 & x^2 + y^4 \\ x^3 + y & x^3 + y^2 & x^3 + y^3 & x^3 + y^4 \end{pmatrix}$$

In[21]:=
```
mf := MatrixForm
```

In[22]:=
```
vv = {1, -1}; mm = {{a, b}, {c, d}}; mm // mf
```
Out[22]//MatrixForm=
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

In[23]:=
```
mm.vv
```
Out[23]=
$$\{a - b,\ c - d\}$$

In[24]:=
```
vv.mm
```
Out[24]=
```
{a - c, b - d}
```

In[25]:=
```
(*both above are vectors*)
```

In[26]:=
```
vv.mm.vv (*scalar*)
```
Out[26]=
```
a - b - c + d
```

In[27]:=
```
vv.vv (*scalar*)
```
Out[27]=
```
2
```

In[28]:=
```
Transpose[mm] // mf
```
Out[28]//MatrixForm=
$$\begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

In[29]:=
```
Inverse[mm] // mf
```
Out[29]//MatrixForm=
$$\begin{pmatrix} \frac{d}{-b\,c+a\,d} & -\frac{b}{-b\,c+a\,d} \\ -\frac{c}{-b\,c+a\,d} & \frac{a}{-b\,c+a\,d} \end{pmatrix}$$

In[30]:=
```
Det[mm]
```
Out[30]=
```
-b c + a d
```

In[31]:=
```
Clear[vv, mm];
```

In[32]:=
```
(* REARRANGING LISTS *)
Sort[{abd, abc, ab, a, d, 10, a, 10, 5.5,
      1/4, 23/2, ada}]
```
Out[32]=
$$\left\{\frac{1}{4}, 5.5, 10, 10, \frac{23}{2}, a, a, ab, abc, abd, ada, d\right\}$$

In[33]:=
```
mylist = Table[i, {i, 10}]
```
Out[33]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In[34]:=

In[35]:=

In[36]:=
```
Permutations[{a, b, c}]
```
Out[36]=
```
{{a, b, c}, {a, c, b}, {b, a, c}, {b, c, a}, {c, a, b}, {c, b, a}}
```

In[37]:=
```
Length[%] == 3! // Simplify
```
Out[37]=
```
True
```

In[38]:=

```
(* Restructuring Lists *)
```

```
Clear[mylist, c]
```

In[39]:=

In[40]:=

In[41]:=

In[42]:=

In[43]:=

```
(*Drop[list, n] gives list with its first n elements dropped
 Drop[list, {n}] gives list with its n"th" element dropped.
  Take[list, n] gives the first n elements of list
 Take[list, {m, n}] gives elements m through n of list
 HW: create a simple list; try all this*)
```

```
(*Element extraction*)
```

```
list = {a, b, c, d};
list[[2]]
```

Out[45]= b

```
Clear[list]; list = {{a, b}, {c, d}}
```

Out[46]= {{a, b}, {c, d}}

In[47]:=

```
list[[2]]
```

Out[47]= {c, d}

In[48]:=
```
list[[1, 2]]
```
Out[48]= b

In[49]:=
```
MatrixForm[list]
```
Out[49]//MatrixForm=

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

```
? Select
```

Select[*list*, *crit*] picks out all elements e_i of *list* for which *crit*[e_i] is True.

Select[*list*, *crit*, *n*] picks out the first *n* elements for which *crit*[e_i] is True.

Select[*crit*] represents an operator form of Select that can be applied to an expression. ≫

In[51]:=
```
list2 = Table[i, {i, 7}]
```
Out[51]= {1, 2, 3, 4, 5, 6, 7}

In[52]:=
```
Select[list2, EvenQ]
```
Out[52]= {2, 4, 6}

In[53]:=

(* Combining Lists *)

```
list1 = Table[i, {i, 3}]
```
Out[53]= {1, 2, 3}

In[54]:=
```
list2 = Table[i, {i, 7}]
```
Out[54]= {1, 2, 3, 4, 5, 6, 7}

In[55]:=
```
Join[list1, list2]
```
Out[55]= {1, 2, 3, 1, 2, 3, 4, 5, 6, 7}

In[56]:=
```
Union[list1, list2]
```
Out[56]= {1, 2, 3, 4, 5, 6, 7}

In[57]:=
```
Intersection[list1, list2]
```
Out[57]= {1, 2, 3}

In[58]:=

(*Operating on Lists*)

In[59]:=
```
Clear[list]
```

In[60]:=
```
list = Table[x^n, {n, 4}]
```

Out[60]= $\{x, x^2, x^3, x^4\}$

In[61]:=
```
D[list, x]
```

Out[61]= $\{1, 2\,x, 3\,x^2, 4\,x^3\}$

In[62]:=
```
Sin[list]
```

Out[62]= $\{\text{Sin}[x], \text{Sin}[x^2], \text{Sin}[x^3], \text{Sin}[x^4]\}$

In[63]:=
```
Sqrt[list]
```

Out[63]= $\{\sqrt{x}, \sqrt{x^2}, \sqrt{x^3}, \sqrt{x^4}\}$

In[64]:=
```
% // PowerExpand (*x>0*)
```

Out[64]= $\{\sqrt{x}, x, x^{3/2}, x^2\}$

In[65]:=

In[66]:=
```
(* Not all fuctions are listable! - Check *)
```

In[67]:=

In[68]:=

In[69]:=
```
Clear[f];
```

In[70]:=
```
Map[f, list]
```

Out[70]= $\{f[x], f[x^2], f[x^3], f[x^4]\}$

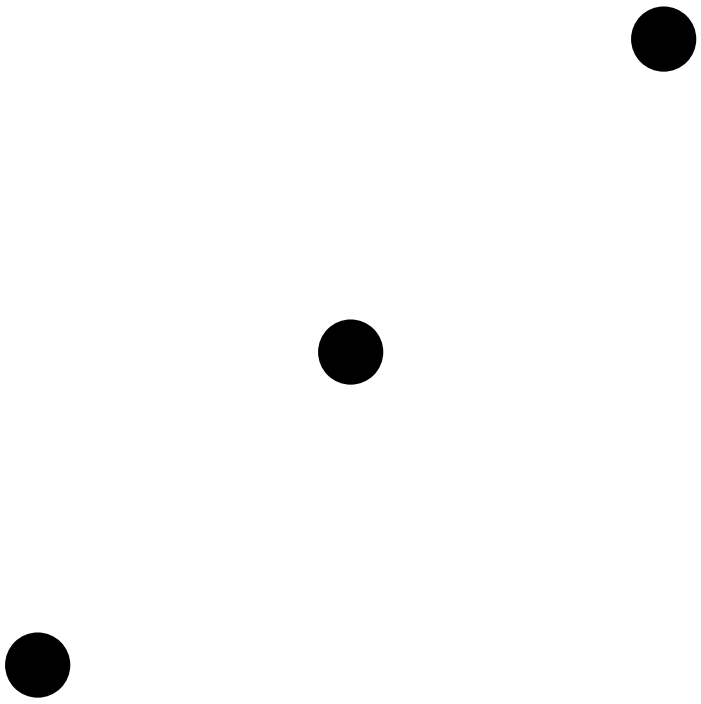
In[71]:=
```
f /@ list (*same thing*)
```

Out[71]= $\{f[x], f[x^2], f[x^3], f[x^4]\}$

In[72]:=
```
Clear[list]; list = {{1, 1}, {2, 2}, {3, 3}};
```

In[73]:=
```
Graphics[{PointSize → .1,
  Point /@ list}]
```

Out[73]=



In[74]:=

In[75]:=

In[76]:=

In[77]:=

In[78]:=
```
(*Working
 with lists*)
```

In[79]:=
```
list1
```

Out[79]= {1, 2, 3}

In[80]:=
```
Plus @@ list1
```

Out[80]= 6

In[81]:=
```
(* HW: write a function av[y_] which
  would find an average of a list y of arbitrary length*)
```

In[82]:=

In[83]:=
```
Mean[list1]
```
Out[83]= 2

In[84]:=
```
% * Length[list1]
```
Out[84]= 6

In[85]:=
```
(*Pure function -reminder*)

(*Note: in f[x_]:=
 Sin[x] the variable x does not matter and can be replaced by anything. In fact,
x_ indicates a slot in which the variable will be place when
  evaluating Sin. The idea is then to skip x alltogether -
 this leads to a "Pure function", analog of an operator*)
```

In[86]:=

In[87]:=

In[88]:=

In[89]:=

In[90]:=
```
(* more useful for differential operators*)
```

In[91]:=
```
der2 := D[#, x, x] &
```

In[92]:=
```
der2[x^2]
```
Out[92]= 2

In[93]:=
```
Clear[a]; der2[E^(a x)]
```
Out[93]= $a^2 e^{a x}$

In[94]:=
```
(*other examples: anything can go into the slot,
not only numbers. Below are Lists and Graphics*)
```

In[95]:=
```
myav := (Plus @@ #) / Length[#] & (*average of a list*)
```

In[96]:=
```
list = {1, 2, 3, 4};
```

In[97]:=
```
myav[list]
```
Out[97]= $\frac{5}{2}$

In[98]:=
```
Mean[list]
```
Out[98]= $\frac{5}{2}$

In[99]:=
```
disp :=
  Export["t.ps", #, "EPS"] &
```

In[100]:=
```
(*this creates a nice postscript outside Mathematica. E.g.,
if you like graphics in fig,
you write disp[fig] and will get a postscript file t.ps with the picture*)
```

In[101]:=

In[102]:=

In[103]:=

In[104]:=

In[105]:=

In[106]:=

In[107]:=

In[108]:=

In[109]:=

In[110]:=

In[111]:=
```
step := # + RandomInteger[] &
```

In[112]:=
```
Nest[step, 0, 10]
```
Out[112]= 5

In[113]:=
```
NestList[step, 0, 10]
```
Out[113]= {0, 0, 0, 1, 1, 2, 2, 2, 2, 3, 3}

In[114]:=
```
(*can use in the coin experiment*)
```

In[115]:=

In[116]:=

In[117]:=

In[118]:=

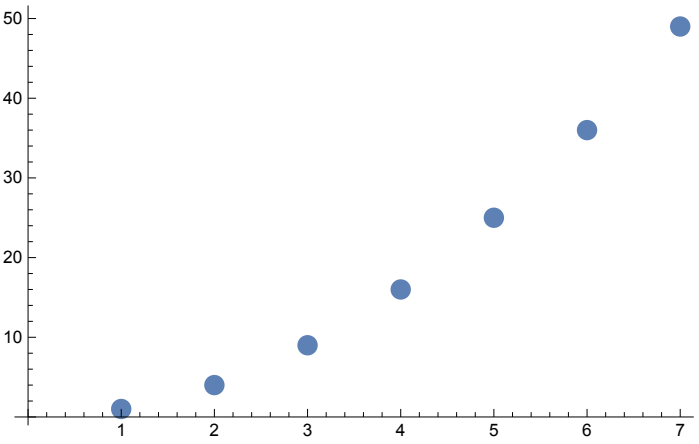
In[119]:=

In[120]:=

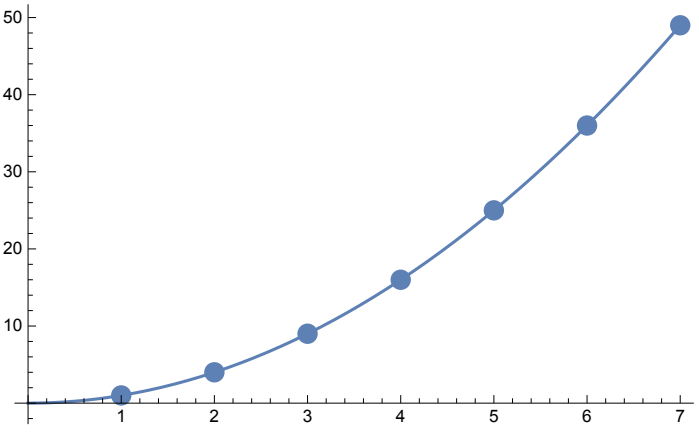
In[121]:=

In[122]:=
```
list = Table[i^2, {i, 7}]
```
Out[122]=
```
{1, 4, 9, 16, 25, 36, 49}
```

In[123]:=
```
Clear[lp]; lp = ListPlot[list, PlotStyle → PointSize[.03]]
```
Out[123]=

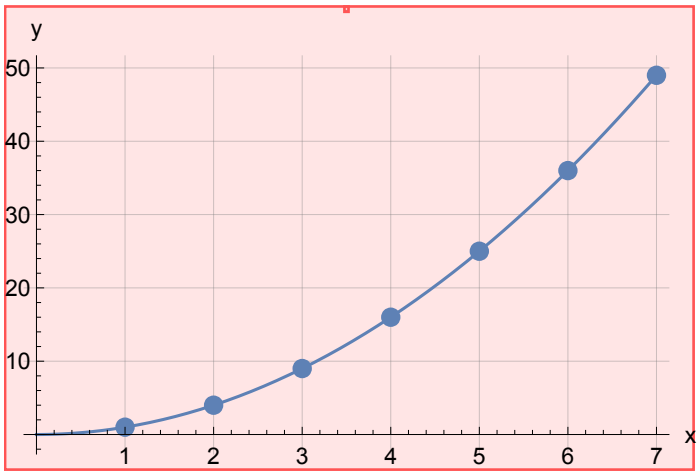


In[124]:=
```
Show[Plot[x^2, {x, 0, 7}], lp]
```
Out[124]=



In[125]:=
```
Show[Plot[x^2, {x, 0, 7}], lp, PlotLabel → "x^2",
 AxesLabel → {"x", "y"}, LabelStyle → Medium, GridLines → Automatic]
```

Out[125]=



In[126]:=

In[127]:=

In[128]:=

In[129]:=

In[130]:=

In[131]:=

In[132]:=

In[133]:=

In[134]:=

In[135]:=

In[136]:=

In[137]:=

In[138]:=

In[139]:=

In[140]:=

In[141]:=

In[142]:=

In[143]:=

In[144]:=

In[145]:=

In[146]:=

In[147]:=

In[148]:=

In[149]:=

In[150]:=

In[151]:=

In[152]:=

In[153]:=

In[154]:=

In[155]:=