Learning Wolfram Language

Exercises for Section 1 | Starting Out: Elementary Arithmetic

1.1 Compute1+2+3.

```
In[ ]:= 1 + 2 + 3
Out[ ]= 6
```

1.2 Add the numbers 1,2,3,4,5.

```
In[ ]:= 1 + 2 + 3 + 4 + 5
Out[ ]= 15
```

1.3 Multiply the numbers 1, 2, 3, 4, 5.

```
In[ ]:= 1 * 2 * 3 * 4 * 5
Out[ ]= 120
```

1.4 Compute 5 squared (i . e . 5*5 or 5 raised to the power 2)

```
In[ ]:= 5 ^ 2
Out[ ]= 25
```

1.5 Compute 3 raised to the fourth power .

```
In[ ]:= 3 ^ 4
Out[ ]= 81
```

1.6 Compute 10 raised to the power 12 (a trillion) .

```
In[ ]:= 10 ^ 12
Out[ ]= 1 000 000 000 000
```

1.7 Compute 3 raised to the power 7 x8 .

```
In[ ]:= 3 ^ (7 * 8)
Out[ ]= 523 347 633 027 360 537 213 511 521
```

1.8 Add parentheses to 4 - 2*3 + 4 to make 14.

```
In[ ]:= (4 - 2) * (3 + 4)
Out[ ]= 14
```

1.9 Compute twenty - nine thousand mutiplied by seventy - three .

```
In[ ]:= 29 000 * 73
Out[ ]= 2 117 000
```

+1.1 Add all integers from - 3 to + 3.

```
In[ ]:= -3 + -2 + -1 + 1 + 2 + 3
Out[ ]= 0
```

+1.2 Compute 24 divided by 3.

```
In[ ]:= 24 / 3
Out[ ]= 8
```

+1.3 Compute 5 raised to the power 100.

```
In[ ]:= 5 ^ 100
Out[ ]= 7 888 609 052 210 118 054 117 285 652 827 862 296 732 064 351 090 230 047 702 789 306 640 625
```

+1.4 Subtract 5 squared from 100

```
In[ ]:= 100 - (5 ^ 2)
Out[ ]= 75
```

+1.5 Multiply 6 by 5 squared, and add 7

```
In[ ]:= (6 * (5 ^ 2)) + 7
Out[ ]= 157
```

+1.6 Compute 3 squared minus 2 cubed .

```
In[ ]:= (3 ^ 2) - (2 ^ 3)
Out[ ]= 1
```

+1.7 Compute 2 cubed times 3 squared

```
In[ ]:= (2 ^ 3) * (3 ^ 2)
Out[ ]= 72
```

+1.8 Compute "double the sum of eight and negative eleven"

```
In[ ]:= (8 - 11) * 2
Out[ ]= -6
```

Exercises for Section 2 | Introducing Functions

2.1 Compute 7 + 6 + 5 using the function Plus

```
In[ ]:= Plus[7, 6, 5]
Out[ ]= 18
```

2.2 Compute 2 x (3 + 4) using Times and Plus

```
In[ ]:= Times[2, Plus[3, 4]]
Out[ ]= 14
```

2.3 Use Max to find the larger of 6x8 and 5x9

```
In[ ]:= Max[Times[6, 8], Times[5, 9]]
Out[ ]= 48
```

2.4 Use RandomInteger to generate a random number between 0 and 1000.

```
In[ ]:= RandomInteger[1000]
Out[ ]= 443
```

2.5 Use Plus and RandomInteger to generate a number between 10 and 20.

```
In[ ]:= Plus[10, RandomInteger[10]]
Out[ ]= 12
```

+2.1 Compute 5 x4x3x2 using Times .

```
In[ ]:= Times[5, 4, 3, 2]
Out[ ]= 120
```

+2.2 Compute 2 - 3 using Subtract

```
In[ ]:= Subtract[2, 3]
Out[ ]= -1
```

+2.3 Compute (8 + 7) + (9 + 2) using Times and Plus

```
In[ ]:= Times[Plus[8, 7], Plus[9, 2]]
Out[ ]= 165
```

+2.4 Compute (26 - 89)/9 using Subtract and Divide

```
In[ ]:= Divide[Subtract[26, 89], 9]
Out[ ]= -7
```

+2.5 Compute 100 - 5^2 using Subtract and Power

```
In[ ]:= Subtract[100, Power[5, 2]]
Out[ ]= 75
```

+2.6 Find the larger of 3^5 and 5^3

```
In[ ]:= Max[Power[3, 5], Power[5, 3]]
Out[ ]= 243
```

+2.7 Multiply 3 and the larger of 4^3 and 3^4

```
In[ ]:= Times[3, Max[Power[4, 3], Power[3, 4]]]
Out[ ]= 243
```

+2.8 Add two random numbers each between 0 and 1000.

```
In[ ]:= Plus[RandomInteger[1000], RandomInteger[1000]]
Out[ ]= 698
```

Exercises for Section 3 | First Look at Lists

3.1 Use Range to create the list {1, 2, 3, 4}

```
In[ ]:= Range[4]
Out[ ]= {1, 2, 3, 4}
```

3.2 Make a list of numbers up to 100

```
In[ ]:= Range[100]
Out[ ]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
  24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
  44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
  63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
  82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100}
```

3.3 Use Range and Reverse to create {4, 3, 2, 1}

```
In[ ]:= Reverse[Range[4]]
Out[ ]= {4, 3, 2, 1}
```

3.4 Make a list of numbers from 1 to 50 in reverse order .

```
In[ ]:= Reverse[Range[50]]
Out[ ]= {50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37,
  36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20,
  19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
```

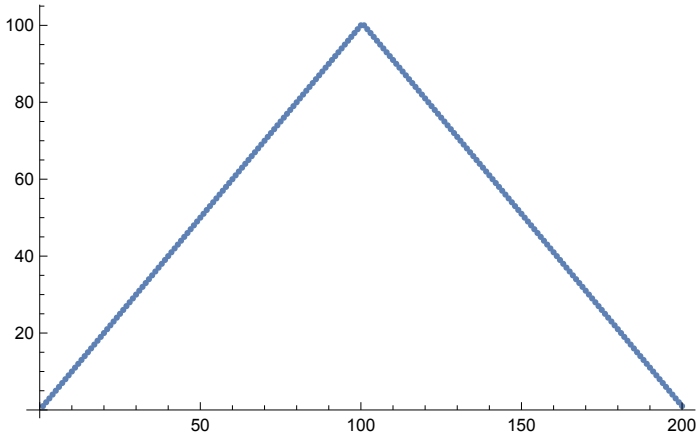
3.5 Use Range, Reverse and Join to create {1, 2, 3, 4, 4, 3, 2, 1}

```
In[ ]:= Join[Range[4], Reverse[Range[4]]]
Out[ ]= {1, 2, 3, 4, 4, 3, 2, 1}
```

3.6 Plot a list that counts up from 1 to 100, then down to 1.

```
In[ ]:= ListPlot[Join[Range[100], Reverse[Range[100]]]]
```

Out[ ]=



3.7 Use Range and RandomInteger to make a list with a random lenght up to 10.

```
In[ ]:= Range[RandomInteger[10]]
Out[ ]= {1, 2, 3}
```

3.8 Find a simpler form for Reverse[Reverse[Range[10]]]

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

3.9 Find a simpler form to Join[{1, 2}, Join[{3, 4}, {5}]]

```
In[ ]:= Range[5]
Out[ ]= {1, 2, 3, 4, 5}
```

3.10 Find a simpler form for Join[Range[10], Join[Range[10], Range[5]]]

```
In[ ]:= Join[Range[10], Range[10], Range[5]]
Out[ ]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5}
```

3.11 Find a simpler form for Reverse[Join[Range[20], Reverse[Range[20]]]]

```
In[ ]:= Join[Range[20], Reverse[Range[20]]]
Out[ ]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
  20, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
```

+3.1 Compute the reverse of the reverse of {1, 2, 3, 4}

```
In[ ]:= Reverse[Reverse[Range[4]]]
Out[ ]= {1, 2, 3, 4}
```

+3.2 Use Range, Reverse and Join to create the list {1, 2, 3, 4, 5, 4, 3, 2, 1} .

```
In[ ]:= Join[Range[4], Reverse[Range[4]]]
Out[ ]= {1, 2, 3, 4, 4, 3, 2, 1}
```

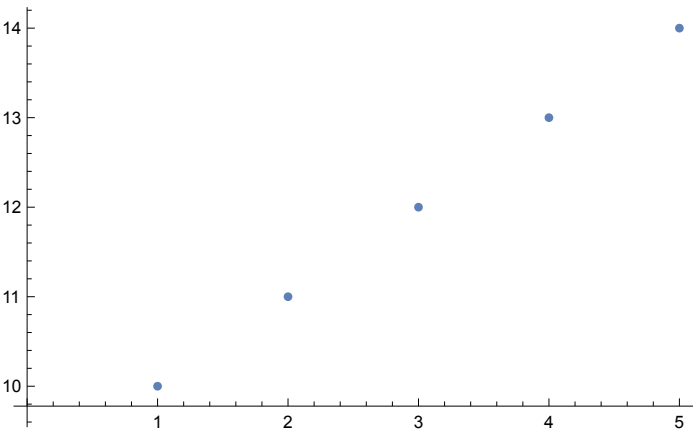
+3.3 Use Range, Reverse and Join to create {3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1}

```
In[ ]:= Join[Reverse[Range[3]], Reverse[Range[4]], Reverse[Range[5]]]
Out[ ]= {3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1}
```

+3.4 Plot the list numbers {10, 11, 12, 13, 14}

```
In[ ]:= ListPlot[Range[10, 14]]
```

Out[ ]=

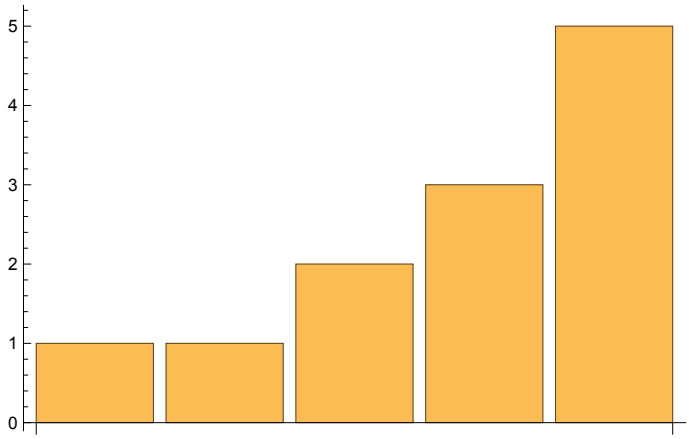


Exercises for Section 4 | Displaying Lists

4.1 Make a bar chart of {1, 1, 2, 3, 5}

```
In[ ]:= BarChart[{1, 1, 2, 3, 5}]
```

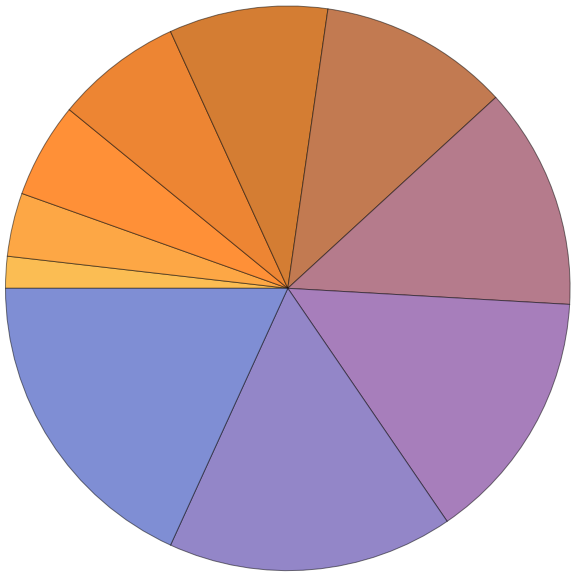
Out[ ]=



4.2 Make a pie chart of numbers from 1 to 10.

```
In[ ]:= PieChart[Range[10]]
```

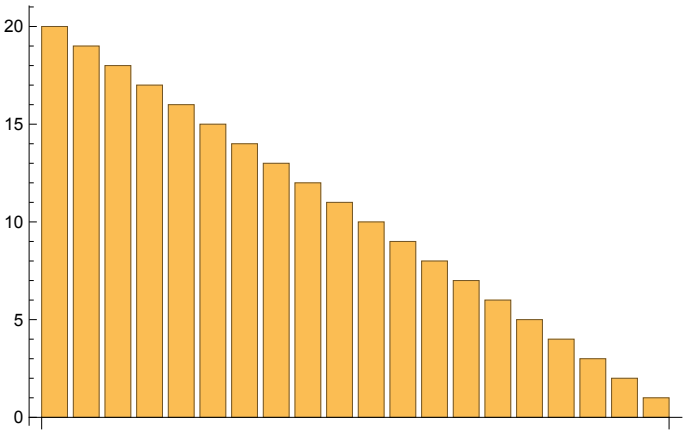
Out[ ]=



4.3 Make a bar chart of numbers counting down from 20 to 1.

In[ ]:=
```
BarChart[Reverse[Range[20]]]
```

Out[ ]=



4.4 Display numbers from 1 to 5 in a column .

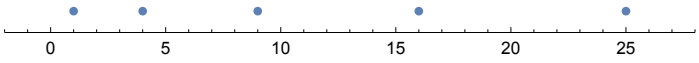
In[ ]:=
```
Column[Range[5]]
```

Out[ ]=
```
1
2
3
4
5
```

4.5 Make a number line plot of the squares {1, 4, 9, 16, 25} .

In[ ]:=
```
NumberLinePlot[Range[5]^2]
```

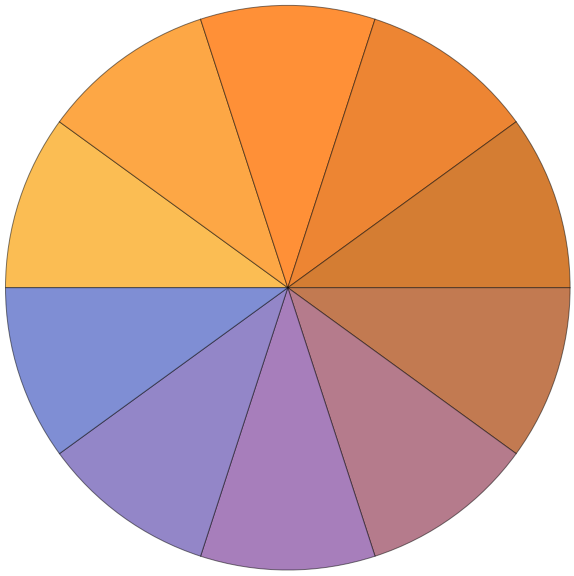
Out[ ]=



4.6 Make a pie chart with 10 identical segments, each of size 1.

In[ ]:=
```
PieChart[ConstantArray[1, 10]]
```

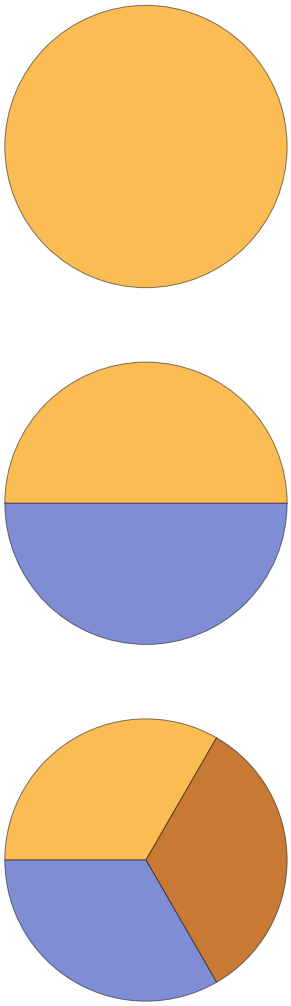
Out[ ]=



4.7 Make a column of pie charts with 1, 2 and 3 identical segments.

In[ ]:=
```
Column[{PieChart[{1}], PieChart[{1, 1}], PieChart[{1, 1, 1}]}, 3]
```

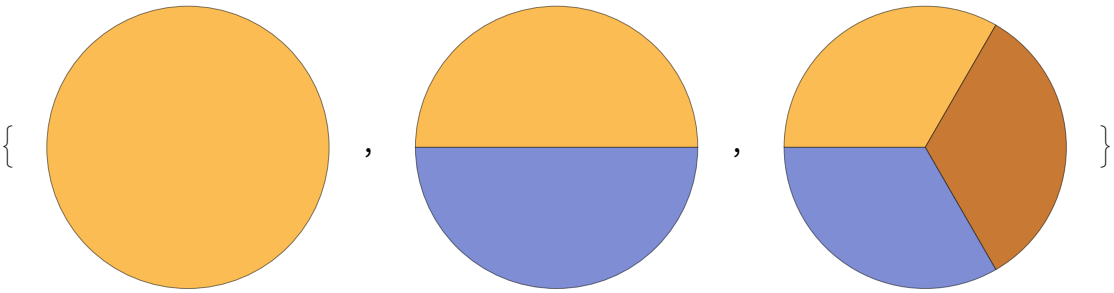
Out[ ]=



+4.1 Make a list of pie charts with 1, 2 and 3 identical segments .

In[ ]:=
```
{PieChart[{1}], PieChart[{1, 1}], PieChart[{1, 1, 1}]}
```

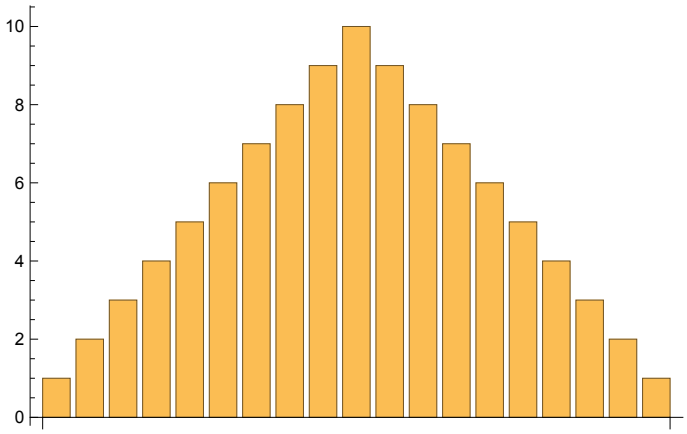
Out[ ]=



+4.2 Make a bar chart of the sequence 1, 2, 3, ..., 9, 10, 9, 8, 7, ..., 1.

```
In[ ]:= BarChart[Join[Range[10], Reverse[Range[9]]]]
```

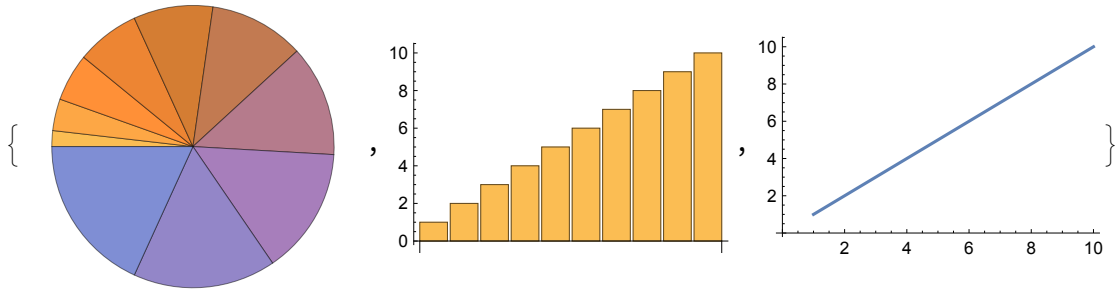
Out[ ]=



+4.3 Make a list of pie chart, bar chart and line plot of the numbers from 1 to 10.

```
In[ ]:= {PieChart[Range[10]], BarChart[Range[10]], ListLinePlot[Range[10]]}
```

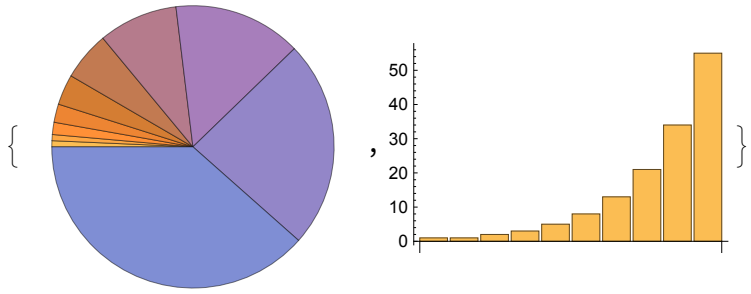
Out[ ]=



+4.4 Make a list of a pie chart and a bar chart of {1, 1, 2, 3, 5, 8, 13, 21, 34, 55}

```
In[ ]:= {PieChart[Fibonacci[Range[10]]], BarChart[Fibonacci[Range[10]]]}
```

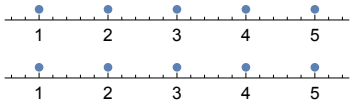
Out[ ]=



+4.5 Make a column of two number line plot of {1, 2, 3, 4, 5}

In[ ]:=
```
Column[{NumberLinePlot[Range[5]], NumberLinePlot[Range[5]]}]
```

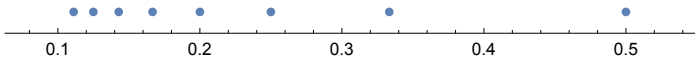
Out[ ]=



+4.6 Mae a number line of fractions 1/2, 1/3, ... through 1/9.

In[ ]:=
```
NumberLinePlot[{ConstantArray[1, 8] / Range[2, 9]}]
```

Out[ ]=



Exercises for Section 5 | Operations on Lists

5.1 Make a list of the first 10 squares, in reverse order .

In[ ]:=
```
Reverse[Range[10]^2]
```

Out[ ]=
```
{100, 81, 64, 49, 36, 25, 16, 9, 4, 1}
```

5.2 Find the total of the first 10 squares .

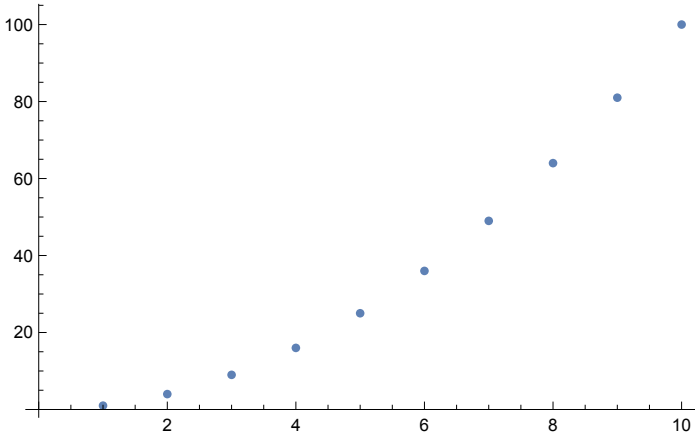
In[ ]:=
```
Total[Reverse[Range[10]^2]]
```

Out[ ]=
```
385
```

5.3 Make a plot of the first 10 squares, starting at 1.

In[ ]:=
```
ListPlot[Range[10]^2]
```

Out[ ]=



5.4 Use Sort, Join and Range to create {1, 1, 2, 2, 3, 3, 4, 4}.

In[ ]:=
```
Sort[Join[Range[4], Range[4]]]
```

Out[ ]=
```
{1, 1, 2, 2, 3, 3, 4, 4}
```

5.5 Use Range and + to make a list of numbers from 10 to 20, inclusive .

```
In[ ]:= Range[0, 10] + 10
Out[ ]= {10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}
```

5.6 Make a combined list of the first 5 squares and cubes (numbers raised to the power 3), sorted into order .

```
In[ ]:= Sort[Join[Range[5]^2, Range[5]^3]]
Out[ ]= {1, 1, 4, 8, 9, 16, 25, 27, 64, 125}
```

5.7 Find the number of digits in 2^128.

```
In[ ]:= Length[IntegerDigits[2^128]]
Out[ ]= 39
```

5.8 Find the first digit of 3^32

```
In[ ]:= First[IntegerDigits[2^32]]
Out[ ]= 4
```

5.9 Find the first 10 digits in 2^100.

```
In[ ]:= Take[IntegerDigits[2^100], 10]
Out[ ]= {1, 2, 6, 7, 6, 5, 0, 6, 0, 0}
```

5.10 Find the largest digit that appears in 2^20.

```
In[ ]:= Max[IntegerDigits[2^20]]
Out[ ]= 8
```

5.11 Find how many zeros appear in the digits of 2^1000.

```
In[ ]:= Count[IntegerDigits[2^1000], 0]
Out[ ]= 28
```

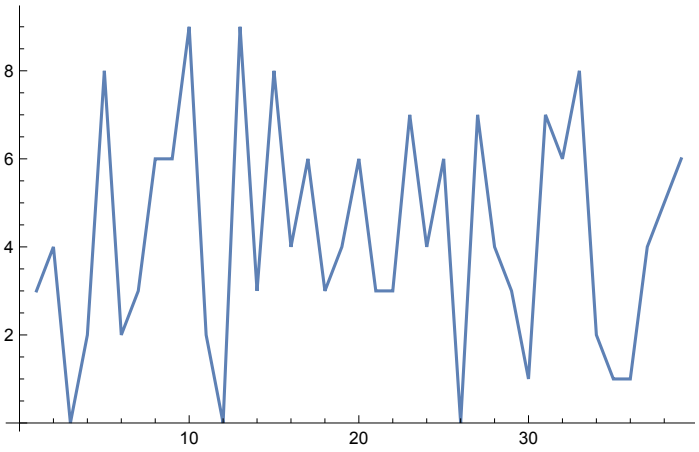
5.12 Use Part, Sort and IntegerDigit to find the second - smallest digit in 2^20.

```
In[ ]:= Part[Sort[IntegerDigits[2^20]], 2]
Out[ ]= 1
```

5.13 Make a line plot of the sequence of digits that appear in 2^128.

```
In[ ]:= ListLinePlot[IntegerDigits[2^128]]
```

Out[ ]=



5.14 Use Take and Drop to get the sequence 11 through 20 from Range[100].

```
In[ ]:= Take[Drop[Range[100], 10], 10]
Out[ ]= {11, 12, 13, 14, 15, 16, 17, 18, 19, 20}
```

+5.1 Make a list of the first 10 multiples of 3.

```
In[ ]:= Range[10] * 3
Out[ ]= {3, 6, 9, 12, 15, 18, 21, 24, 27, 30}
```

+5.2 Make a list of the first 10 squares using only Range and Times .

```
In[ ]:= Times[Range[10], Range[10]]
Out[ ]= {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

+5.3 Find the last digit of 2^37.

```
In[ ]:= Last[IntegerDigits[2^37]]
Out[ ]= 2
```

+5.4 Find the second-to-last digit of 2^32.

```
In[ ]:= Last[Drop[IntegerDigits[2^32], -1]]
Out[ ]= 9
```

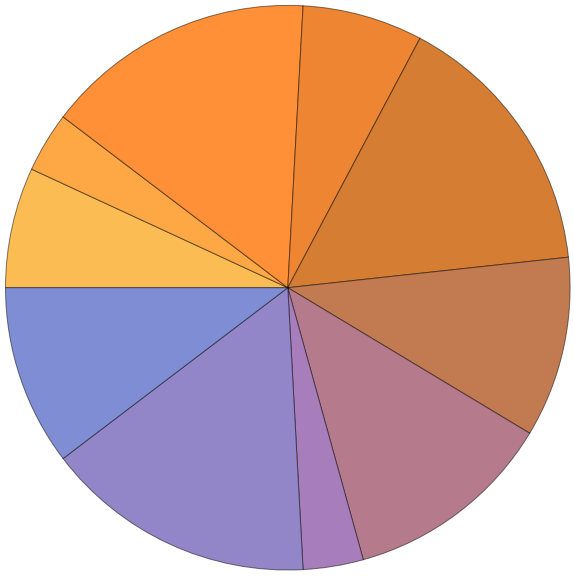
+5.5 Find the sum of all the digits of 3^126.

```
In[ ]:= Total[IntegerDigits[3^126]]
Out[ ]= 234
```

+5.6 Make a pie chart of the sequence of digits that appear in 2^32.

```
In[ ]:= PieChart[IntegerDigits[2^32]]
```

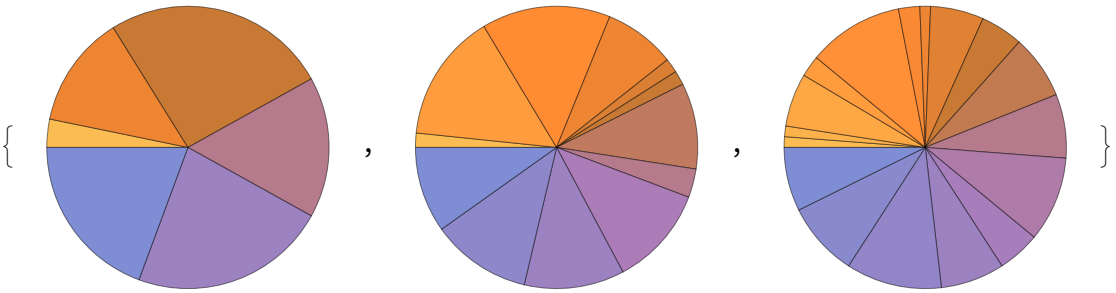
Out[ ]=



+5.7 Make a list of pie charts for the sequence of digits in 2^20, 2^40, 2^60.

```
In[ ]:= {PieChart[IntegerDigits[2^20]],
          PieChart[IntegerDigits[2^40]], PieChart[IntegerDigits[2^60]]}
```

Out[ ]=



Exercises for Section 6 | Making Tables

6.1 Make a list in which the number 1000 is repeated 5 times .

```
In[ ]:= Table[1000, {5}]
Out[ ]= {1000, 1000, 1000, 1000, 1000}
```

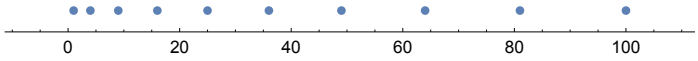
6.2 Make a table of the value of n^3 for n from 10 to 20.

```
In[ ]:= Table[n^3, {n, 10, 20}]
Out[ ]= {1000, 1331, 1728, 2197, 2744, 3375, 4096, 4913, 5832, 6859, 8000}
```

6.3 Make a number line plot of the first 20 squares .

```
In[ ]:= NumberLinePlot[Table[n^2, {n, 10}]]
```

Out[ ]=



6.4 Make a list of the even numbers (2, 4, 6, ...) up to 20.

```
In[ ]:= Table[n * 2, {n, 10}]
Out[ ]= {2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
```

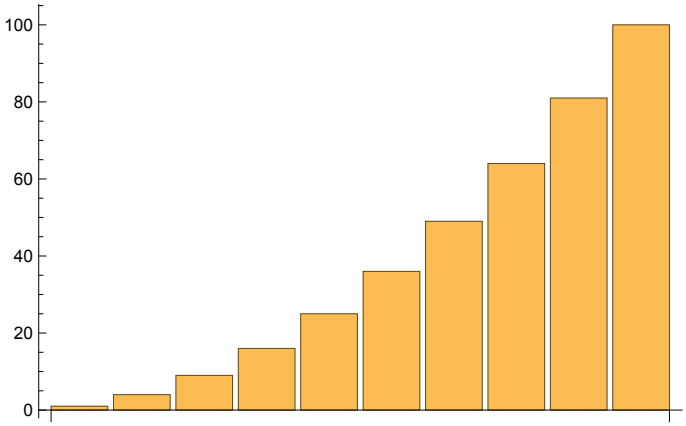
6.5 Use Table to get the same result as Range[10].

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

6.6 Make a bar chart of the first 10 squares .

```
In[ ]:= BarChart[Table[n^2, {n, 10}]]
```

Out[ ]=



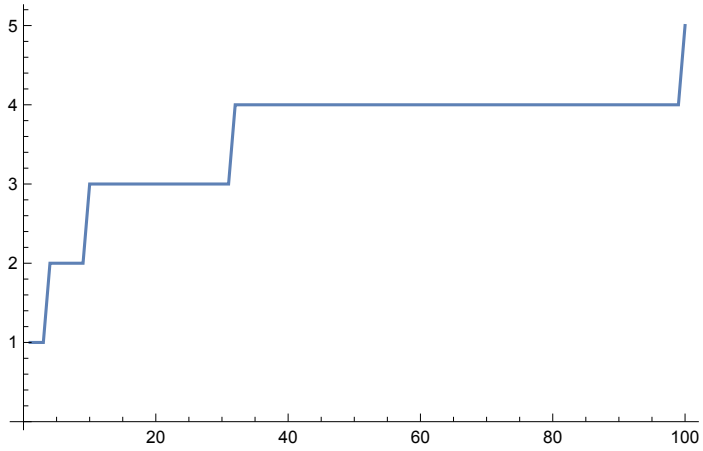
6.7 Make a table of lists of digits for the first 10 squares.

```
In[ ]:= Table[IntegerDigits[n^2], {n, 10}]
Out[ ]= {{1}, {4}, {9}, {1, 6}, {2, 5}, {3, 6}, {4, 9}, {6, 4}, {8, 1}, {1, 0, 0}}
```

6.8 Make a list line plot of the length of the sequence of digits for each of the first 100 squares.

```
In[ ]:= ListLinePlot[Table[Length[IntegerDigits[n^2]], {n, 100}]]
```

Out[ ]=



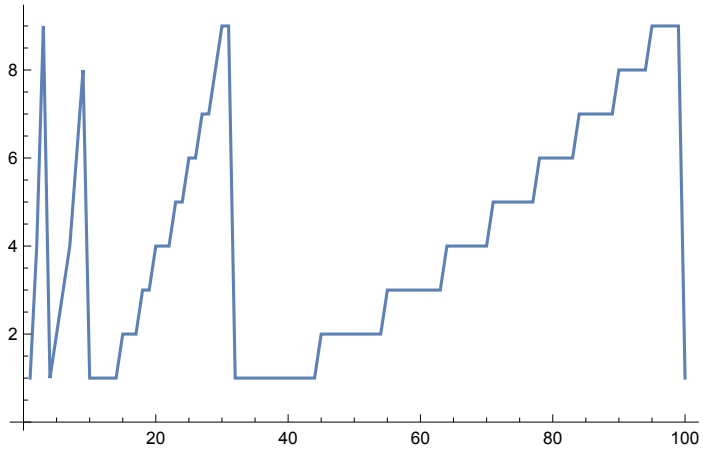
6.9 Make a table of the first digit of the first 20 squares.

```
In[ ]:= Table[First[IntegerDigits[n^2]], {n, 20}]
Out[ ]= {1, 4, 9, 1, 2, 3, 4, 6, 8, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 4}
```

6.10 Make a list line plot of the first digits of the first 100 squares.

```
In[ ]:= ListLinePlot[Table[First[IntegerDigits[n^2]], {n, 100}]]
```

Out[ ]=



+6.1 Make a list of the differences between n^3 and n^2 with n up to 10.

```
In[ ]:= Table[(n^3) - (n^2), {n, 10}]
Out[ ]= {0, 4, 18, 48, 100, 180, 294, 448, 648, 900}
```

+6.2 Make a list of the odd numbers (1, 3, 5, ...) up to 100.

```
In[ ]:= Table[(n * 2) - 1, {n, 50}]
```

```
Out[ ]= {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31,
 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65,
 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99}
```

+6.3 Make a list of the squares of even numbers up to 100.

```
In[ ]:= Table[(n * 2)^2, {n, 50}]
```

```
Out[ ]= {4, 16, 36, 64, 100, 144, 196, 256, 324, 400, 484, 576, 676, 784, 900,
 1024, 1156, 1296, 1444, 1600, 1764, 1936, 2116, 2304, 2500, 2704, 2916,
 3136, 3364, 3600, 3844, 4096, 4356, 4624, 4900, 5184, 5476, 5776, 6084,
 6400, 6724, 7056, 7396, 7744, 8100, 8464, 8836, 9216, 9604, 10 000}
```

+6.4 Create the list {-3, -2, -1, 0, 1, 2} using Range .

```
In[ ]:= Range[-3, 2]
```

```
Out[ ]= {-3, -2, -1, 0, 1, 2}
```

+6.5 Make a list for the numbers n up to 20 in which each element is a column of the value of n, n^2 and n^3.

```
In[ ]:= Table[Column[{n, n^2, n^3}], {n, 20}]
```

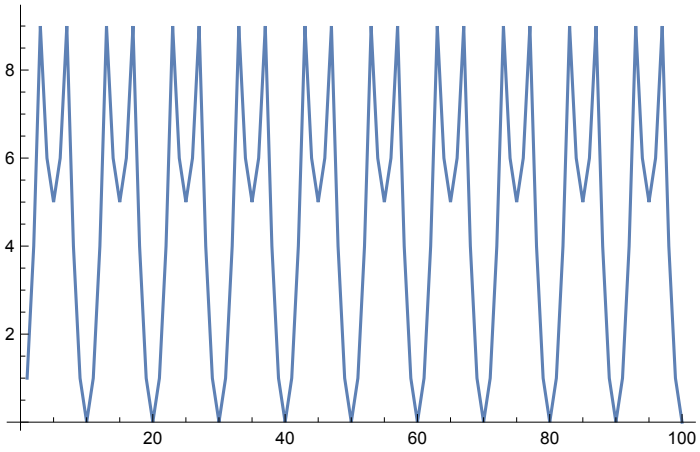
```
Out[ ]= { 1   2   3   4   5   6   7   8   9   10   11
          1 , 4 , 9 , 16 , 25 , 36 , 49 , 64 , 81 , 100 , 121 ,
          1   8   27  64  125 216 343 512 729 1000 1331

          12    13    14    15    16    17    18    19    20
          144 , 169 , 196 , 225 , 256 , 289 , 324 , 361 , 400 }
          1728  2197  2744  3375  4096  4913  5832  6859  8000
```

+6.6 Make a list line plot of the last digit of the first 100 squares.

```
In[ ]:= ListLinePlot[Table[Last[IntegerDigits[n^2]], {n, 100}]]
```

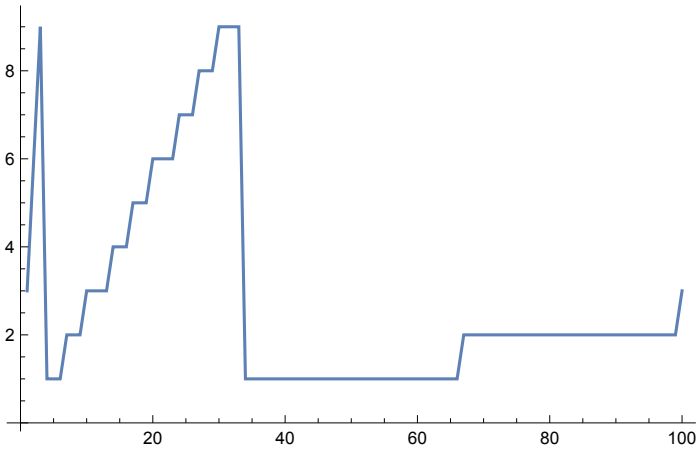
```
Out[ ]=
```



+6.7 Make a list line plot of the first digit of the first 100 multiplies of 3.

```
In[ ]:= ListLinePlot[Table[First[IntegerDigits[n * 3]], {n, 100}]]
```

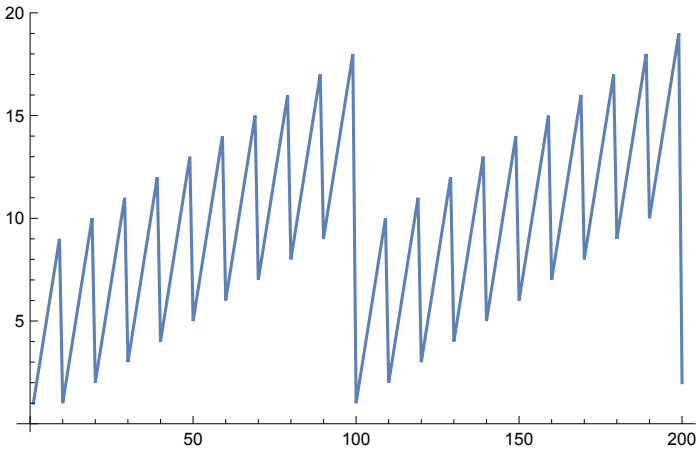
Out[ ]=



+6.8 Make a list line plot of the total of the digits for each number up to 200.

```
In[ ]:= ListLinePlot[Table[Total[IntegerDigits[n]], {n, 200}]]
```

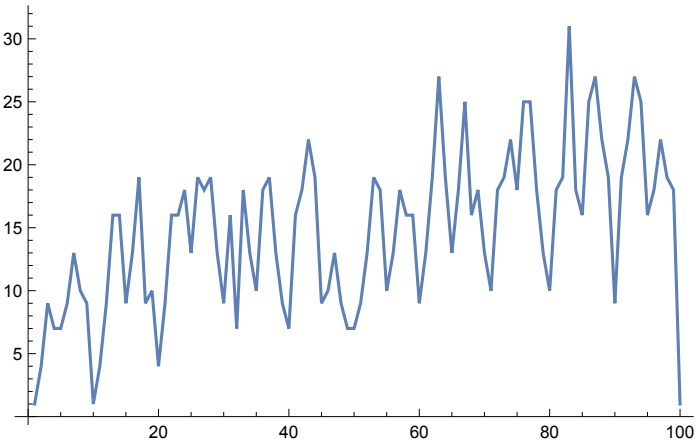
Out[ ]=



+6.9 Make a list line plot of the total of the digits for each of the first 100 squares.

```
In[•]:= ListLinePlot[Table[Total[IntegerDigits[n^2]], {n, 100}]]
```

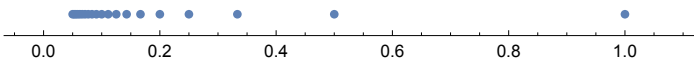
Out[•]=



+6.10 Make a number line plot of the numbers 1/n with n from 1 to 20.

```
In[•]:= NumberLinePlot[Table[1 / n, {n, 1, 20}]]
```

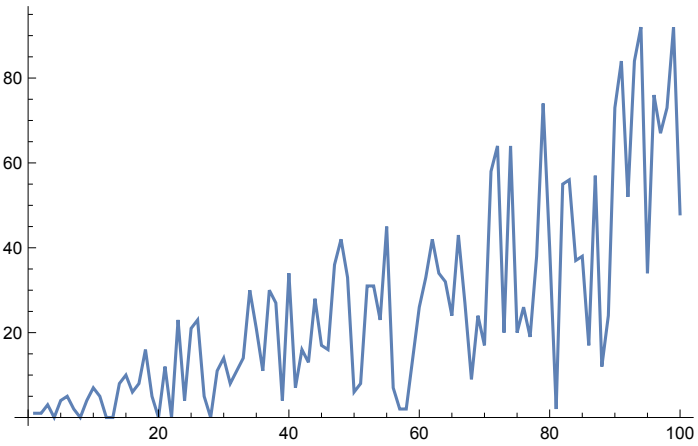
Out[•]=



+6.11 Make a line plot of a list of 100 random integers where the nth integer is between 0 and n.

```
In[•]:= ListLinePlot[Table[RandomInteger[n], {n, 100}]]
```

Out[•]=



Exercises for Section 7 | Colors and Styles

7.1 Make a list of red, yellow and green.

```
In[•]:= {Red, Yellow, Green}
```

Out[•]= {■, ■, ■}

7.2 Make a red, yellow, green column ("traffic light")

```
In[ ]:= Column[{Red, Yellow, Green}]
```

Out[ ]=

7.3 Compute the negation of the color Orange.

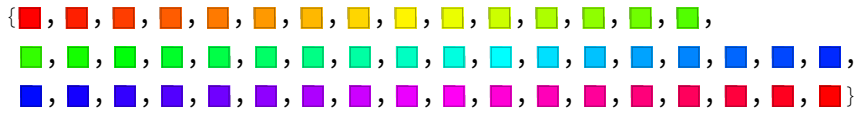
```
In[ ]:= ColorNegate[Orange]
```

Out[ ]=

7.4 Make a list of color with hues varying from 0 to 1 in steps of 0.02.

```
In[ ]:= Table[Hue[n], {n, 0, 1, 0.02}]
```

Out[ ]=



7.8 Make a list of numbers from 0 to 1 in steps of 0.1, each with a hue equal to its value.

```
In[ ]:= Table[{Hue[n], Style[n, Hue[n]]}, {n, 0, 1, 0.1}]
```

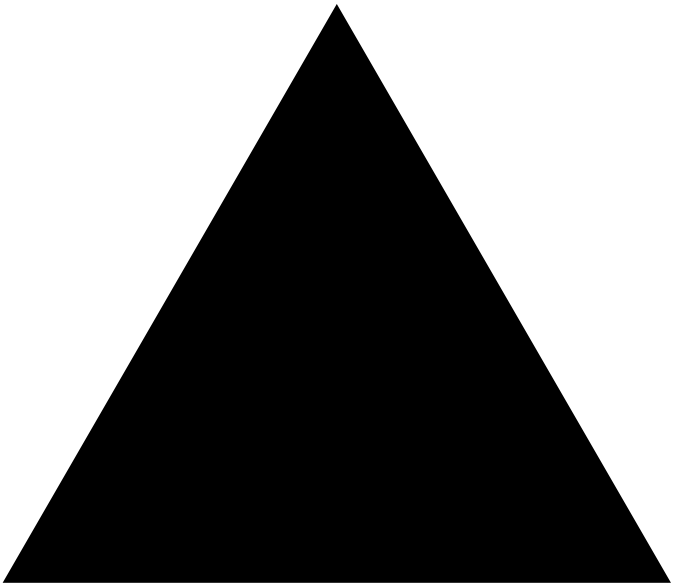
Out[ ]=

Exercises for Section 8 | Basic Graphics Objects

8.1 Use RegularPolygon to draw a triangle.

In[ ]:=
```
Graphics[RegularPolygon[3]]
```

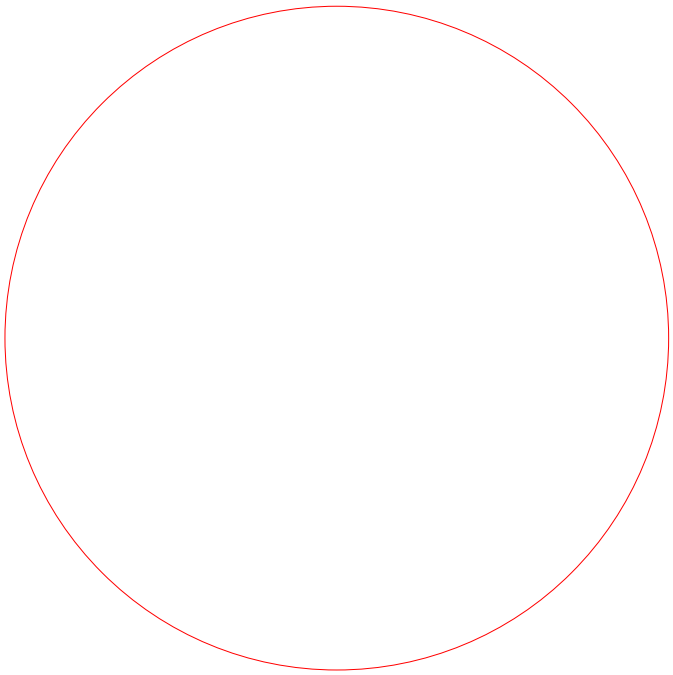
Out[ ]=



8.2 Make graphics of a red circle.

In[ ]:=
```
Graphics[Style[Circle[], Red]]
```

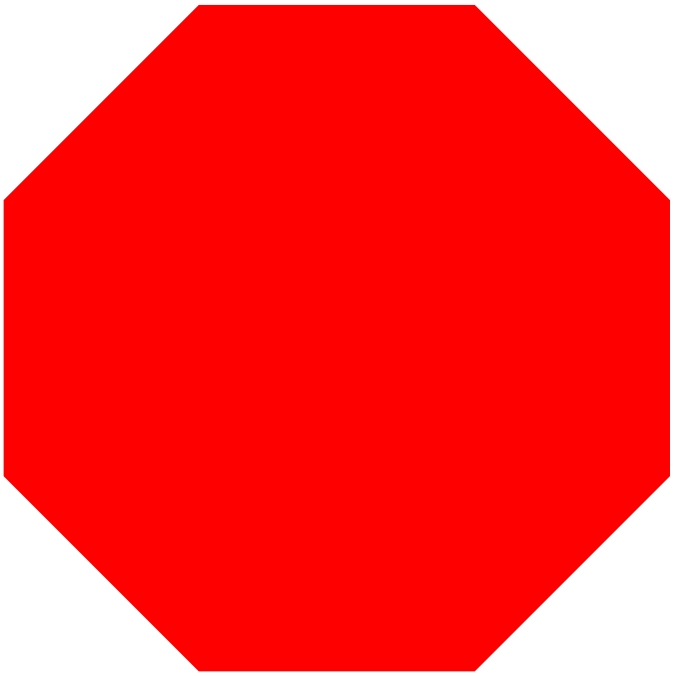
Out[ ]=



8.3 Make a red octagon.

In[ ]:=
```
Graphics[Style[RegularPolygon[8], Red]]
```

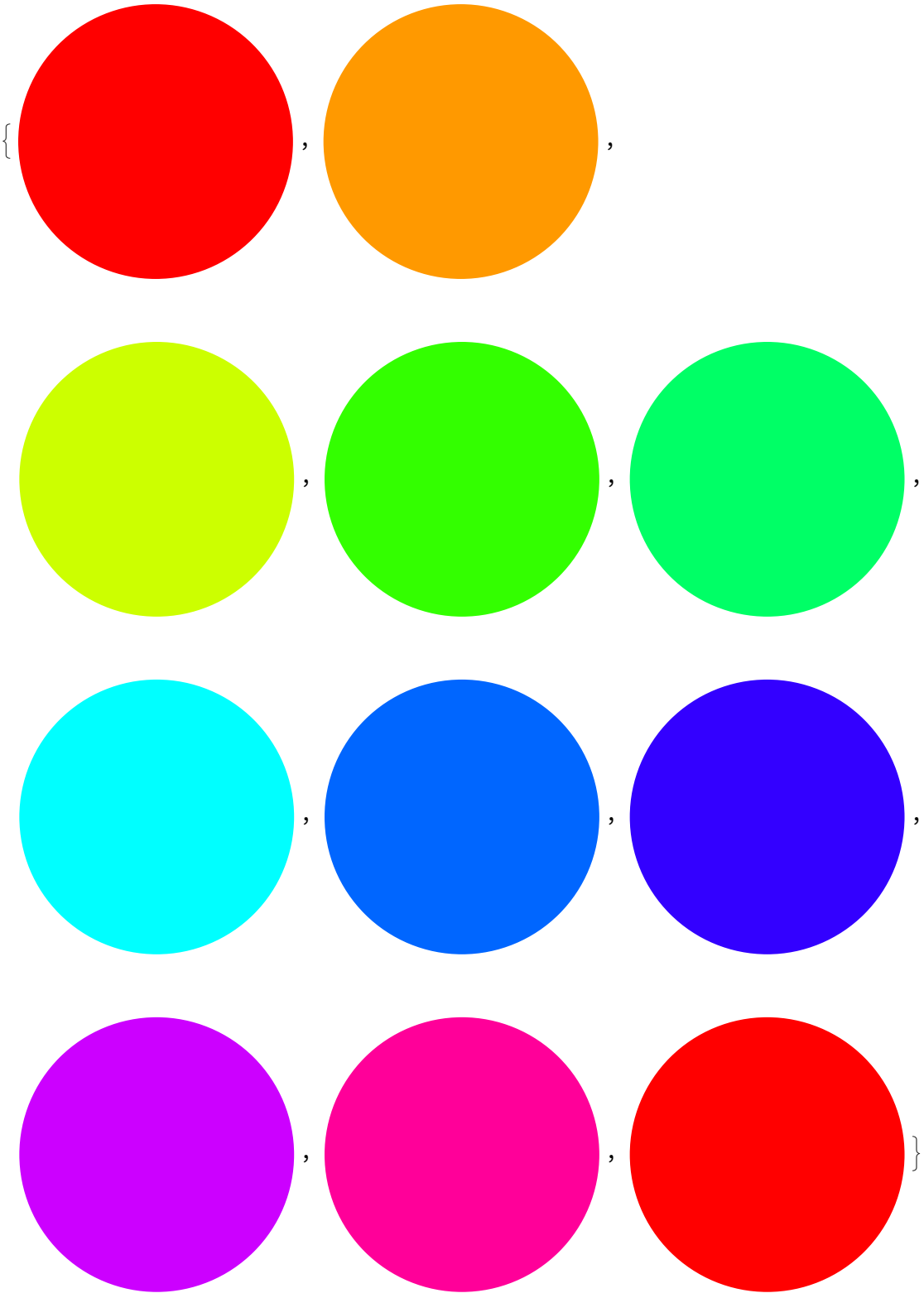
Out[ ]=



8.4 Make a list whose elements are disks with hues varying from 0 to 1 in steps of 0.1.

```
In[•]:= Table[Graphics[Style[Disk[], Hue[n]]], {n, 0, 1, 0.1}]
```

Out[•]=

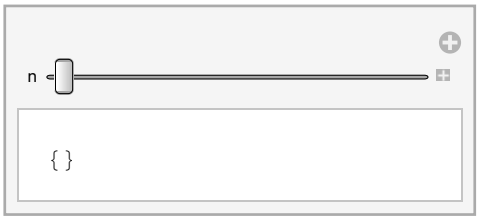


Exercise for Section 9 | Interactive Manipulation

9.1 Make a Manipulate to show Range[n] with n varying from 0 to 100.

In[ ]:=
```
Manipulate[Range[n], {n, 0, 100}]
```

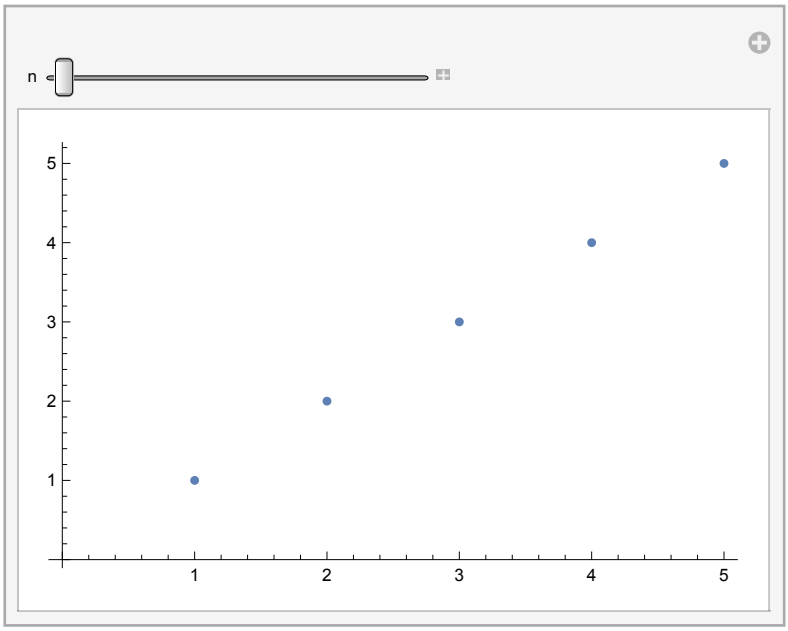
Out[ ]=



9.2 Make a Manipulate to plot the whole numbers up to n, where n can range from 5 to 50.

In[ ]:=
```
Manipulate[ListPlot[Range[1, n]], {n, 5, 50}]
```

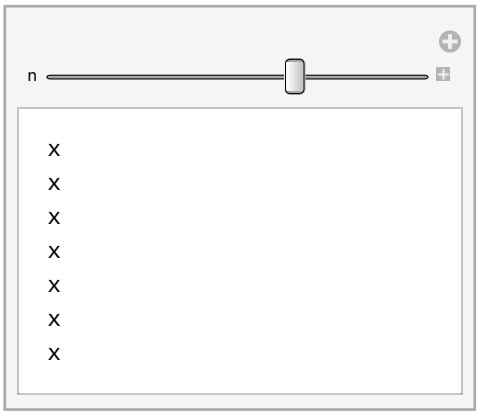
Out[ ]=



9.3 Make a Manipulate to show a column of between 1 and 10 copies of x.

In[ ]:=
```
Manipulate[Column[Table[x, n]], {n, 1, 10, 1}]
```

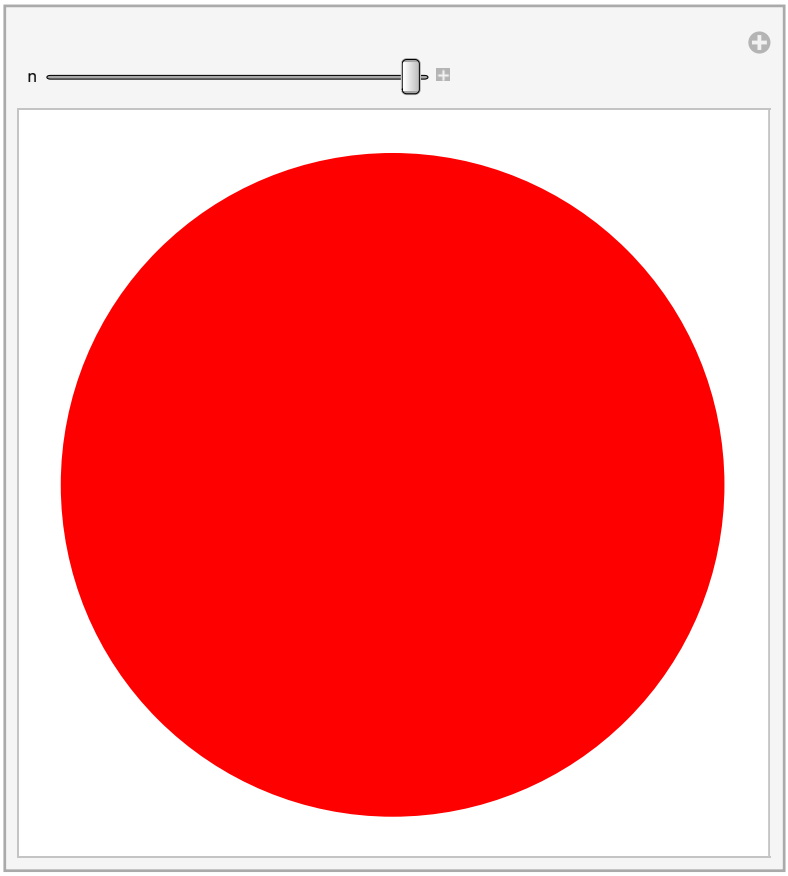
Out[ ]=



9.4 Make a Manipulate to show a disk with a hue varying from 0 to 1.

In[ ]:=
```
Manipulate[Graphics[Style[Disk[], Hue[n]]], {n, 0, 1}]
```

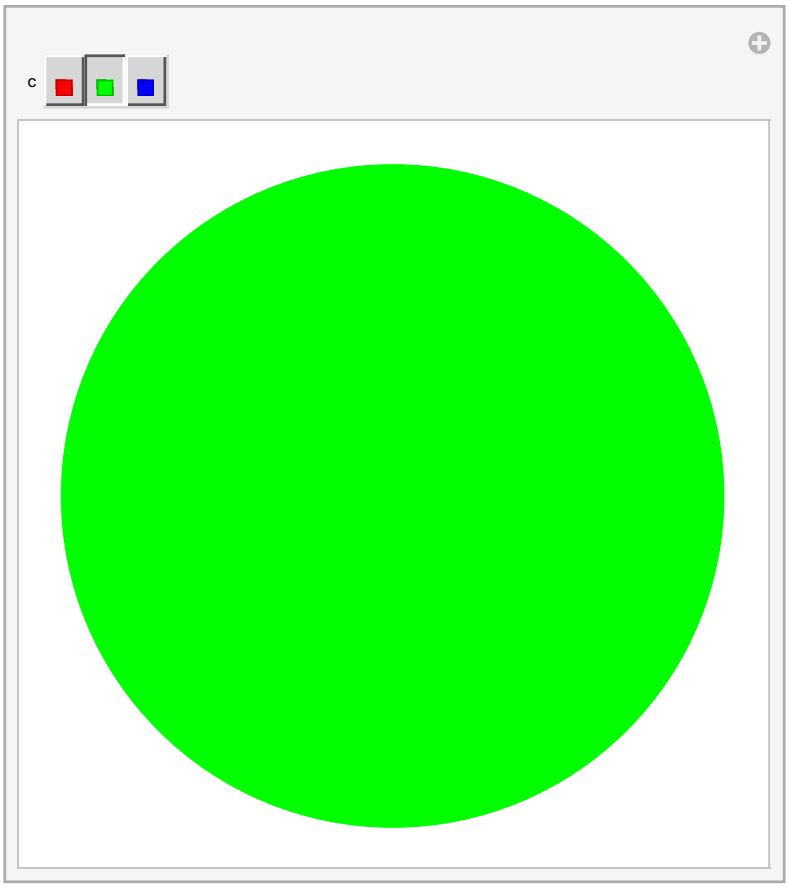
Out[ ]=



9.5 Make a manipulate to show a disk with red, green and blue color components varying from 0 to 1.

In[ ]:=
```
Manipulate[Graphics[Style[Disk[], c]], {c, {Red, Green, Blue}}]
```

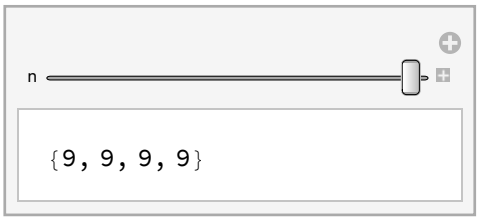
Out[ ]=



9.6 Make a Manipulate to show digit sequences of 4 - digit integers (between 1000 and 9999).

In[ ]:=
```
Manipulate[IntegerDigits[n], {n, 1000, 9999, 1}]
```

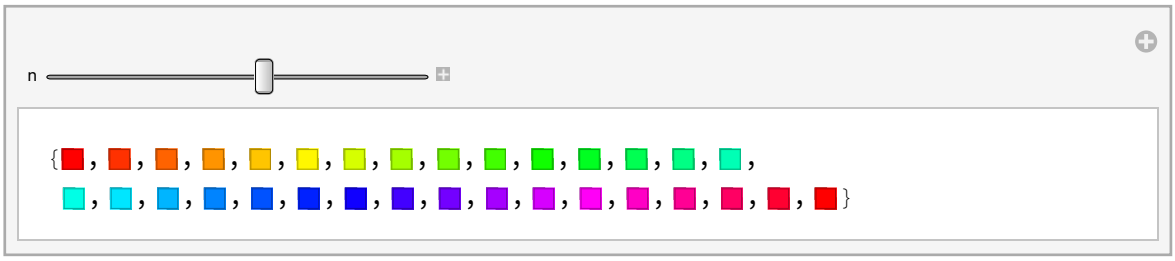
Out[ ]=



9.7 Make a Manipulate to create a list of between 5 and 50 equally spaced hues.

In[ ]:=
```
Manipulate[Table[Hue[h], {h, 0, 1, 1 / n}], {n, 5, 50, 1}]
```

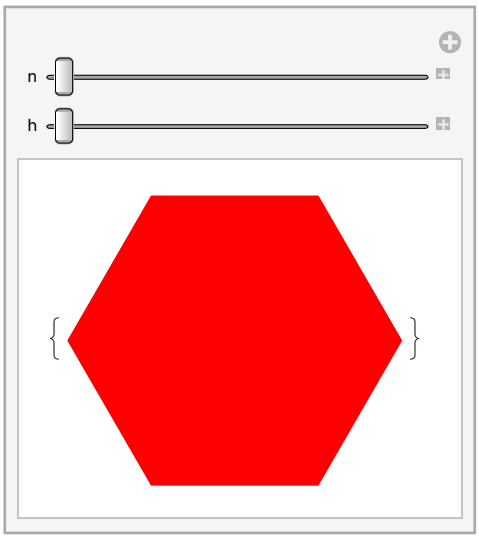
Out[ ]=



9.8 Make a Manipulate that shows a list of a variable number of hexagons (between 1 and 10), and with variable hues.

In[ ]:=
```
Manipulate[Table[Graphics[Style[RegularPolygon[6], Hue[h]]], n],
 {n, 1, 10, 1}, {h, 0, 1}]
```

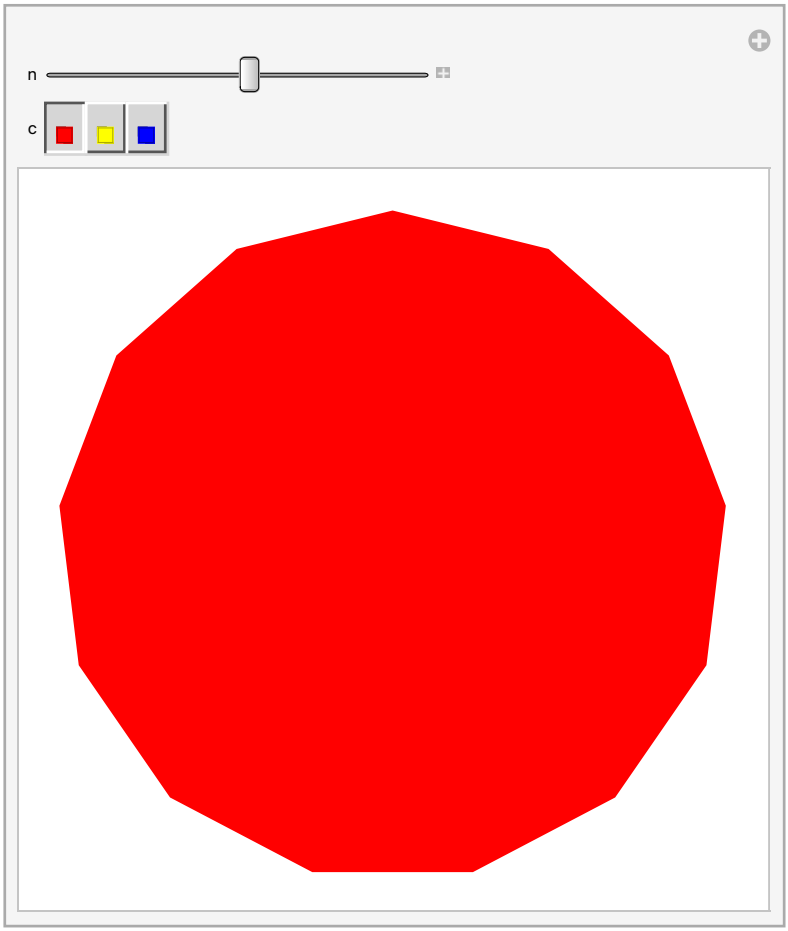
Out[ ]=



9.9 Make a Manipulate that lets you show a regular polygon with between 5 and 20 sides, in red, yellow or blue.

In[ ]:=
```
Manipulate[Graphics[Style[RegularPolygon[n], c]],
 {n, 5, 20, 1}, {c, {Red, Yellow, Blue}}]
```

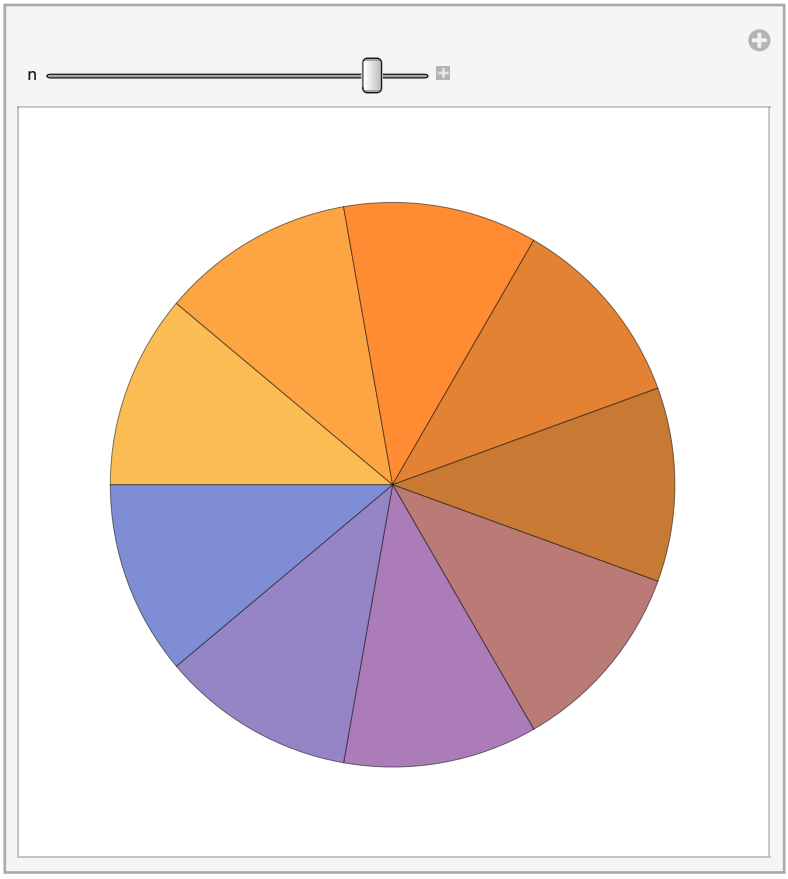
Out[ ]=



9.10 Make a Manipulate that shows a pie chart with a number of equal segments varying from 1 to 10.

In[ ]:=
```
Manipulate[PieChart[ConstantArray[1, n]], {n, 1, 10, 1}]
```

Out[ ]=



9.11 Make a Manipulate that gives a bar chart of the 3 digits in integers from 100 to 999.

In[ ]:=
```
Manipulate[BarChart[IntegerDigits[n]], {n, 100, 999, 1}]
```

Out[ ]=

9.12 Make a Manipulate that shows n random colors, where n can range from 1 to 50.

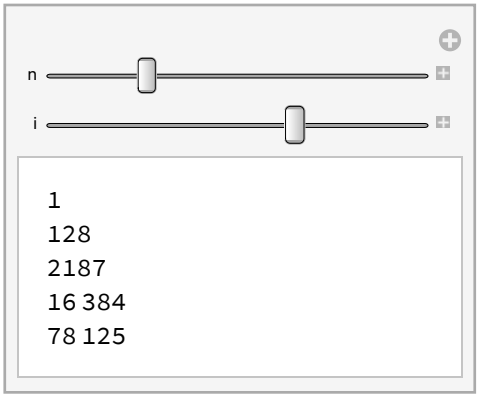
In[ ]:=
```
Manipulate[RandomColor[n], {n, 1, 50, 1}]
```

Out[ ]=

9.13 Make a Manipulate to display a column of integer powers with bases from 1 to 25 and exponents from 1 to 10.

In[ ]:=
```
Manipulate[Column[Range[n]^i], {n, 1, 20}, {i, 1, 10, 1}]
```

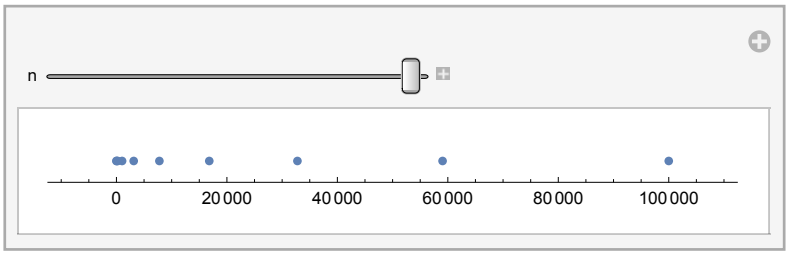
Out[ ]=



9.14 Make a Manipulate of a number line of values of x^n for integer x from 1 to 10, with n varying from 0 to 5.

In[ ]:=
```
Manipulate[NumberLinePlot[Range[10]^n], {n, 0, 5}]
```

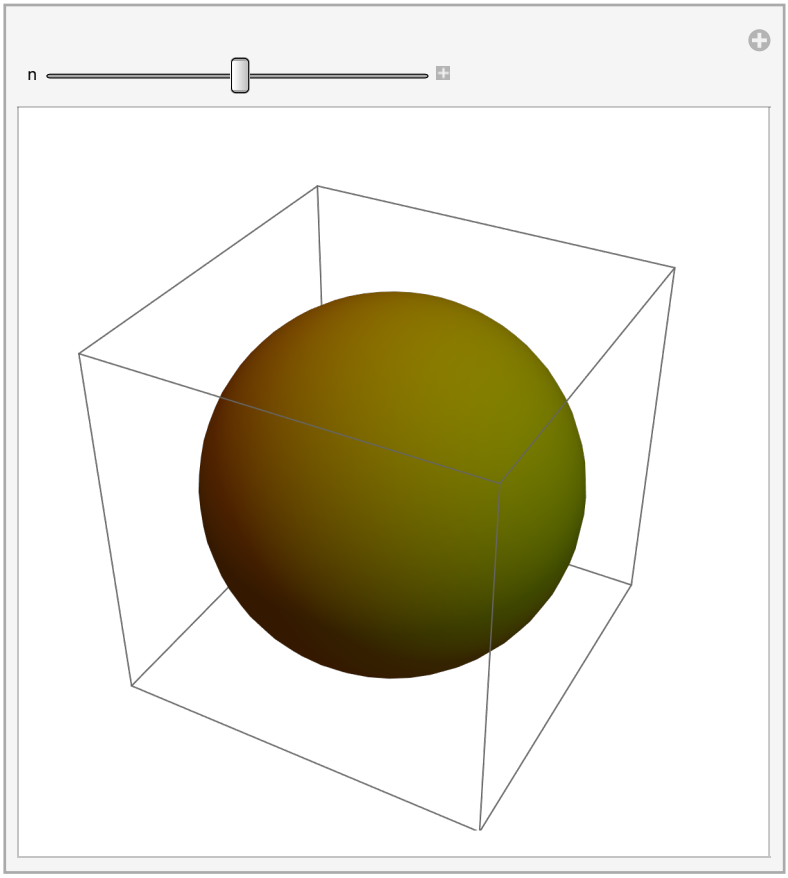
Out[ ]=



9.15 Make a Manipulate to show a sphere that can vary in color from green to red.

In[ ]:=
```
Manipulate[Graphics3D[Style[Sphere[], RGBColor[n, 1 - n, 0]]], {n, 0, 1}]
```

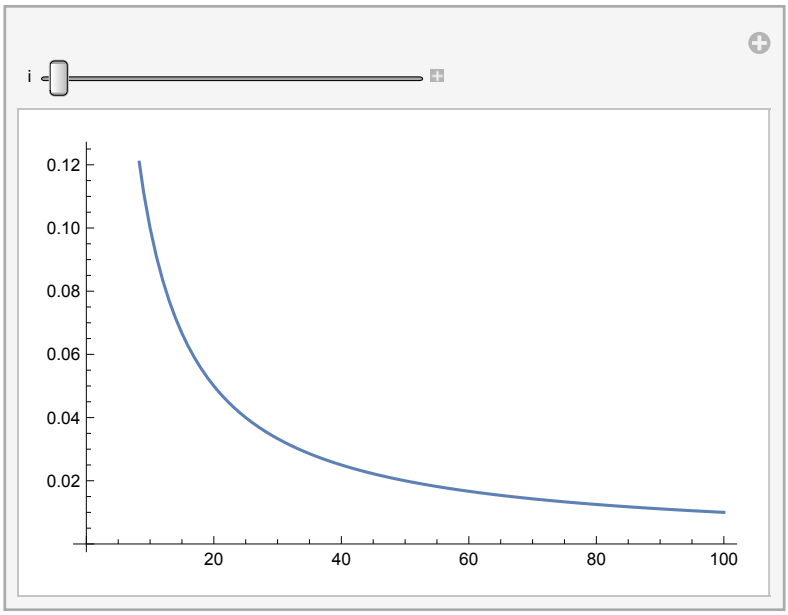
Out[ ]=



+9.1 Make a Manipulate to plot numbers from 1 to 100 raised to powers that can vary between - 1 and + 1.

In[ ]:=
```
Manipulate[ListLinePlot[Range[100]^i], {i, -1, 1}]
```

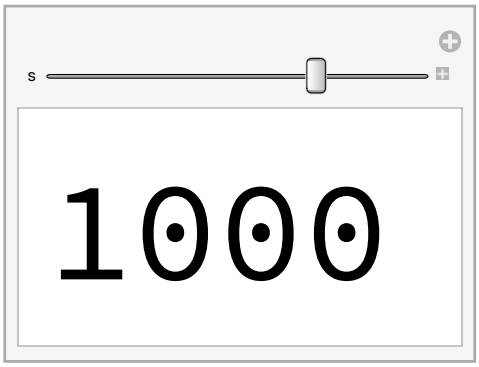
Out[ ]=



+9.2 Make a Manipulate to display 1000 at sizes between 5 and 100.

In[ ]:=
```
Manipulate[Style[1000, s], {s, 5, 100}]
```

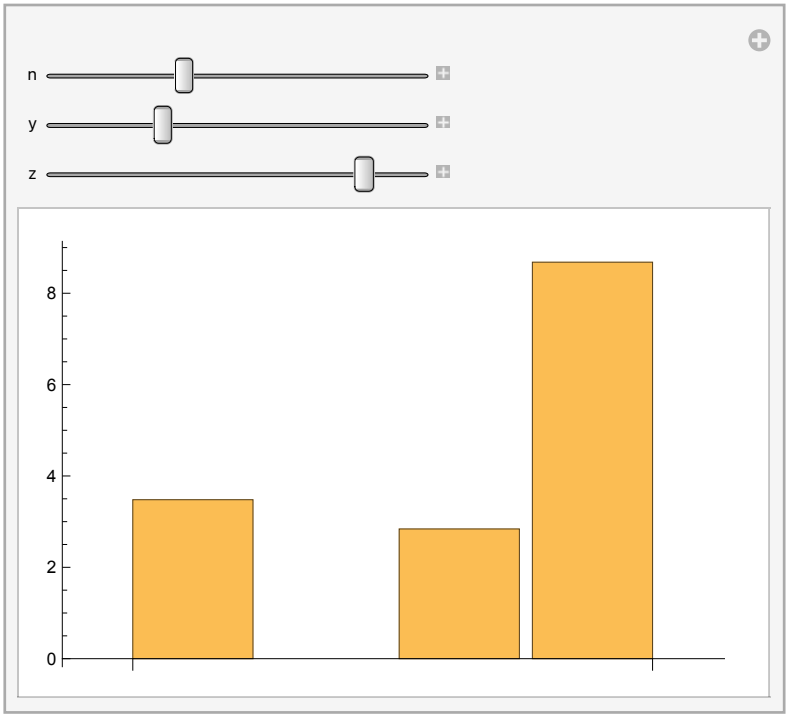
Out[ ]=



+9.3 Make a Manipulate to show a bar chart with 4 bars, each with a height that can be between 0 and 10.

In[ ]:=
```
Manipulate[BarChart[{n, x, y, z}], {n, 0, 10}, {y, 0, 10}, {z, 0, 10}]
```

Out[ ]=



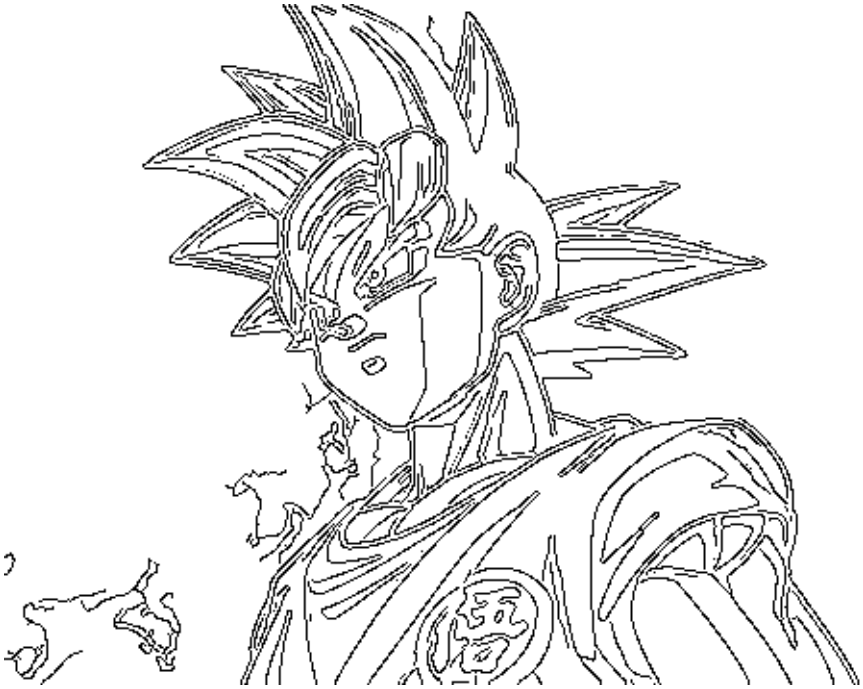
Exercises for Section 10 | Images

10.1 Color negate the result of edge detecting and image . (Use CurrentImage[] or any other image).

In[ ]:=
```
img =
```


Out[ ]=

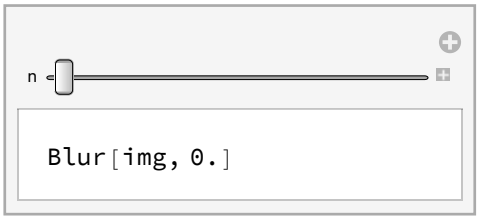

In[ ]:=
```
ColorNegate[EdgeDetect[img]]
```

Out[ ]=


10.2 Use Manipulate to make an interface for blurring and image from 0 to 20.

In[ ]:=
```
Manipulate[Blur[img, n], {n, 0, 20}]
```

Out[ ]=



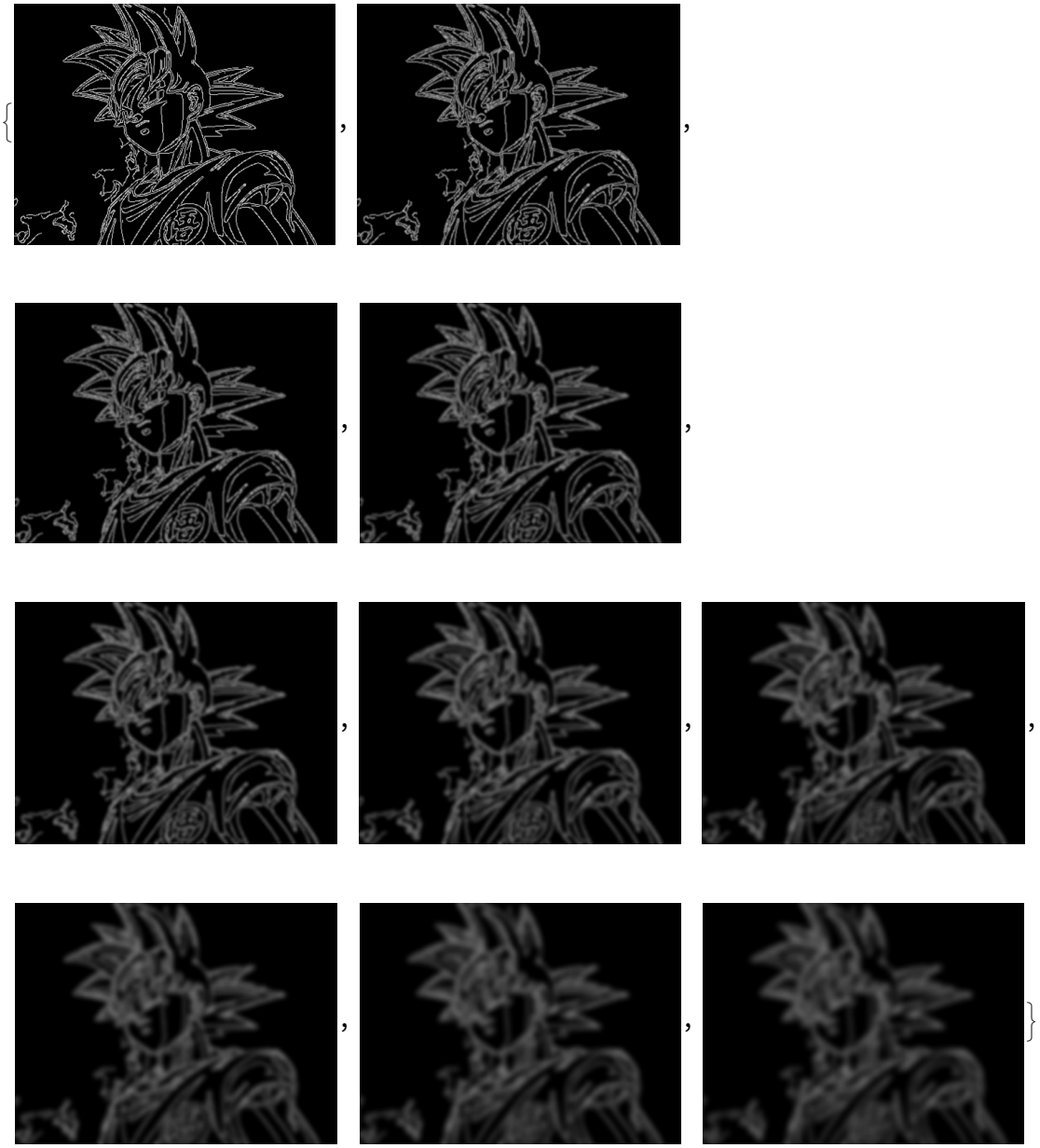
Blur : Expecting an image or graphics instead of img.

Blur : Expecting an image or graphics instead of img.

10.3 Make a table of the results from edge detecting an image with blurring from 1 to 10.

In[ ]:=
```
Table[Blur[EdgeDetect[img], n], {n, 1, 10}]
```

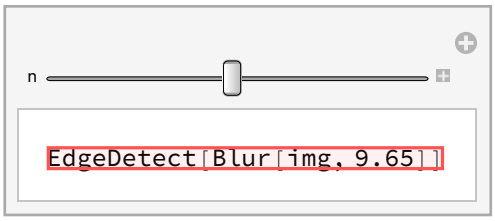
Out[ ]=



10.6 Create a Manipulate to display edges of an image as it gets blurred from 0 to 20.

```
In[ ]:= Manipulate[EdgeDetect[Blur[img, n]], {n, 0, 20}]
```

Out[ ]=



Blur : Expecting an image or graphics instead of img.

EdgeDetect : Expecting an image or graphics instead of Blur [img, 9.65].

Exercises for Section 11 | Strings and Text

11.1 Join two copies of the string "Hello"

```
In[ ]:= StringJoin["Hello", "Hello"]
Out[ ]= HelloHello
```

11.2 Make a single string of the whole alphabet, in uppercase.

```
In[ ]:= StringJoin[ToUpperCase[Alphabet[]]]
Out[ ]= ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

11.3 Generate a string of the alphabet in reverse order.

```
In[ ]:= StringReverse[StringJoin[Alphabet[]]]
Out[ ]= zyxwvutsrqponmlkjihgfedcba
```

11.4 Join 100 copies of the string "AGCT".

```
In[ ]:= StringJoin[ConstantArray["AGCT", 100]]
```

Out[ ]=

```
AGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAG
CTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTA
GCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCT
AGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGC
TAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAG
CTAGCTAGCTAGCTAGCTAGCT
```

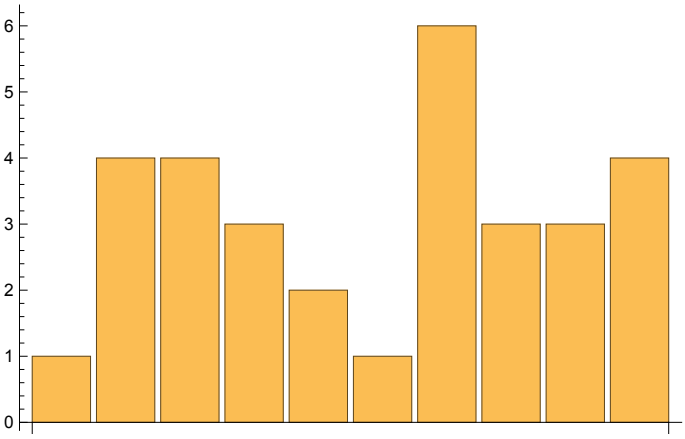
11.5 Use StringTake, StringJoin and Alphabet to get “abcdef” .

```
In[ ]:= StringTake[StringJoin[Alphabet[]], 6]
Out[ ]= abcdef
```

11.7 Make a bar chart of the lengths of the words in "A long time ago, in a galaxy far, far away".

```
In[ ]:= BarChart[
         StringLength[TextWords["A long time ago, in a galaxy far, far away."]]]
```

Out[ ]=



11.8 Find the string length of the Wikipedia article for "computer".

```
In[ ]:= StringLength[WikipediaData["computer"]]
```

```
Out[ ]= 57 696
```

11.9 Find how many words are in the Wikipedia article for “computer”.

```
In[ ]:= Length[TextWords[WikipediaData["computer"]]]
```

```
Out[ ]= 8892
```

11.10 Find the first sentence in the Wikipedia article about “strings”.

```
In[ ]:= First[TextSentences[WikipediaData["strings"]]]
```

```
Out[ ]= String or strings may refer to:
```

11.11 Make a string from the first letters of all sentences in the Wikipedia article about computers.

```
In[ ]:= StringJoin[StringTake[TextSentences[WikipediaData["computer"]], 1]]
```

```
Out[ ]= AMTATASCESEMTTTCTPP=ATTDBTTT==DTLTTTITIIMTTAAATTTIASIBIAITITIITS=CCATFTTTAEBNH=
        DHTTTTAB==BTDETTITIRT=PTEITDTTHACIINCTLOTIIIHBT==TTHTVTE=ECWATIHJTIIAAGTBAIT=
        TJFCJTHATHTTIWITT=TTDTKIHKNNHPNIMTGFTWISTIS=TTLTTTT=C=A=SH=TC==ATIET=WTTSC=
        TSC=TCATRDIRTPIWJSAIT=TES=TTSHTALTSG=AETTLSIETWOAMTTRACrRIIISFIIG=IDOHCIAMA=
        WTOBItSTBSIT=SMSTSS=SSCIW=T=TTMIAL=TITHTFMPWSTCBOTOI=ITTSTTITMWITC=PUTTS=MF=
        ATHHIT=PALTP=ETBOBSA=CTITTITCIIA"=AWA=TMH=TQCVSLTTT=ACARPE=AT====M
```

11.12 Find the maximum word length among English words from WordList[].

```
In[ ]:= Max[StringLength[WordList[]]]
Out[ ]= 23
```

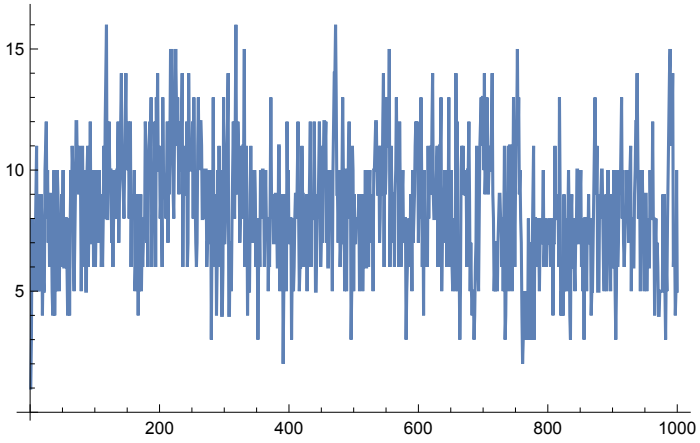
11.13 Count the number of words in WordList[] that start with “q”.

```
In[ ]:= Count[StringTake[WordList[], 1], "q"]
Out[ ]= 194
```

11.14 Make a line plot of the lengths of the first 1000 words from WordList[]

```
In[ ]:= ListLinePlot[StringLength[Take[WordList[], 1000]]]
```

Out[ ]=



11.15 Use StringJoin and Characters to make a word cloud of all letters in the words from WordList[].

In[ ]:=
```
WordCloud[Characters[StringJoin[WordList[]]]]
```

Out[ ]=

11.16 Use StringReverse to make a word cloud of the last letters in the words from WordList[].

In[ ]:=
```
WordCloud[StringTake[StringReverse[WordList[]], 1]]
```

Out[ ]=

11.17 Find the Roman numerals for the year 1959.

```
In[ ]:= RomanNumeral[1959]
Out[ ]= MCMLIX
```

11.18 Find the maximum string length of any Roman-numeral year from 1 to 2020.

```
In[ ]:= Max[StringLength[RomanNumeral[Range[2020]]]]
Out[ ]= 13
```

11.19 Make a word cloud from the first characters of the Roman numerals up to 100.

```
In[ ]:= WordClod[StringTake[RomanNumeral[Range[100]], 1]]
Out[ ]= WordClod[{I, I, I, I, V, V, V, V, I, X, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
    L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L,
    L, L, L, L, L, L, L, L, L, L, L, L, L, X, X, X, X, X, X, X, X, X, C}]
```

11.20 Use Length to find the length of the Russian alphabet.

```
In[ ]:= Length[Alphabet["Russian"]]
Out[ ]= 33
```

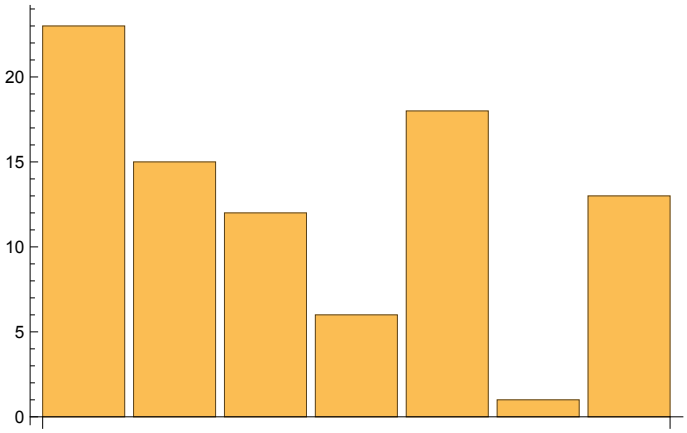
11.21 Generate the uppercase Greek alphabet.

```
In[ ]:= ToUpperCase[Alphabet["Greek"]]
Out[ ]= {Α, Β, Γ, Δ, Ε, Ζ, Η, Θ, Ι, Κ, Λ, Μ, Ν, Ξ, Ο, Π, Ρ, Σ, Τ, Υ, Φ, Χ, Ψ, Ω}
```

11.22 Make a bar chart of the letter numbers in “wolfram”

```
In[ ]:= BarChart[LetterNumber["wolfram"]]
```

Out[ ]=



11.23 Use FromLetterNumber to make a string of 1000 random letters.

In[ ]:=
```
StringJoin[FromLetterNumber[Table[RandomInteger[25] + 1, 1000]]]
```

Out[ ]=
```
xqyundcqtfebfucsaghlafrzxkueptkkcvjwmhqsnqyitnilzjaijyvkbtnqfkyecmivkttslvlnjg
suaoidqhiduastrdhrgfxacgsvjnpjmlmhzbkbqccdxyhiarfbupymjllqlplrkxznunypvrpea
jvyjtwqmtrkwzrpegpahajepfgebbzsxhglmojtuokvnwcforgqhxmqdqiwhjuqqpfxlcqzevbi
ezvhmkfanbjjcfzfwyhuigjnezdwxwgulmwlthilccupjztajmtklgicdeecddabpsowxxgdufr
fwjstbqkfrogkfmoqisxxlnwsyukowmwmmqjcyjthylvyahxhvwufnrvvjmhtqvgvkddjzkjnui
apbhbqrjzrelrbcubcvanxcbgklnlxmqrdsehkuauuztfsxoftkvuxlcfjjgjivdqhzlnlprlow
xxponrdiarkyhbxjtnnbpgngzbeizvmwzrxnltyxydqicfzqgahrhhjeaeynjahlwrcuostyzli
ongqkunkallrczsfbuzsodcxntpwmvyiexujpldmluwlcdgksosmboryhtdipcqnycbbttrogxl
ouzsdynibgujshijptfimtxqveuxrmrixgywvziwwxgfbntmjvwjfjczusfrmerlterarzoggwm
kknrgprjqsmkwocatqfwkhvutrcuchhpkmthhpdqplfrsqmdxcrzjrdfafsasfmziitcdsbcsvb
xozudofmhxnqxytetmipzeuzxvvylrjczsczcbczpkzpukiklhyvpklrlucqdbhodlcqqbwvhgg
izgkjumgagzvkahzjwkraqxdxcshcwjeharpkptixjoxbxeduylryriqssqzduwmxgalhzrwytv
lnxcstpojwkuilybdrkkiuomfswxepxkpqfzzlqkxcpolgitaqfxpgrvlsjosxplsmkmzvvvful
eewclpbirslgqfxczouqkq
```

11.24 Make a list of 100 random 5-letter stirngs.

In[ ]:=
```
Table[StringJoin[Table[FromLetterNumber[RandomInteger[25] + 1], 5]], 100]
```

Out[ ]=
```
{dxlhm, scqvc, cptot, eulwu, qlrua, dftym, ccskn, zphsm, tsqns, dxnpj, iijpc, cnemu,
 zexei, kdnxo, qbsgl, mmdyy, rbnqw, dirkv, vtsaa, iteli, qygth, ptvuo, bhsms,
 igxkz, tsnet, shwul, zodul, wovdl, iyoqp, yihzt, thrwr, bghqa, tbznq, buyyf,
 vfuwj, rwzrp, egbhr, ssxzl, vaslb, vqbtr, bdrcc, fgmgw, knhhg, dxgxl, cthbd,
 nefwx, tvbav, qnrgn, fkaga, hmuym, swtdt, emjiy, bcwen, motdd, mdlvr, erkvw,
 hqloz, qfebg, pzujl, sqmlg, jsqdd, mzxmy, rfnbh, rhekl, vimjh, iqhdw, hcnvg,
 iyens, luxne, zcusk, agfvi, xufcs, pdukm, vztfq, zjemu, vjhzw, flxyk, wnfrt,
 hxctu, rucyz, rjvny, okksl, uuuen, khqdt, eaexm, hkdia, xfrps, coedd, hlexb,
 tnkdb, wniit, bjbkl, agmrb, aclrk, pspdb, orgdg, nojzt, ddxdc, ltlhw, zmoxg}
```

11.25 Transliterate “wolfram” into Greek.

In[ ]:=
```
Transliterate["wolfram", "Greek"]
```

Out[ ]=
```
βολφραμ
```

11.26 Get the Arabic alphabet and transliterate it into English.

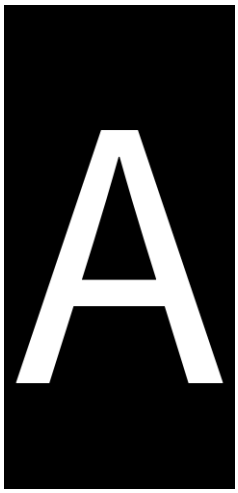
In[ ]:=
```
Transliterate[Alphabet["Arabic"]]
```

Out[ ]=
```
{a, b, t, th, j, h, kh, d, dh, r, z, s, sh, s, d, t, z, ʿ, gh, f, q, k, l, m, n, h, w, y}
```

11.27 Make a white-on-black size-200 letter “A”.

In[ ]:=
```
ColorNegate[Rasterize[Style["A", 200]]]
```

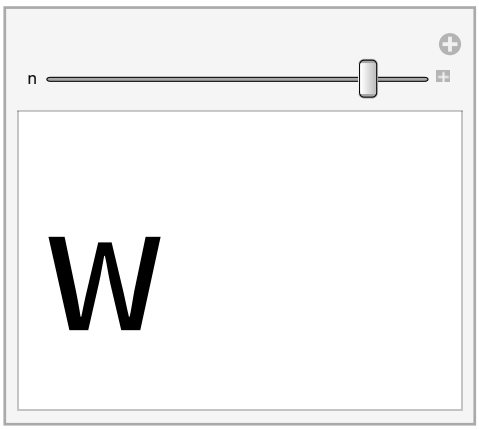
Out[ ]=



11.28 Use Manipulate to make an interactive selector of size-100 characters from the alphabet, controlled by a slider.

In[ ]:=
```
Manipulate[Style[FromLetterNumber[n], 100], {n, 1, Length[Alphabet[]], 1}]
```

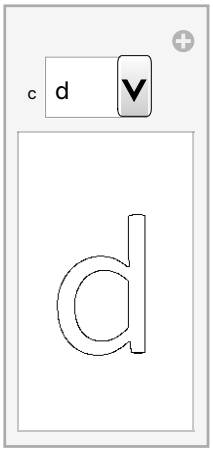
Out[ ]=



11.29 Use Manipulate to make an interactive selector of black-on-white outlines of rasterized size-100 characters from the alphabet, controlled by a menu.

```
In[ ]:= Manipulate[ColorNegate[EdgeDetect[Rasterize[Style[c, 100]]]], {c, Alphabet[]}]
```

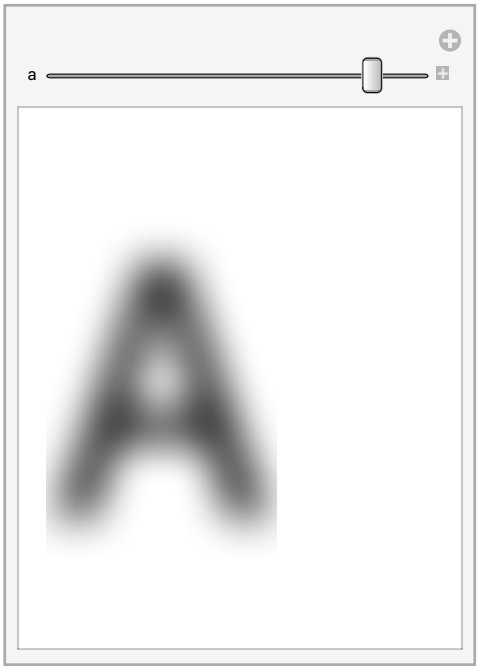
Out[ ]=



11.30 Use Manipulate to create a “vision simulator” that blurs a size-200 letter “A” by an amount from 0 to 50.

```
In[ ]:= Manipulate[Blur[Rasterize[Style["A", 200]], a], {a, 0, 50}]
```

Out[ ]=



+11.1 Generate a string of the alphabet followed by the alphabet written in reverse.

```
In[ ]:= StringJoin[StringJoin[Alphabet[]], StringReverse[StringJoin[Alphabet[]]]]
Out[ ]= abcdefghijklmnopqrstuvwxyzzyxwvutsrqponmlkjihgfedcba
```

+11.3 Find how many sentences are in the Wikipedia article for “computer”.

```
In[ ]:= Length[TextSentences[WikipediaData["computer"]]]
Out[ ]= 449
```

+11.4 Join together without spaces, etc. the words in the first sentence in the Wikipedia article for “strings”

```
In[ ]:= StringJoin[TextWords[First[TextSentences[WikipediaData["strings"]]]]]
Out[ ]= Stringorstringsmayreferto
```

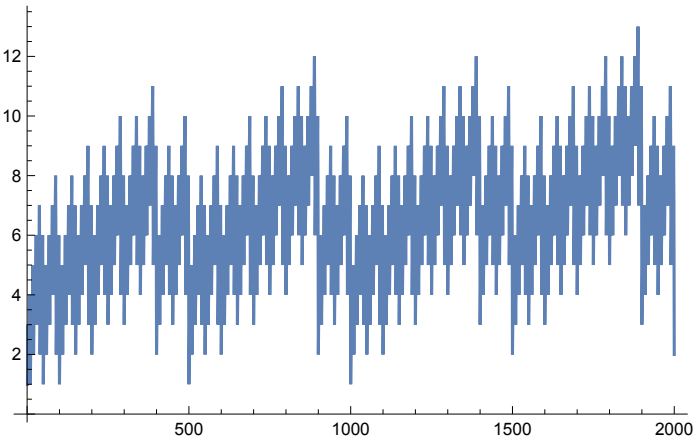
+11.5 Find the length of the longest word in the Wikipedia article about computers.

```
In[ ]:= Max[StringLength[TextWords[WikipediaData["computers"]]]]
Out[ ]= 25
```

+11.6 Plot the lengths of Roman numerals for numbers up to 2000.

```
In[ ]:= ListLinePlot[StringLength[RomanNumeral[Range[2000]]]]
```

Out[ ]=



+11.7 Generate a string by joining the Roman numerals up to 100.

```
In[ ]:= StringJoin[RomanNumeral[Range[100]]]
```

```
Out[ ]= IIIIIIIVVVIVIIVIIIIXXXIXIIXIIIXIVXVXVIXVIIXVIIIXIXXXXXIXXIIXXIIIXXIVXXVXXVIXXV↕
        IIXXVIIIXXIXXXXXXXIXXXIIXXXIIIXXXIVXXXVXXXVIXXXVIIXXXVIIIXXXIXXLXLIXLIIXLII↕
        IXLIVXLVXLVIXLVIIXLVIIIXLIXLLILIILIIILIVLVLVILVIILVIIILIXLXLXILXIILXIIILXIV↕
        LXVLXVILXVIILXVIIILXIXLXXLXXILXXIILXXIIILXXIVLXXVLXXVILXXVIILXXVIIILXXIXLXX↕
        XLXXXILXXXIILXXXIIILXXXIVLXXXVLXXXVILXXXVIILXXXVIIILXXXIXXCXCIXCIIXCIIIXCIV↕
        XCVXCVIXCVIIXCVIIIXCIXC
```

+11.9 Find the maximum string length of the name of any integer up to 1000.

```
In[ ]:= Max[StringLength[IntegerName[Range[1000]]]]
Out[ ]= 27
```

+11.10 Make a list of uppercase size-20 letters of the alphabet in random colors.

```
In[ ]:= Style[ToUpperCase[Alphabet[]], 20, RandomColor[]]
Out[ ]= {A, B, C, D, E, F, G, H, I, J, K, L,
          M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
```

+11.11 Make a list of 100 random 5-letter strings with the Russian alphabet.

```
In[ ]:= Table[
         StringJoin[Table[FromLetterNumber[RandomInteger[25] + 1, "Russian"], 5]], 100]
Out[ ]= {цшнох, лбпоё, ёлжно, йохпе, лфзёц, оечдг, жпзир, ёопгр, жалиб, бцдсг, снкбн, бавцх,
         ушкхё, вкубт, фйане, ацмйм, сфшзк, дхуйш, фмрйа, хфкто, пдикм, тлшбу, рблёс,
         ебихч, ёвжех, кёпое, вкйгл, гншгп, лзхвц, рбрпз, уддда, шдбиу, ииаёц, бееум,
         дзцум, мфрчф, винсж, швнаш, пижгё, дмвог, метех, кйшоу, цзкеп, днфри, лйагж,
         чфйгб, лбкнё, дпрсф, гуофн, ииаро, лорпн, анкзв, клдут, ехедй, ётзшп, дмзмп,
         кнлхт, зфлйж, цёхгё, хггфч, зтфбк, пштчх, оезчш, тесчж, пйбхз, мштно, хдвзб,
         фиуйм, иоужч, жчупц, вркзж, вцчнд, чёзвп, шмткч, всфшд, зфхтд, емжзё, тйшбб,
         уфжап, цучач, нфмжс, хчусг, ччнбн, нцчшр, цжатц, йёццу, хеевх, ёвклг, дшопж,
         вшнсх, ёйбжк, рёрсг, алдлл, утзшр, мпссг, леффе, цфчжт, увнлз, гуйфл, мврйе}
```

Exercises for Section 13 | Arrays, or Lists of Lists

13.1 Make a 12x12 multiplication table.

```
In[ ]:= Grid[Table[i * k, {i, 1, 12}, {k, 1, 12}]]
Out[ ]=
 1  2  3  4  5  6  7  8  9  10  11  12
 2  4  6  8  10 12 14 16 18 20  22  24
 3  6  9  12 15 18 21 24 27 30  33  36
 4  8  12 16 20 24 28 32 36 40  44  48
 5  10 15 20 25 30 35 40 45 50  55  60
 6  12 18 24 30 36 42 48 54 60  66  72
 7  14 21 28 35 42 49 56 63 70  77  84
 8  16 24 32 40 48 56 64 72 80  88  96
 9  18 27 36 45 54 63 72 81 90  99  108
 10 20 30 40 50 60 70 80 90 100 110 120
 11 22 33 44 55 66 77 88 99 110 121 132
 12 24 36 48 60 72 84 96 108 120 132 144
```

13.2 Make a 5x5 multiplication table for Roman numerals.

In[ ]:=
```
Grid[Table[RomanNumeral[i * j], {i, 1, 5}, {j, 1, 5}]]
```

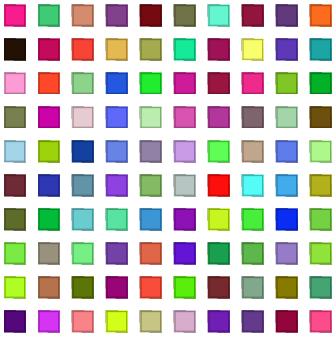
Out[ ]=

I	II	III	IV	V
II	IV	VI	VIII	X
III	VI	IX	XII	XV
IV	VIII	XII	XVI	XX
V	X	XV	XX	XXV

13.3 Make a 10x10 grid of random colors.

In[ ]:=
```
Grid[Table[RandomColor[], 10, 10]]
```

Out[ ]=



13.4 Make a 10x10 grid of randomly colored random integers between 0 and 10.

In[ ]:=
```
Grid[Table[Style[RandomInteger[10], RandomColor[]], 10, 10]]
```

Out[ ]=

4	3	3	0	9	4	5	5	5	6
1	1	3	1	10	2	5	9	4	6
9	0	4	7	6	2	8	6	8	10
7	1	2	9	1	8	7	1	5	9
5	6	3	3	3	4	4	3	9	2
2	6	9	0	0	9	6	1	5	2
8	5	5	5	5	0	3	0	8	3
4	1	2	7	8	1	1	7	6	4
7	10	9	4	3	8	0	6	3	4
6	10	8	7	9	4	5	2	5	3

13.5 Make a grid of all possible strings consisting of pairs of letters of the alphabet (“aa”, ”ab”, etc.).

In[•]:=
```
Grid[Table[StringJoin[FromLetterNumber[i], FromLetterNumber[j]],
  {i, 1, Length[Alphabet[]]}, {j, 1, Length[Alphabet[]]}]]
```

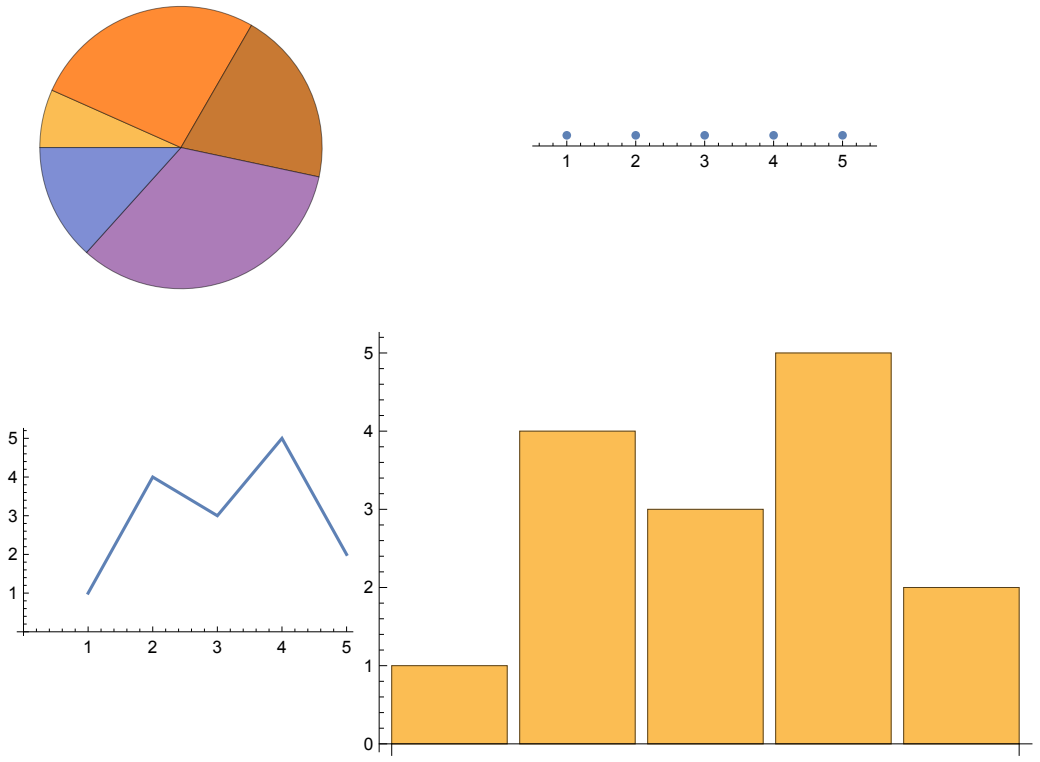
Out[•]=

aa	ab	ac	ad	ae	af	ag	ah	ai	aj	ak	al	am	an	ao	ap	aq	ar	as	at	au	av	aw	ax	ay	az
ba	bb	bc	bd	be	bf	bg	bh	bi	bj	bk	bl	bm	bn	bo	bp	bq	br	bs	bt	bu	bv	bw	bx	by	bz
ca	cb	cc	cd	ce	cf	cg	ch	ci	cj	ck	cl	cm	cn	co	cp	cq	cr	cs	ct	cu	cv	cw	cx	cy	cz
da	db	dc	dd	de	df	dg	dh	di	dj	dk	dl	dm	dn	do	dp	dq	dr	ds	dt	du	dv	dw	dx	dy	dz
ea	eb	ec	ed	ee	ef	eg	eh	ei	ej	ek	el	em	en	eo	ep	eq	er	es	et	eu	ev	ew	ex	ey	ez
fa	fb	fc	fd	fe	ff	fg	fh	fi	fj	fk	fl	fm	fn	fo	fp	fq	fr	fs	ft	fu	fv	fw	fx	fy	fz
ga	gb	gc	gd	ge	gf	gg	gh	gi	gj	gk	gl	gm	gn	go	gp	gq	gr	gs	gt	gu	gv	gw	gx	gy	gz
ha	hb	hc	hd	he	hf	hg	hh	hi	hj	hk	hl	hm	hn	ho	hp	hq	hr	hs	ht	hu	hv	hw	hx	hy	hz
ia	ib	ic	id	ie	if	ig	ih	ii	ij	ik	il	im	in	io	ip	iq	ir	is	it	iu	iv	iw	ix	iy	iz
ja	jb	jc	jd	je	jf	jg	jh	ji	jj	jk	jl	jm	jn	jo	jp	jq	jr	js	jt	ju	jv	jw	jx	jy	jz
ka	kb	kc	kd	ke	kf	kg	kh	ki	kj	kk	kl	km	kn	ko	kp	kq	kr	ks	kt	ku	kv	kw	kx	ky	kz
la	lb	lc	ld	le	lf	lg	lh	li	lj	lk	ll	lm	ln	lo	lp	lq	lr	ls	lt	lu	lv	lw	lx	ly	lz
ma	mb	mc	md	me	mf	mg	mh	mi	mj	mk	ml	mm	mn	mo	mp	mq	mr	ms	mt	mu	mv	mw	mx	my	mz
na	nb	nc	nd	ne	nf	ng	nh	ni	nj	nk	nl	nm	nn	no	np	nq	nr	ns	nt	nu	nv	nw	nx	ny	nz
oa	ob	oc	od	oe	of	og	oh	oi	oj	ok	ol	om	on	oo	op	oq	or	os	ot	ou	ov	ow	ox	oy	oz
pa	pb	pc	pd	pe	pf	pg	ph	pi	pj	pk	pl	pm	pn	po	pp	pq	pr	ps	pt	pu	pv	pw	px	py	pz
qa	qb	qc	qd	qe	qf	qg	qh	qi	qj	qk	ql	qm	qn	qo	qp	qq	qr	qs	qt	qu	qv	qw	qx	qy	qz
ra	rb	rc	rd	re	rf	rg	rh	ri	rj	rk	rl	rm	rn	ro	rp	rq	rr	rs	rt	ru	rv	rw	rx	ry	rz
sa	sb	sc	sd	se	sf	sg	sh	si	sj	sk	sl	sm	sn	so	sp	sq	sr	ss	st	su	sv	sw	sx	sy	sz
ta	tb	tc	td	te	tf	tg	th	ti	tj	tk	tl	tm	tn	to	tp	tq	tr	ts	tt	tu	tv	tw	tx	ty	tz
ua	ub	uc	ud	ue	uf	ug	uh	ui	uj	uk	ul	um	un	uo	up	uq	ur	us	ut	uu	uv	uw	ux	uy	uz
va	vb	vc	vd	ve	vf	vg	vh	vi	vj	vk	vl	vm	vn	vo	vp	vq	vr	vs	vt	vu	vv	vw	vx	vy	vz
wa	wb	wc	wd	we	wf	wg	wh	wi	wj	wk	wl	wm	wn	wo	wp	wq	wr	ws	wt	wu	wv	ww	wx	wy	wz
xa	xb	xc	xd	xe	xf	xg	xh	xi	xj	xk	xl	xm	xn	xo	xp	xq	xr	xs	xt	xu	xv	xw	xx	xy	xz
ya	yb	yc	yd	ye	yf	yg	yh	yi	yj	yk	yl	ym	yn	yo	yp	yq	yr	ys	yt	yu	yv	yw	yx	yy	yz
za	zb	zc	zd	ze	zf	zg	zh	zi	zj	zk	zl	zm	zn	zo	zp	zq	zr	zs	zt	zu	zv	zw	zx	zy	zz

13.6 Visualize {1, 4, 3, 5, 2} with a pie chart, number line, line plot and bar chart. Place these in a 2x2 grid.

In[ ]:=
```
Grid[{{PieChart[{1, 4, 3, 5, 2}], NumberLinePlot[{1, 4, 3, 5, 2}]},
  {ListLinePlot[{1, 4, 3, 5, 2}], BarChart[{1, 4, 3, 5, 2}]}}]
```

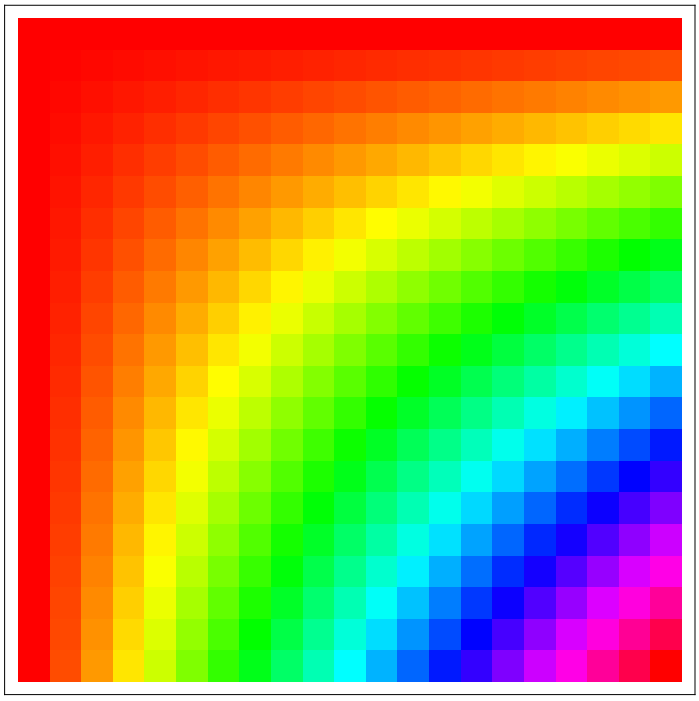
Out[ ]=



13.7 Make an array plot of hue values x*y, where x and y each run from 0 to 1 in steps of 0.05.

In[ ]:=
```
ArrayPlot[Table[Hue[x * y], {x, 0, 1, 0.05}, {y, 0, 1, 0.05}]]
```

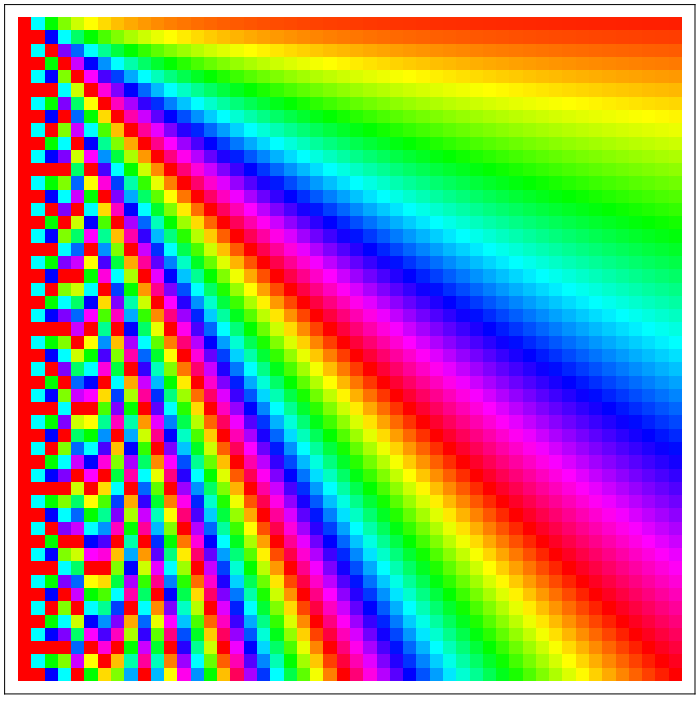
Out[ ]=



13.8 Make an array plot of hue values x/y, where x and y each run from 1 to 50 in steps of 1.

In[ ]:=
```
ArrayPlot[Table[Hue[x / y], {x, 1, 50, 1}, {y, 1, 50, 1}]]
```

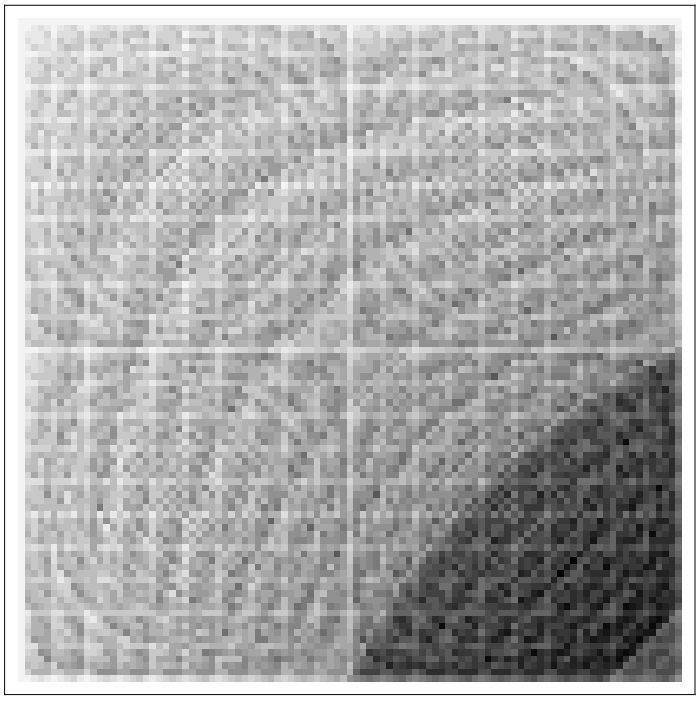
Out[ ]=



13.9 Make an array plot of the lengths of Roman numeral strings in a multiplication table up to 100x100.

In[ ]:=
```
ArrayPlot[Table[Length[Characters[RomanNumeral[i * j]]], {i, 0, 100}, {j, 0, 100}]]
```

Out[ ]=



+13.1 Make a 20x20 addition table.

In[ ]:=
```
Grid[Table[x + y, {x, 1, 20}, {y, 1, 20}]]
```

Out[ ]=

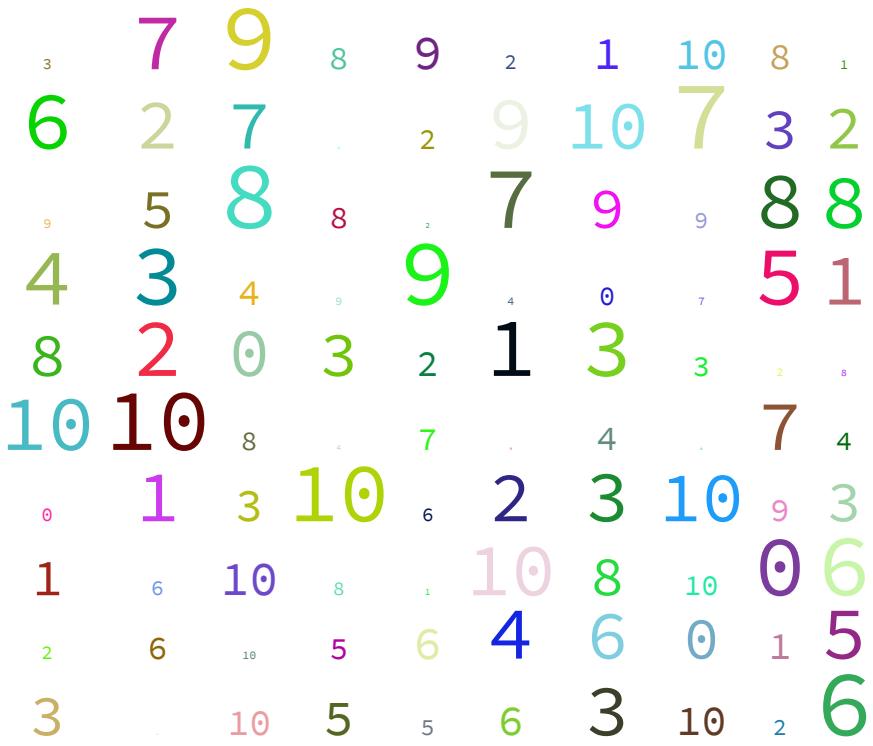
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40

+13.2 Make a 10x10 grid of randomly colored random integers between 0 and

10 that have random size up to 32.

In[ ]:=
```
Grid[Table[Style[RandomInteger[10], RandomColor[], RandomInteger[52]], 10, 10]]
```

Out[ ]=

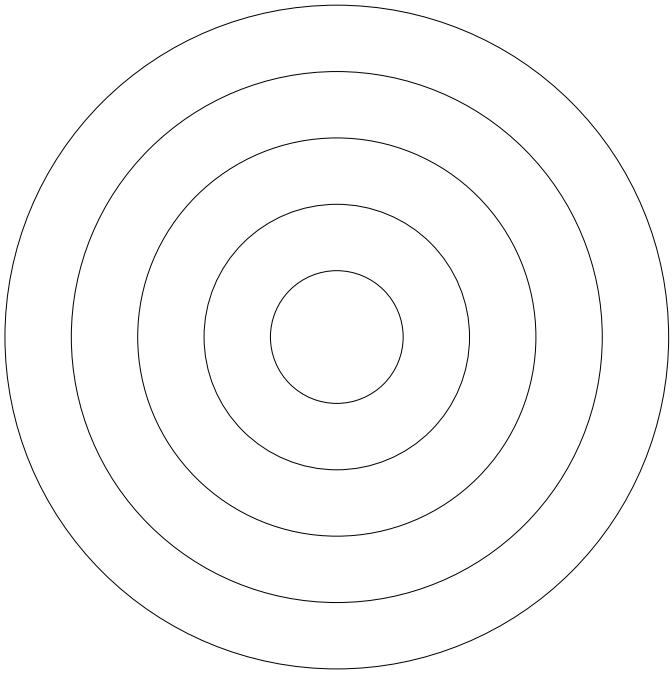


Exercises for Section 14 | Coordinates and Graphics

14.1 Make a graphics of 5 concentric circles centered at {0, 0} with radio 1, 2, ..., 5.

In[ ]:=
```
Graphics[Table[Circle[{0, 0}, r], {r, 1, 5}]]
```

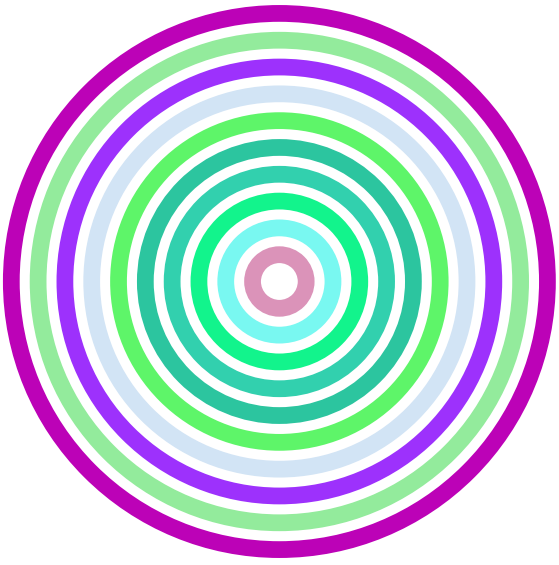
Out[ ]=



14.2 Make 10 concentric circles with random colors.

In[ ]:=
```
Graphics[
 Table[Style[Circle[{0, 0}, r], RandomColor[], Thickness[0.03]], {r, 1, 10}]]
```

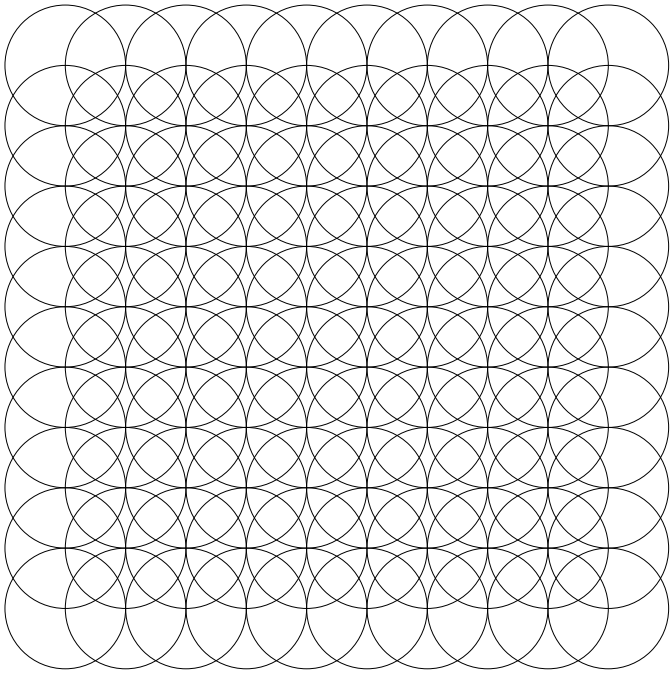
Out[ ]=



14.3 Make graphics of a 10x10 grid of circles with radius 1 centered at integer points {x, y}.

In[ ]:=
```
Graphics[Table[Circle[{x, y}, 1], {x, 1, 10}, {y, 1, 10}]]
```

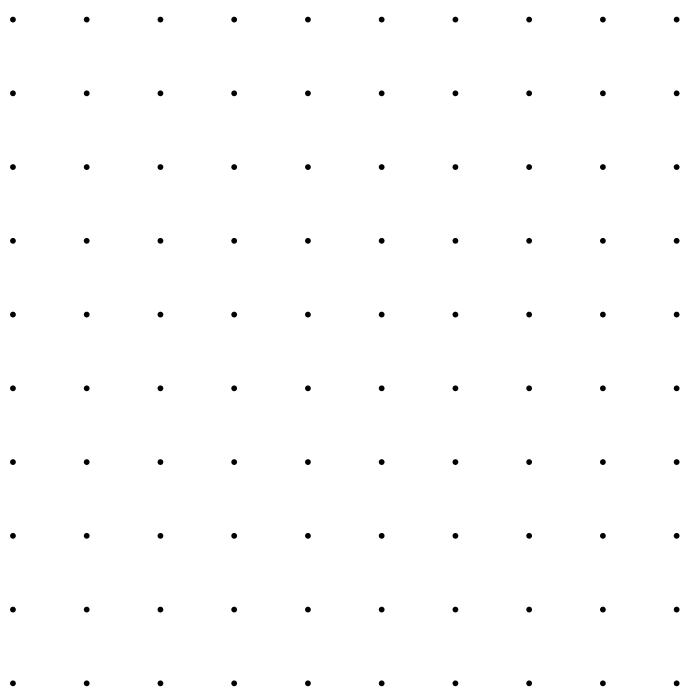
Out[ ]=



14.4 Make a 10x10 grid of points with coordinates at integer positions up to 10.

In[ ]:=
```
Graphics[Table[Point[{x, y}], {x, 10}, {y, 10}]]
```

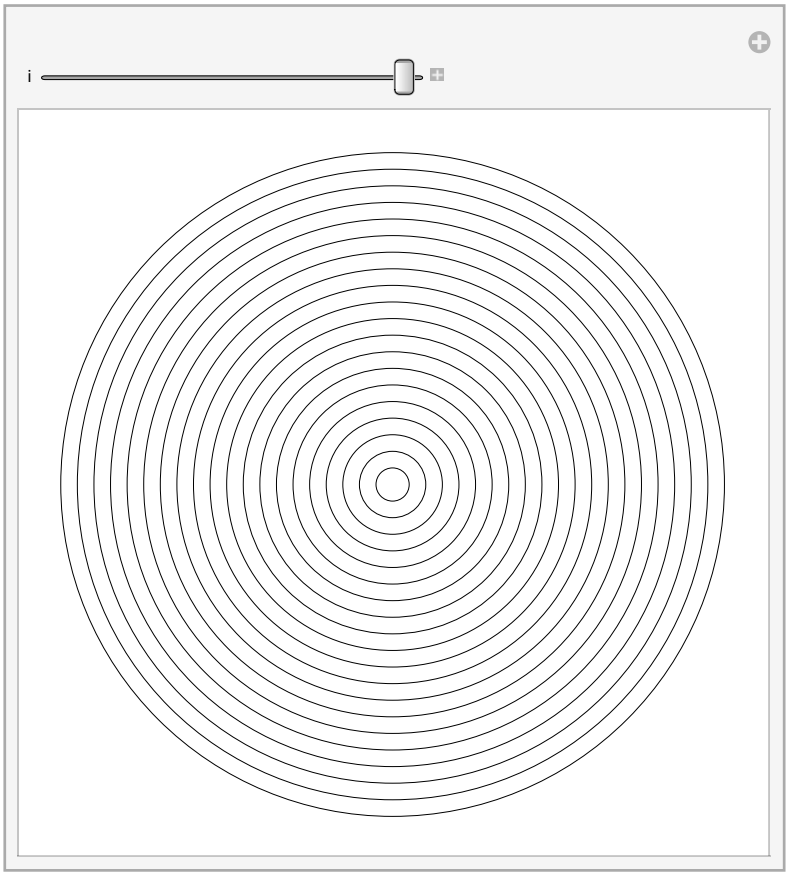
Out[ ]=



14.5 Make a Manipulate with between 1 and 20 concentric circles.

In[ ]:=
```
Manipulate[Graphics[Table[Circle[{0, 0}, r], {r, 1, i}]], {i, 1, 20}]
```

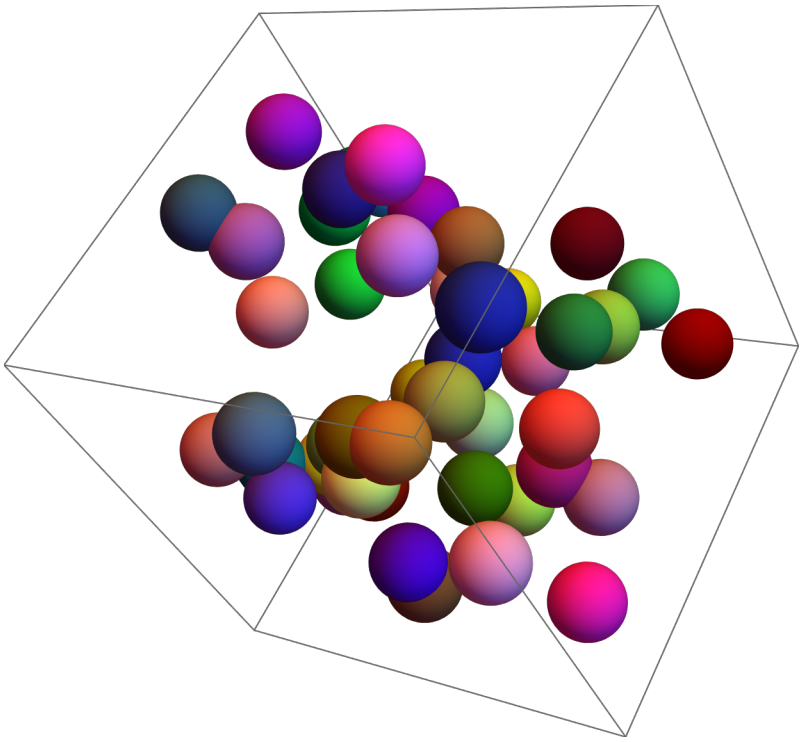
Out[ ]=



14.6 Place 50 spheres with random colors at random integer coordinates up to 10.

In[ ]:=
```
Graphics3D[
 Table[Style[Sphere[{RandomInteger[10], RandomInteger[10], RandomInteger[10]}],
   RandomColor[]], 50]]
```

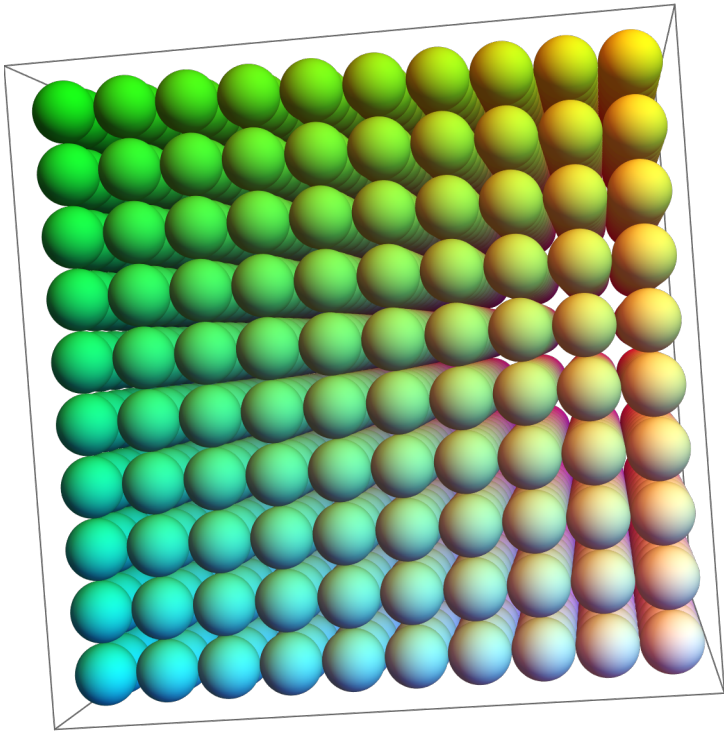
Out[ ]=



14.7 Make a 10x10x10 array of spheres with RGB components ranging from 0 to 10. The spheres should be centered at integer coordinates, and should touch each other.

In[ ]:=
```
Graphics3D[Table[Style[Sphere[{x, y, z}, 1 / 2], RGBColor[{x / 10, y / 10, z / 10}]],
  {x, 10}, {y, 10}, {z, 10}]]
```

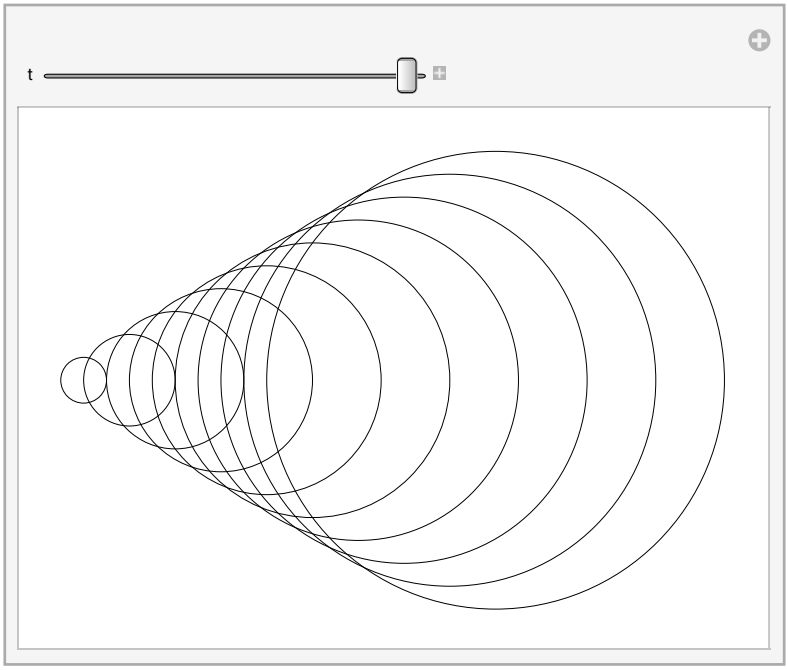
Out[ ]=



14.8 Make a Manipulate with t varying between -2 and +2 that contains circles of radius x centered at {t*x, 0} with x going from 1 to 10.

In[ ]:=
```
Manipulate[Graphics[Table[Circle[{t * x, 0}, x], {x, 1, 10}]], {t, -2, 2}]
```

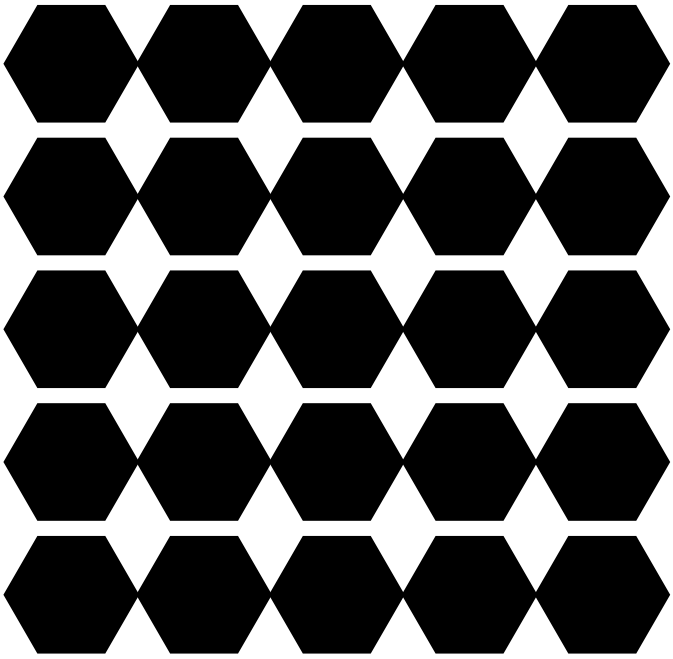
Out[ ]=



14.9 Make a 5x5 array of regular hexagons with size 1/2, centered at integer points.

In[ ]:=
```
Graphics[Table[RegularPolygon[{x, y}, 1 / 2, 6], {x, 5}, {y, 5}]]
```

Out[ ]=

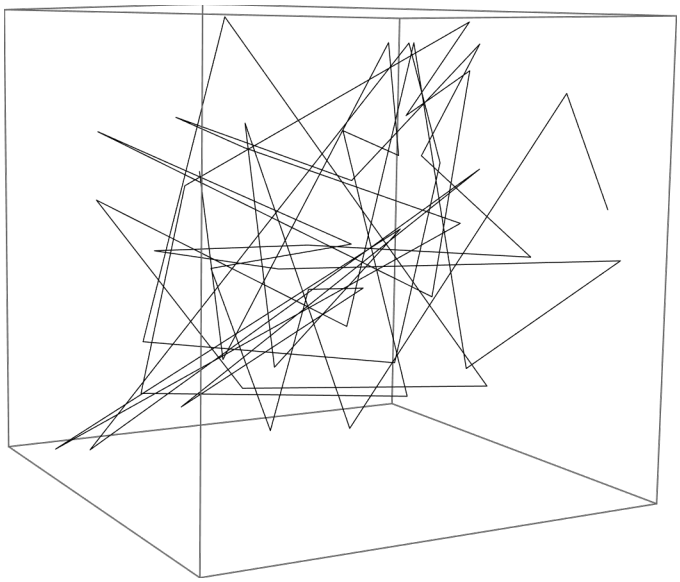


14.10 Make a line in 3D that goes through 50 random points with integer

coordinates randomly chosen up to 50.

In[ ]:=
```
Graphics3D[Line[Table[RandomInteger[50], 50, 3]]]
```

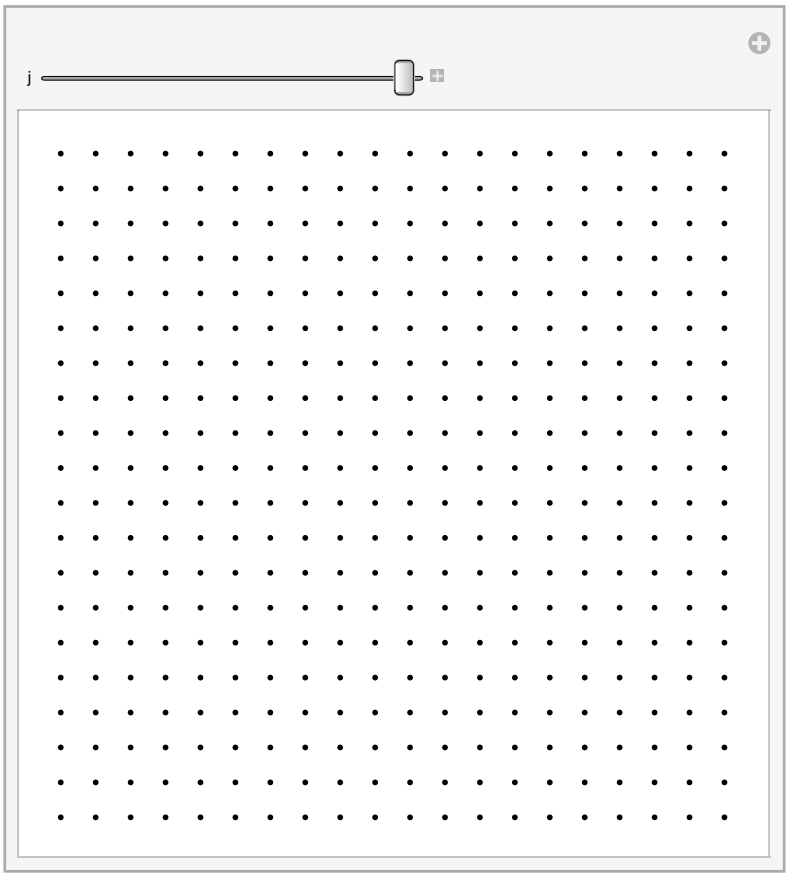
Out[ ]=



+14.1 Make a Manipulate to create an n x n regular grid of points at integer positions, with n going from 5 to 20.

In[ ]:=
```
Manipulate[Graphics[Table[Point[{x, y}], {x, j}, {y, j}]], {j, 5, 20}]
```

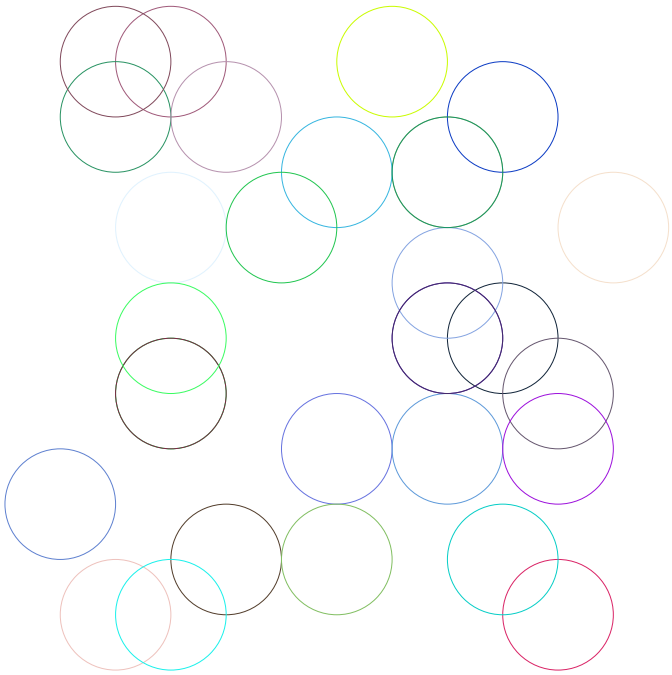
Out[ ]=



+14.2 Place 30 radius-1 circles with random colors at random integer coordinates up to 10.

In[ ]:=
```
Graphics[Table[
  Style[Circle[{RandomInteger[10], RandomInteger[10]}, 1], RandomColor[]], 30]]
```

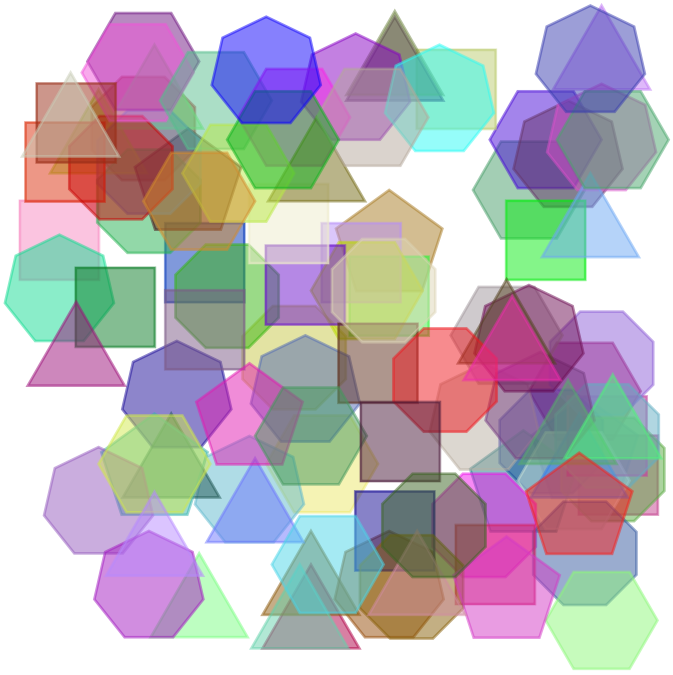
Out[ ]=



+14.3 Display 100 polygons with side length 10, opacity 5, and random choices of colors, sides between 3 and 8, and integer coordinates up to 100.

In[ ]:=
```
Graphics[
 Table[Style[RegularPolygon[{RandomInteger[100], RandomInteger[100]}, 10,
    RandomInteger[5] + 3], RandomColor[], Opacity[.5]], 100]]
```

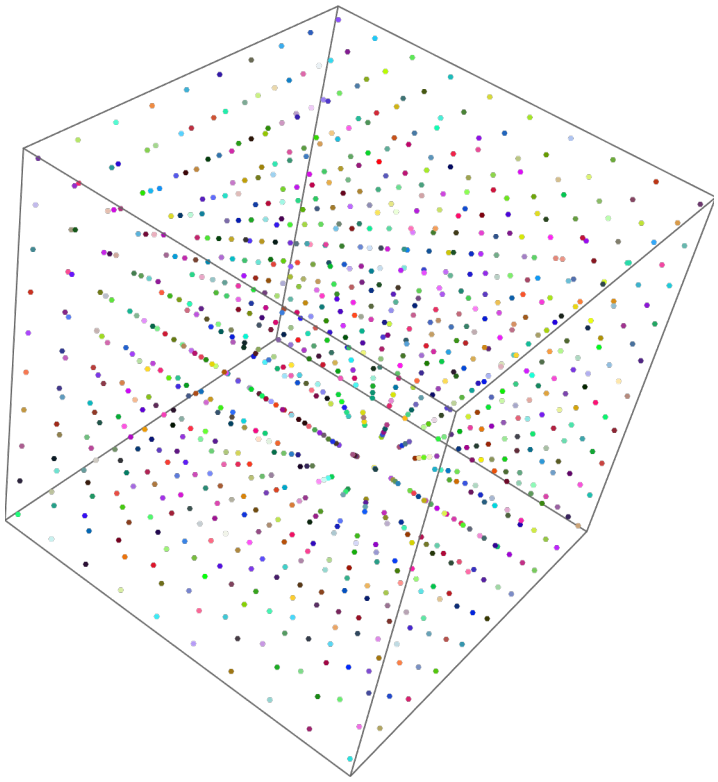
Out[ ]=



+14.4 Make a 10x10x10 array of points in 3D, with each point having a random color.

In[ ]:=
```
Graphics3D[
 Table[Style[Point[{x, y, z}], RandomColor[]], {x, 1, 10}, {y, 1, 10}, {z, 1, 10}]]
```

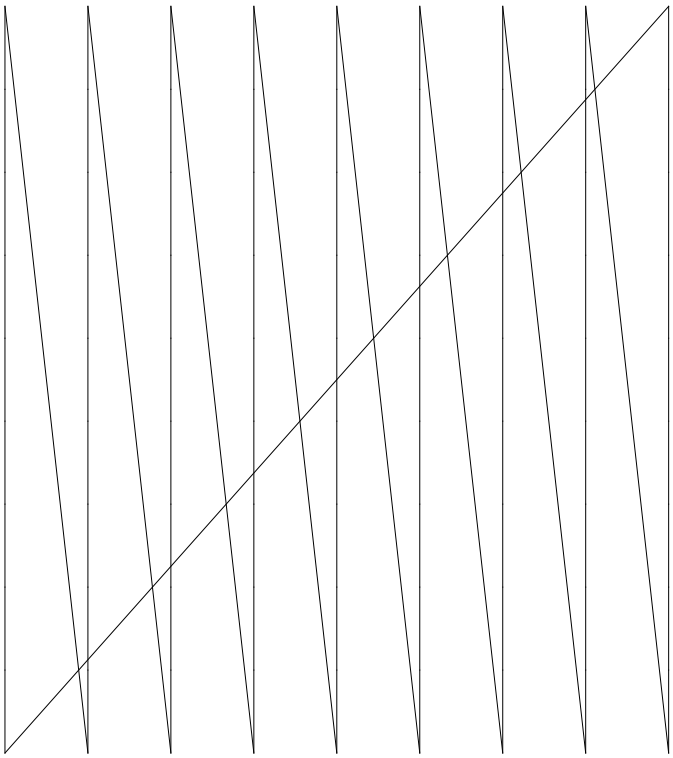
Out[ ]=



+14.5 Take the first two digits of numbers from 10 to 100 and draw a line that uses them as coordinates.

In[•]:=
```
Graphics[
 Table[Line[{IntegerDigits[n], Take[IntegerDigits[n + 1], 2]}], {n, 10, 99}]]
```

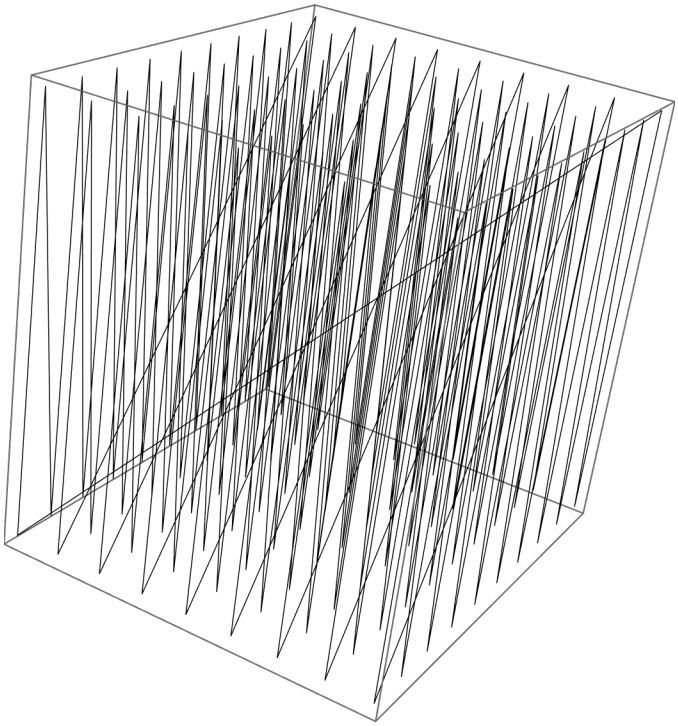
Out[•]=



+14.6 Take the first 3 digits of numbers from 100 to 1000 draw a line in 3D that uses them as coordinates.

```
In[ ]:= Graphics3D[Table[Line[{IntegerDigits[n],
          Take[IntegerDigits[n + 1], 3], Take[IntegerDigits[n + 2], 3]}], {n, 100, 998}]]
```

Out[ ]=

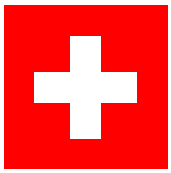


Exercises for Section 16 | Real-World Data

16.1 Find the flag of Switzerland

```
In[ ]:= Switzerland COUNTRY [ flag ]
```

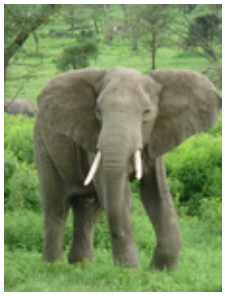
Out[ ]=



16.2 Get an image of an elephant.

In[ ]:=
```
African bush elephant SPECIES SPECIFICATION [ image ]
```

Out[ ]=



16.3 Use the “Mass” property to generate a list of the masses of the planets.

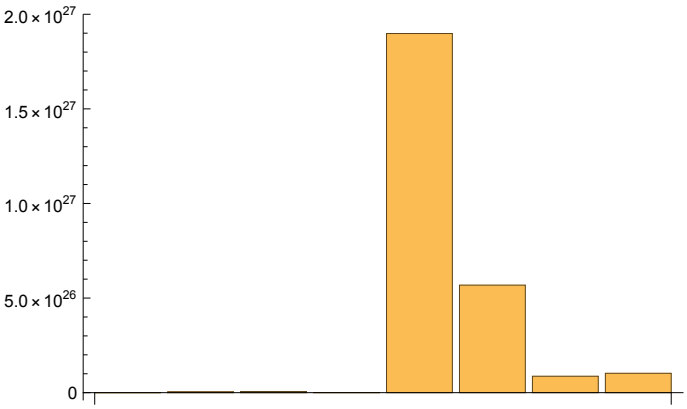
In[ ]:=
```
Mass of the planets »
planets PLANETS [ mass ]
```

Out[ ]=
$$\{3.301\times10^{23}\ \text{kg},\ 4.867\times10^{24}\ \text{kg},\ 5.97\times10^{24}\ \text{kg},\ 6.417\times10^{23}\ \text{kg},\ 1.898\times10^{27}\ \text{kg},\ 5.683\times10^{26}\ \text{kg},\ 8.681\times10^{25}\ \text{kg},\ 1.0243\times10^{26}\ \text{kg}\}$$

16.4 Make a bar chart of the masses of the planets.

In[ ]:=
```
BarChart [ planets PLANETS [ mass ] ··· ✓ ]
```

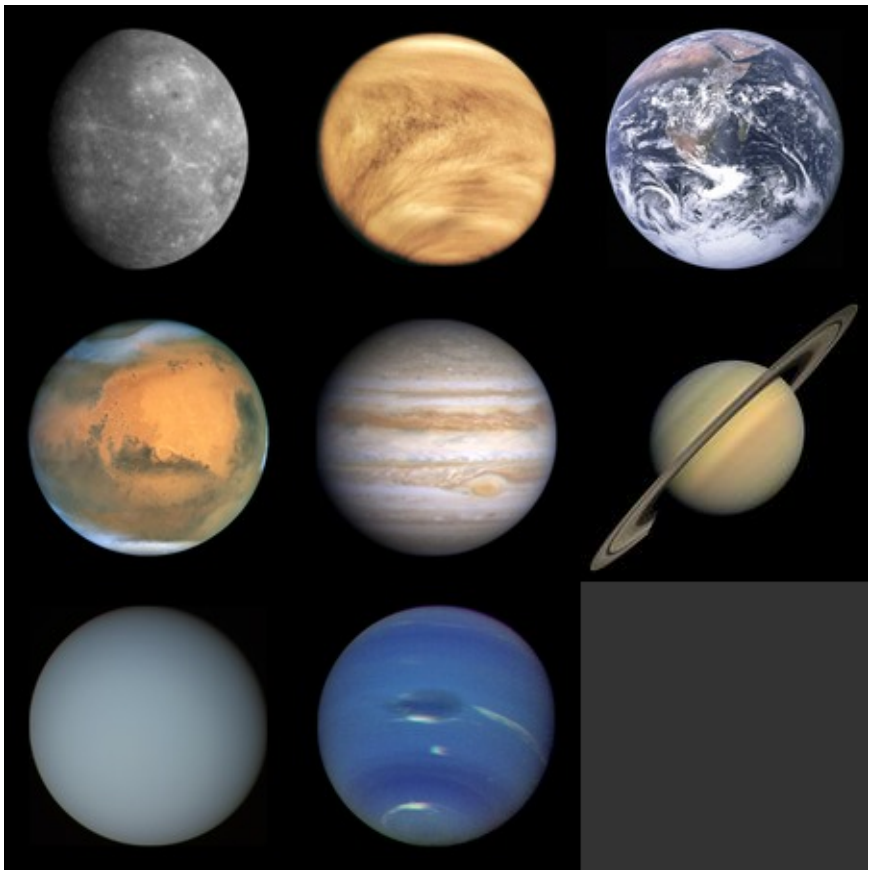
Out[ ]=



16.5 Make an image collage of images of the planets.

In[ ]:= `ImageCollage[ planets PLANETS [ image ] ]`

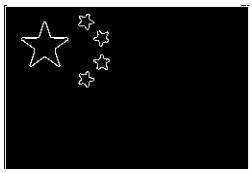
Out[ ]=



16.6 Edge detect the flag of China.

In[ ]:= `EdgeDetect[ China COUNTRY [ flag ] ... ✓ ]`

Out[ ]=



16.7 Find the height of the Empire State Building.

In[ ]:= `Empire State Building BUILDING [ total height ]`

Out[ ]= `381. m`

16.8 Compute the height of the Empire State Building divided by the height of the Great Pyramid.

```
In[ ]:= Empire State Building BUILDING [ total height ] /
          Egyptian Pyramids of Giza BUILDINGS [ total height ]
```

```
Out[ ]= {2.74101, 21.7714, 12.5743, 12.7, 12.8716, 2.80147, 19.05, 5.77273}
```

```
In[ ]:= CurrencyConvert[ €100.00 , "dollars"]
```

```
Out[ ]= $113.10
```

Exercises for Section 18 | Geocomputation

18.1 Find the distance from New York to London.

```
In[ ]:= GeoDistance[ New York City CITY , London CITY ]
```

```
Out[ ]= 5558.2 km
```

18.2 Divide the distance from New York to London by the distance from New york to San Francisco.

```
In[ ]:= GeoDistance[ New York City CITY , London CITY ] /
          GeoDistance[ New York City CITY , San Francisco CITY ]
```

```
Out[ ]= 1.3511
```

18.3 Find the distance from Sydney to Moscow in kilometers.

```
In[ ]:= UnitConvert[GeoDistance[ Sydney CITY , Moscow CITY ]]
```

```
Out[ ]= 1.4462 × 10^7 m
```

18.4 Generate a map of the United States.

In[ ]:=
```
GeoGraphics[United States COUNTRY]
```

Out[ ]=

18.5 Plot on a map Brazil, Russia, India and China.

In[ ]:=
```
GeoGraphics[
 {Brazil COUNTRY, Russia COUNTRY, India COUNTRY, China COUNTRY} ...]
```

Out[ ]=

18.6 Plot on a map the path from New York City to Beijing.

```
In[ ]:= GeoGraphics[GeoPath[{New York City CITY, Beijing CITY}]]
```

Out[ ]=

18.7 Plot a disk centered on the Great Pyramid, with radius 10 miles.

```
In[ ]:= GeoGraphics[GeoDisk[Egyptian Pyramids of Giza BUILDINGS, 10 mi]]
```

GeoDisk :

{GeoPosition [{29.9792, 31.1338 }], GeoPosition [{29.9761, 31.1328 }], GeoPosition [{29.9788, 31.1363 }], GeoPosition [{29.9783, 31.1361 }], GeoPosition [{29.9779, 31.1363 }], GeoPosition [{29.9759, 31.1307 }], GeoPosition [{29.9753, 31.1375 }], GeoPosition [{29.9725, 31.1282 }]} is not a valid center location.

Out[ ]=

In[ ]:=
```
GeoGraphics[GeoDisk[
  Entity["Building", "GreatPyramidOfGiza::jbm66"], Quantity[10, "Miles"]]]
```

Out[ ]=


Exercises for Section 19 | Dates and Times

19.1 Compute how many days have elapsed since January 1, 1900.

In[ ]:=
```
Now - DateObject[{1900, 1, 1}]
```

Out[ ]=
```
44 552. days
```

19.2 Compute what day of the week January 1, 2000 was.

In[ ]:=
```
DayName[DateObject[{2000, 1, 1}]]
```

Out[ ]=
```
Saturday
```

19.3 Find the date a hundred thousand days ago.

In[ ]:=
```
Today - 100 000 days
```

Out[ ]=
```
Sun 10 Mar 1748
```

19.4 Find the local time in Delhi.

```
In[ ]:= LocalTime[Delhi CITY]
```

```
Out[ ]= Sat 25 Dec 2021 04:50:51 GMT+5.5
```

19.5 Find the length of daylight today by subtracting today's sunrise from today's sunset.

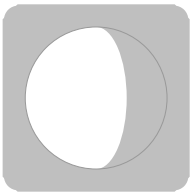
```
In[ ]:= Sunset[Here, Today] - Sunrise[Here, Today]
```

```
Out[ ]= 867 min
```

19.6 Generate an icon for the current phase of the moon.

```
In[ ]:= MoonPhase[Now, "Icon"]
```

Out[ ]=



19.7 Make a list of the numerical phase of the moon for each of the next 10 days.

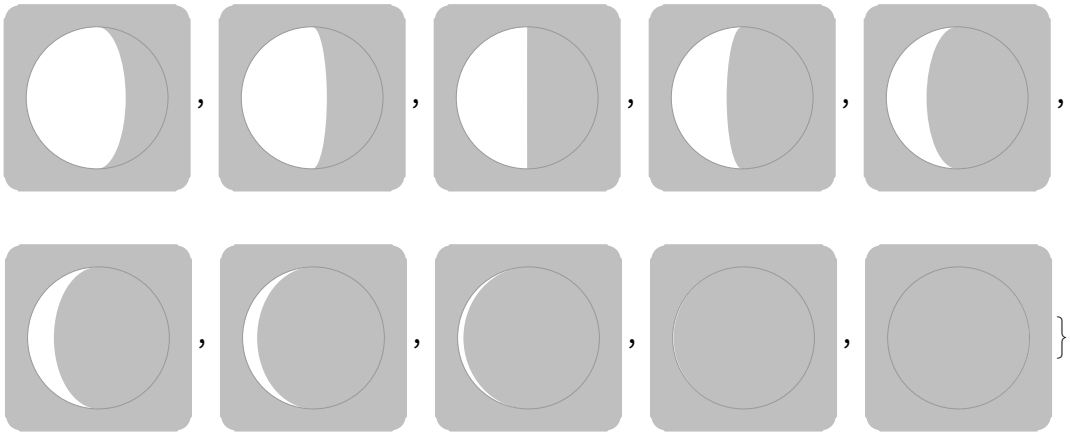
```
In[ ]:= Table[MoonPhase[Today + n days], {n, 1, 10}]
```

```
Out[ ]= {0.7016, 0.6038, 0.4987, 0.3903, 0.2839,
  0.1857, 0.1025, 0.04111, 0.006980, 0.002951}
```

19.8 Generate a list of icons for the moon phases from today until 10 days from now.

```
In[ ]:= Table[MoonPhase[Today + n days, "Icon"], {n, 1, 10}]
```

Out[ ]=



19.9 Compute the time today between sunrise in New York City and in London.

```
In[ ]:= Sunrise[Entity["City", {"NewYork", "NewYork", "UnitedStates"}], Today] -
         Sunrise[Entity["City", {"London", "GreaterLondon", "UnitedKingdom"}], Today]
```

Out[ ]= 253 min

19.10 Find the air temperature at the Eiffel Tower at noon yesterday.

```
In[ ]:= AirTemperatureData[Eiffel Tower BUILDING,
          DateObject[Yesterday, TimeObject[{12, 0, 0}]]]
```

Out[ ]= 7. °C

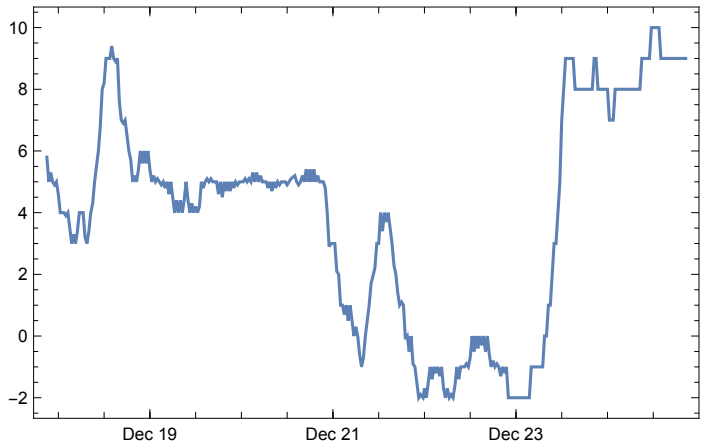
```
In[ ]:= AirTemperatureData[Dublin CITY, Yesterday]
```

Out[ ]= (5. to 13.) °C

19.11 Plot the temperature at the Eiffel Tower over the past week.

```
In[ ]:= DateListPlot[AirTemperatureData[
          Entity["Building", "EiffelTower::5h9w8"], {Now - Quantity[1, "Weeks"], Now}]]
```

Out[ ]=



19.12 Find the difference in air temperatures between Los Angeles and New York City now.

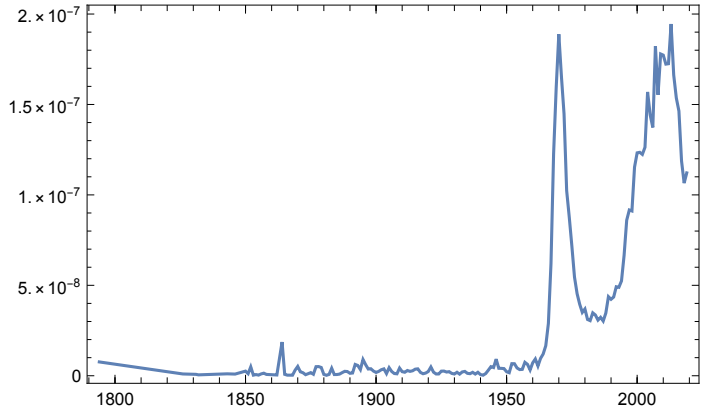
```
In[ ]:= AirTemperatureData[Los Angeles CITY, Now] -
         AirTemperatureData[New York City CITY, Now]
```

Out[ ]= 12.2 °C

19.13 Plot the historical frequency of the word “groovy”

```
In[ ]:= DateListPlot[WordFrequencyData["groovy", "TimeSeries"]]
```

Out[ ]=



+19.1 Compute how many weeks have elapsed since January 1, 1900.

```
In[ ]:= 6365. wk
```

Out[ ]= 6365. wk

```
In[ ]:= UnitConvert[Now - DateObject[{1900, 1, 1}], "Weeks"]
Out[ ]= 6364.57 wk
```

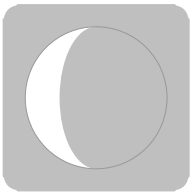
+19.2 Compute the time between 3 pm today and sunset today.

```
In[ ]:= DateObject[Today, TimeObject[{15}]] - Sunset[Today]
Out[ ]= -5 h
```

+19.3 Generate an icon of the phase of the moon on August 29, 1959.

```
In[ ]:= MoonPhase[DateObject[{1959, 8, 29}], "Icon"]
```

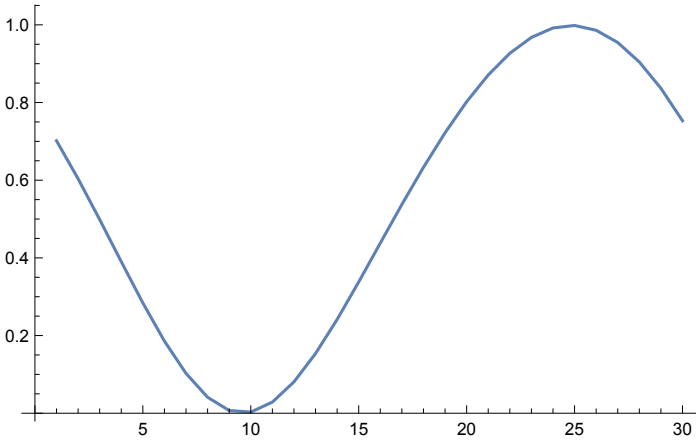
Out[ ]=



+19.4 Make a line plot of the numerical phase of the moon for each of the next 30 days.

```
In[ ]:= ListLinePlot[Table[MoonPhase[Today + n days], {n, 1, 30}]]
```

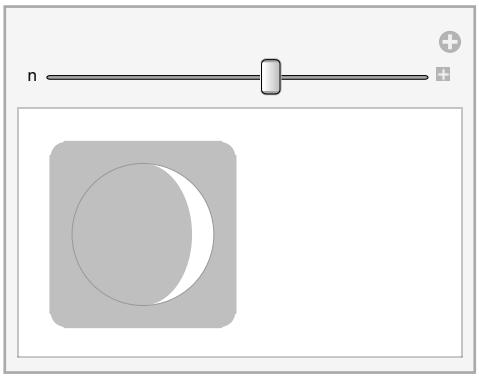
Out[ ]=



+19.5 Make a Manipulate of the icon for the moon phase over the net 15 days.

In[ ]:=
```
Manipulate[MoonPhase[Today + n days, "Icon"], {n, 1, 15}]
```

Out[ ]=



+19.6 Show in a column the times of sunrise for 10 days, starting today.

In[ ]:=
```
Column[Table[Sunrise[Today + n days], {n, 0, 10}]]
```

Out[ ]=

Minute: Fri 24 Dec 2021 05:44 GMT–3

Minute: Sat 25 Dec 2021 05:45 GMT–3

Minute: Sun 26 Dec 2021 05:45 GMT–3

Minute: Mon 27 Dec 2021 05:46 GMT–3

Minute: Tue 28 Dec 2021 05:47 GMT–3

Minute: Wed 29 Dec 2021 05:47 GMT–3

Minute: Thu 30 Dec 2021 05:48 GMT–3

Minute: Fri 31 Dec 2021 05:49 GMT–3

Minute: Sat 1 Jan 2022 05:50 GMT–3

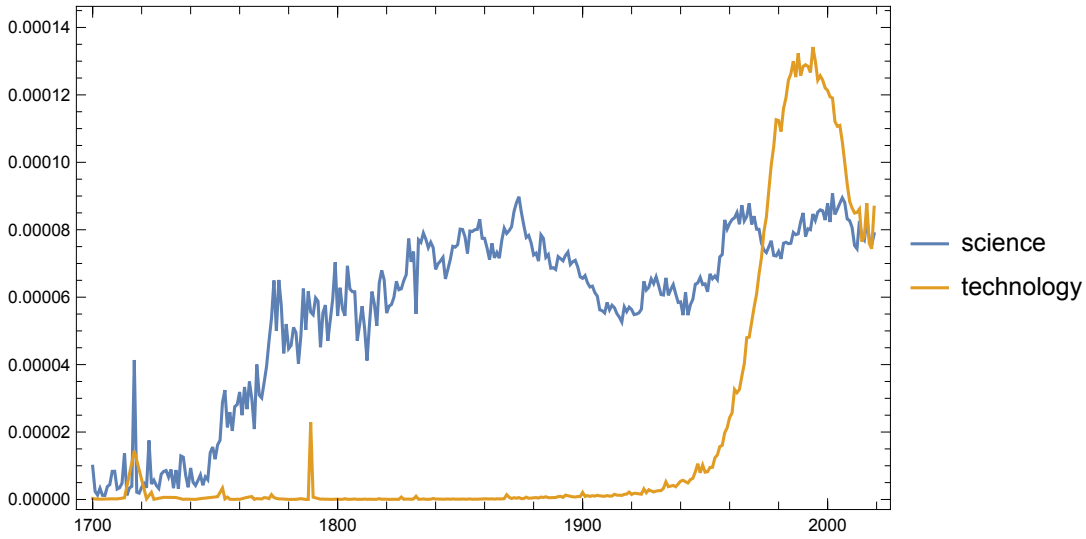
Minute: Sun 2 Jan 2022 05:50 GMT–3

Minute: Mon 3 Jan 2022 05:51 GMT–3

+19.7 Plot together the historical frequencies of the words “science” and “technology”

```
In[ ]:= DateListPlot[WordFrequencyData[{"science", "technology"}, "TimeSeries"]]
```

Out[ ]=

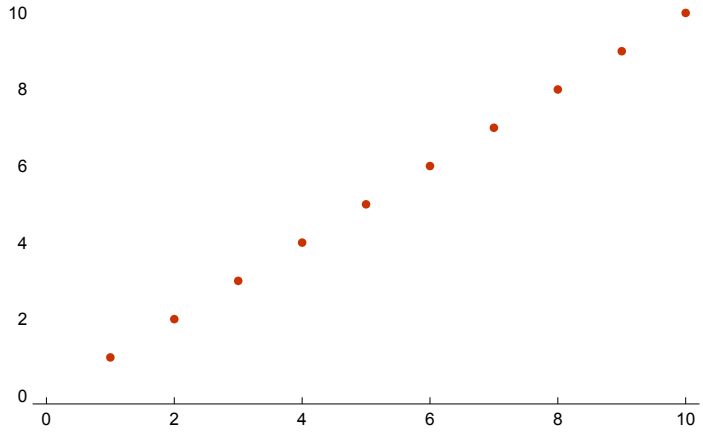


Exercises for Section 20 | Options

20.1 Create a list plot of Range[10] themed for the web.

```
In[ ]:= ListPlot[Range[10], PlotTheme → "Web"]
```

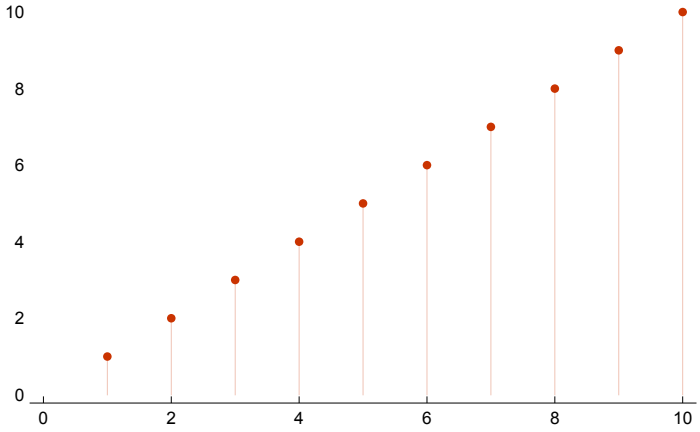
Out[ ]=



20.2 Create a list plot of Range[10] with filling to the axis.

In[ ]:=
```
ListPlot[Range[10], PlotTheme → "Web", Filling → Axis]
```

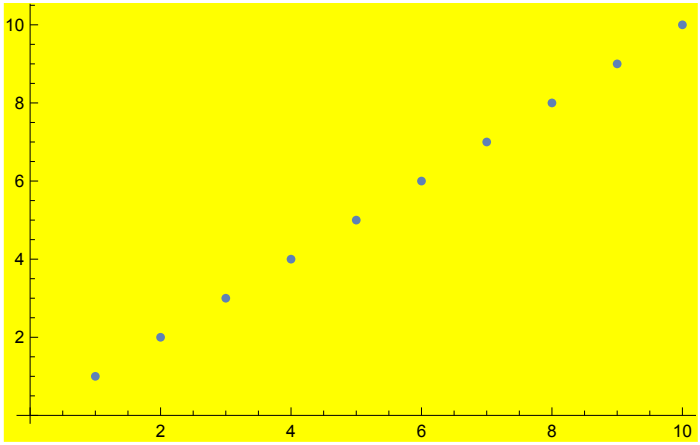
Out[ ]=



20.3 Create a list plot of Range[10] with yellow background.

In[ ]:=
```
ListPlot[Range[10], Background → Yellow]
```

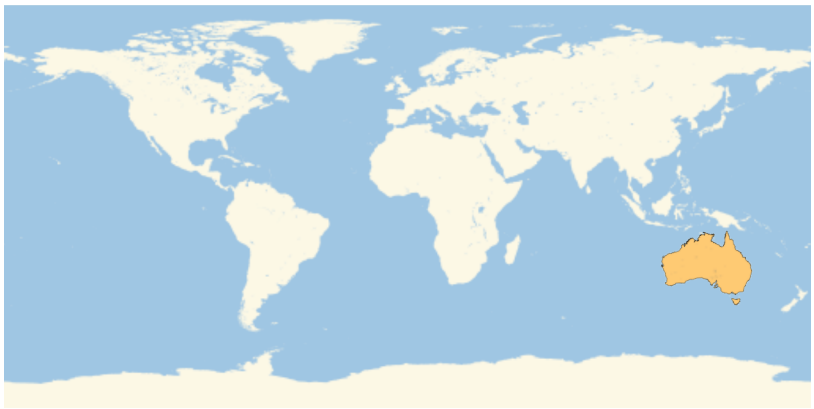
Out[ ]=



20.4 Create a map of the world with Australia highlighted.

In[ ]:=
```
GeoListPlot[Australia COUNTRY ··· ✓, GeoRange → All]
```

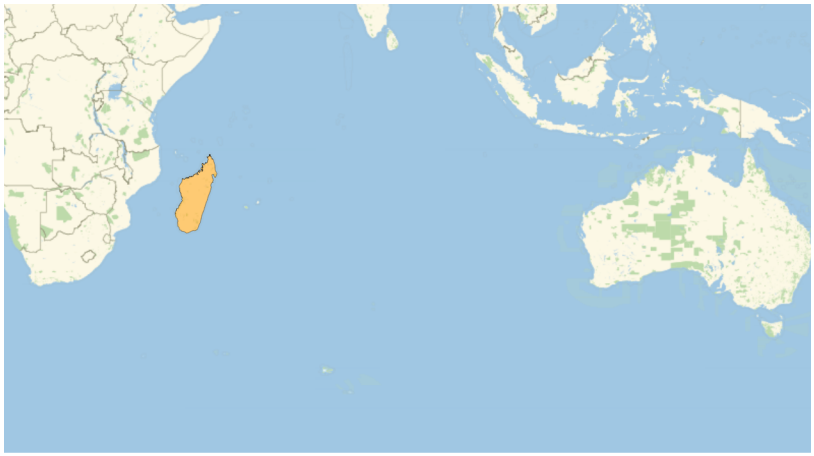
Out[ ]=



20.5 Create a map of the Indian Ocean with Madagascar highlighted.

In[ ]:=
```
GeoListPlot[Madagascar COUNTRY, GeoRange → Indian Ocean OCEAN]
```

Out[ ]=



20.6 Use GeoGraphics to create a map of South America showing topography (relief map).

In[ ]:=
```
GeoGraphics[South America COUNTRIES, GeoBackground → "ReliefMap"]
```

Out[ ]=

20.7 Make a map of Europe with France, Finland and Greece highlighted and labeled

In[ ]:=
```
GeoListPlot[{France COUNTRY, Finland COUNTRY, Greece COUNTRY}, GeoLabels → True]
```

Out[ ]=

20.8 Make a 12x12 multiplication table as a grid with white type on a black background.

In[ ]:=
```
Grid[Table[Style[x * y, White], {x, 0, 12}, {y, 0, 12}],
 Frame → All, Background → Black]
```

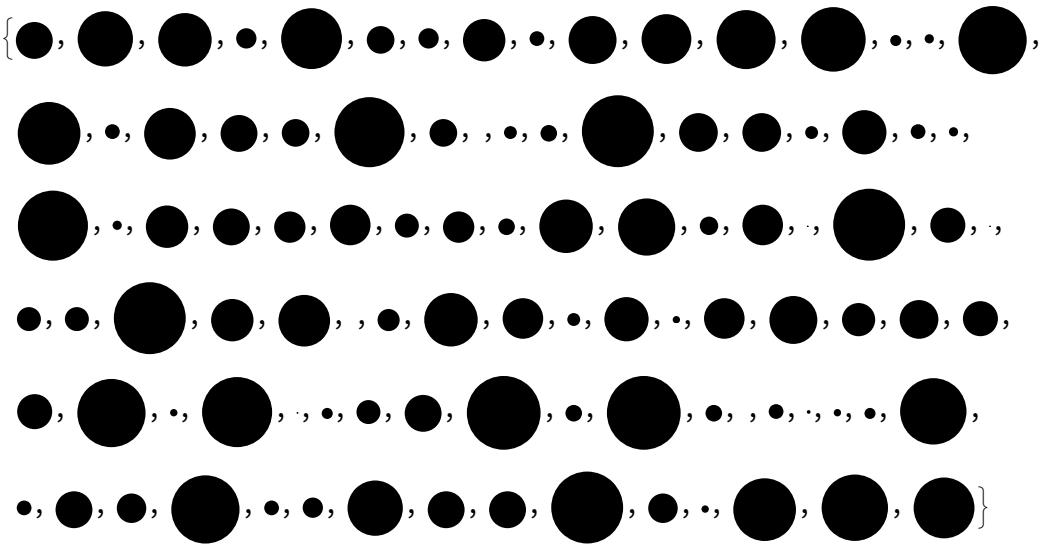
Out[ ]=

0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12
0	2	4	6	8	10	12	14	16	18	20	22	24
0	3	6	9	12	15	18	21	24	27	30	33	36
0	4	8	12	16	20	24	28	32	36	40	44	48
0	5	10	15	20	25	30	35	40	45	50	55	60
0	6	12	18	24	30	36	42	48	54	60	66	72
0	7	14	21	28	35	42	49	56	63	70	77	84
0	8	16	24	32	40	48	56	64	72	80	88	96
0	9	18	27	36	45	54	63	72	81	90	99	108
0	10	20	30	40	50	60	70	80	90	100	110	120
0	11	22	33	44	55	66	77	88	99	110	121	132
0	12	24	36	48	60	72	84	96	108	120	132	144

20.9 Make a list of 100 disks with random integer image sizes up to 40.

```
In[ ]:= Table[Graphics[Disk[], ImageSize → RandomInteger[40]], 100]
```

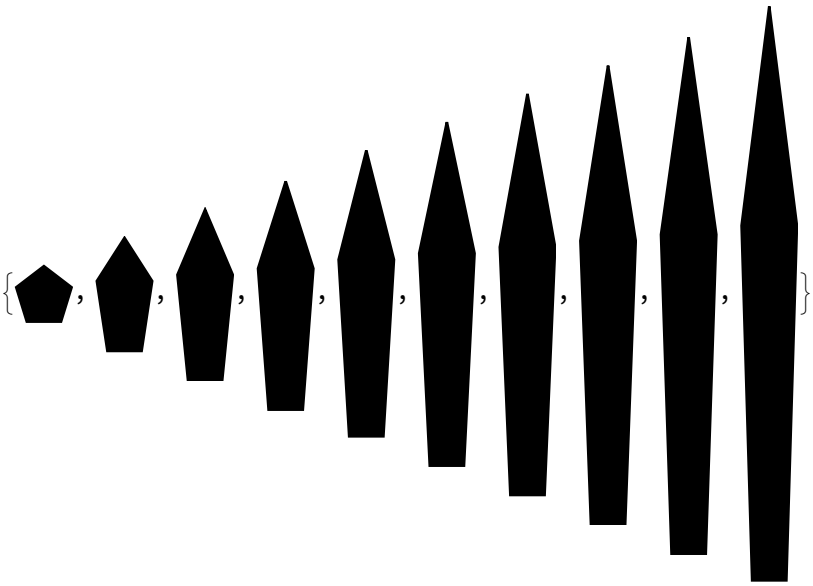
Out[ ]=



20.10 Make a list of pictures of regular pentagons with image size 30 and aspect ratios from 1 to 10.

```
In[ ]:= Table[Graphics[RegularPolygon[5], ImageSize → 30, AspectRatio → n], {n, 1, 10}]
```

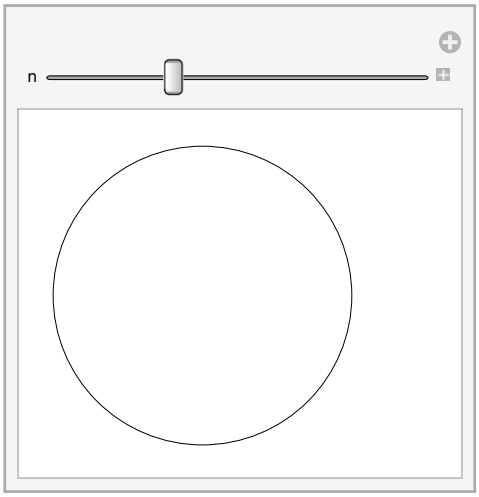
Out[ ]=



20.11 Make a Manipulate that varies the size of a circle between 5 and 500.

In[ ]:=
```
Manipulate[Graphics[Circle[], ImageSize → n], {n, 5, 500}]
```

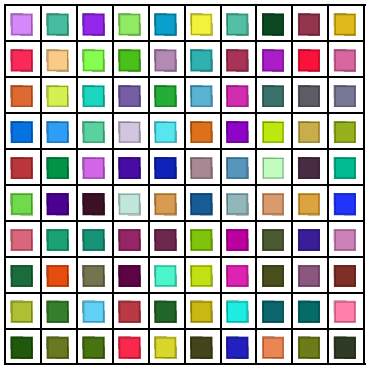
Out[ ]=



20.12 Create a framed 10x10 grid of random colors.

In[ ]:=
```
Grid[Table[RandomColor[], 10, 10], Frame → All]
```

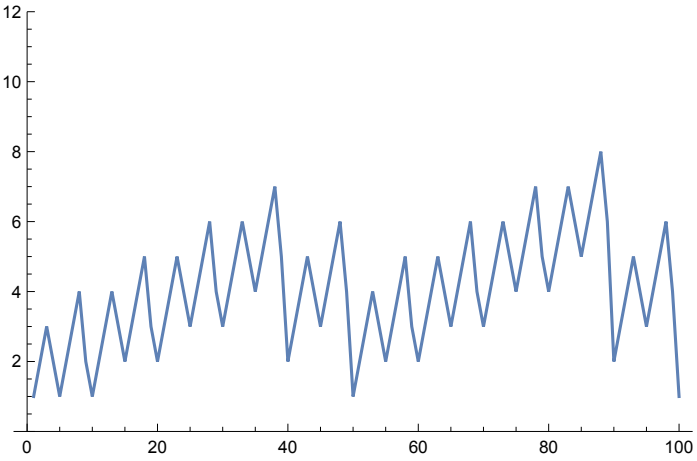
Out[ ]=



20.13 Make a line plot of the lengths of Roman numerals up to 100, with a plot range that would be sufficient for all numerals up to 1000.

In[ ]:=
```
ListLinePlot[Table[StringLength[RomanNumeral[n]], {n, 100}],
 PlotRange → Max[Table[StringLength[RomanNumeral[n]], {n, 1000}]]]
```

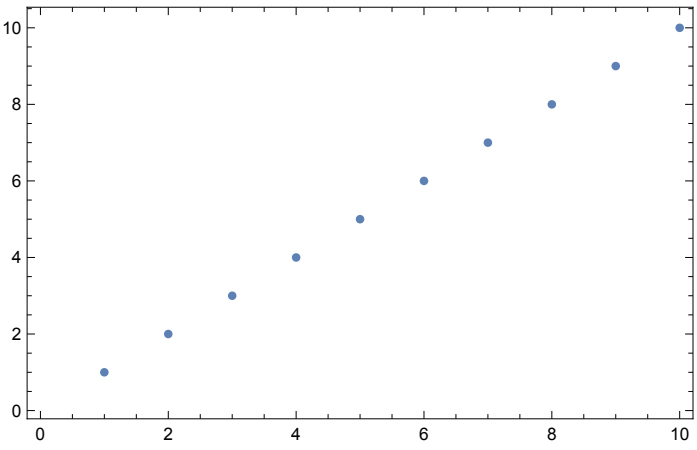
Out[ ]=



+20.1 Create a list plot of Range[10] with a frame included.

In[ ]:=
```
ListPlot[Range[10], Frame → True]
```

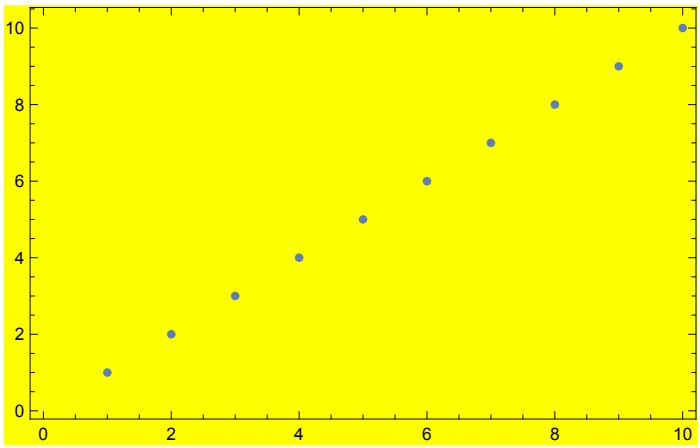
Out[ ]=



+20.2 Create a list plot of Range[10] with a yellow background and a frame.

In[ ]:=
```
ListPlot[Range[10], Background → Yellow, Frame → True]
```

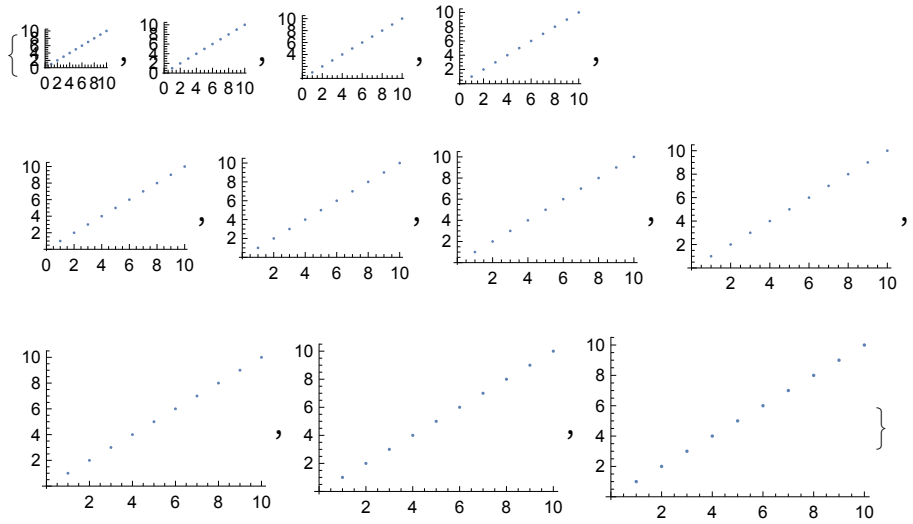
Out[ ]=



+20.3 Make a list of list plots of Range[10], with sizes ranging from 50 to 150 in steps of 10.

In[ ]:=
```
Table[ListPlot[Range[10], ImageSize → n], {n, 50, 150, 10}]
```

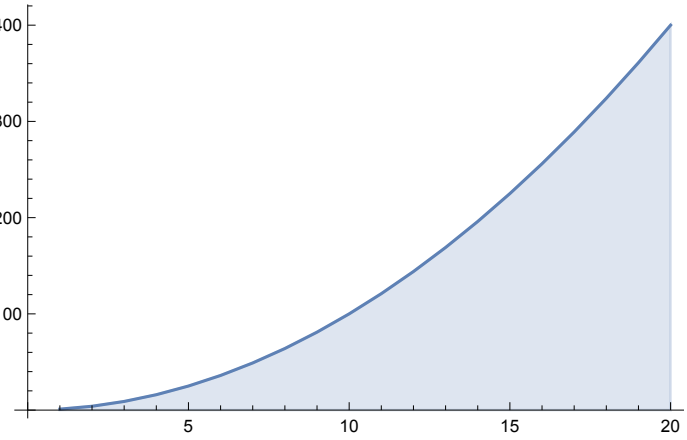
Out[ ]=



+20.4 Make a line plot of the first 20 squares, filling to the axis.

In[ ]:=
```
ListLinePlot[Table[n^2, {n, 1, 20}], Filling → Axis]
```

Out[ ]=

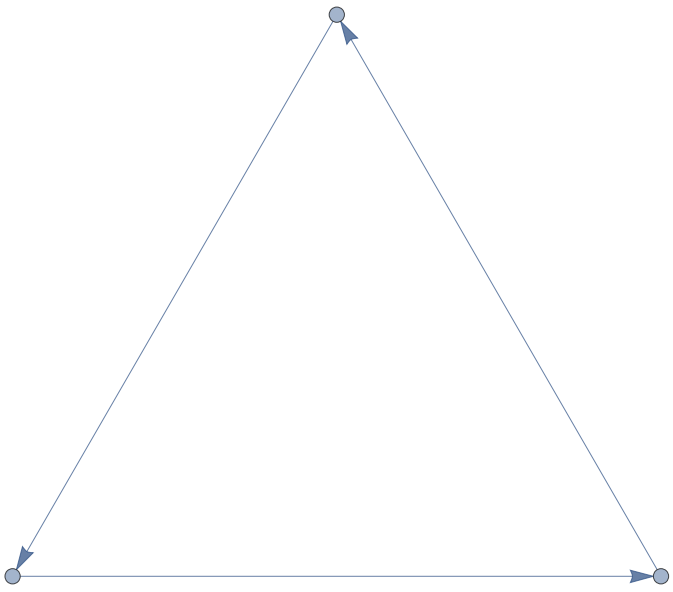


Exercises for Section 21 | Graphs and Networks

21.1 Make a graph consisting of a loop of 3 nodes.

In[ ]:=
```
Graph[{1 → 2, 2 → 3, 3 → 1}]
```

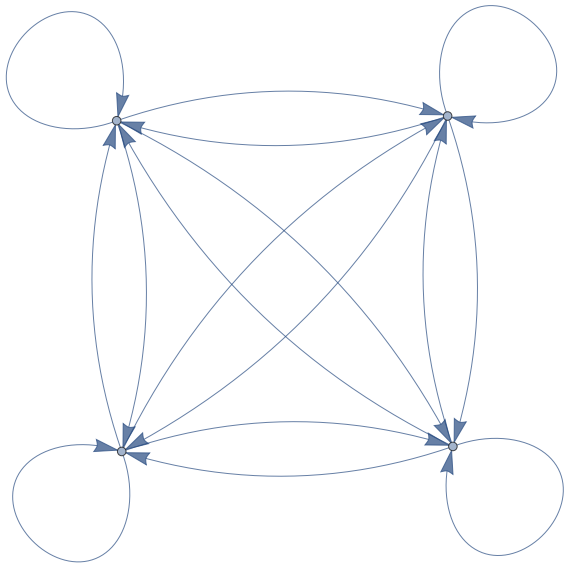
Out[ ]=



21.2 Make a graph with 4 nodes in which every node is connected.

In[ ]:=
```
Graph[Flatten[Table[j → i, {i, 1, 4}, {j, 1, 4}]]]
```

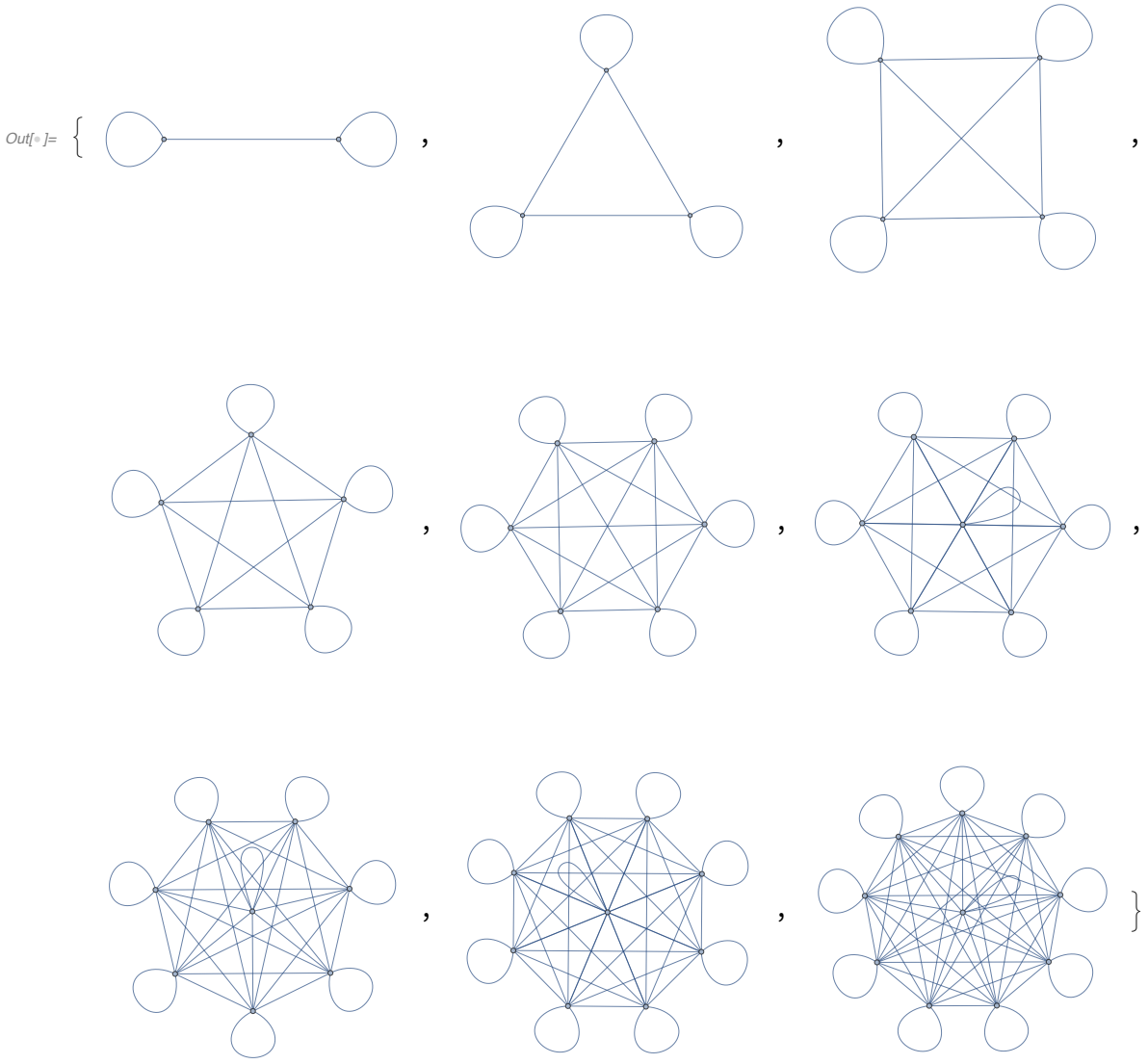
Out[ ]=



21.3 Make a table of undirected graphs with between 2 and 10 nodes in which every node is connected.

```
In[•]:= Table[UndirectedGraph[Flatten[Table[i → j, {i, n}, {j, n}]]], {n, 2, 10}]
```

Out[•]=



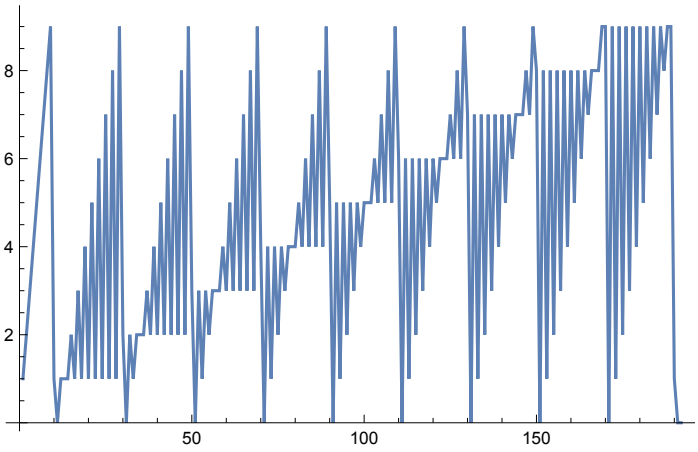
21.4 Use Table and Flatten to get {1, 2, 1, 2, 1, 2}.

```
In[•]:= Flatten[Table[Table[i, {i, 1, 2}], 3]]
Out[•]= {1, 2, 1, 2, 1, 2}
```

21.5 Make a line plot of the result of concatenating all digits of all integers from 1 to 100 (i.e. ..., 8, 9, 1, 0, 1, 1, 1, 2, ...).

```
In[ ]:= ListLinePlot[Flatten[IntegerDigits[Range[100]]]]
```

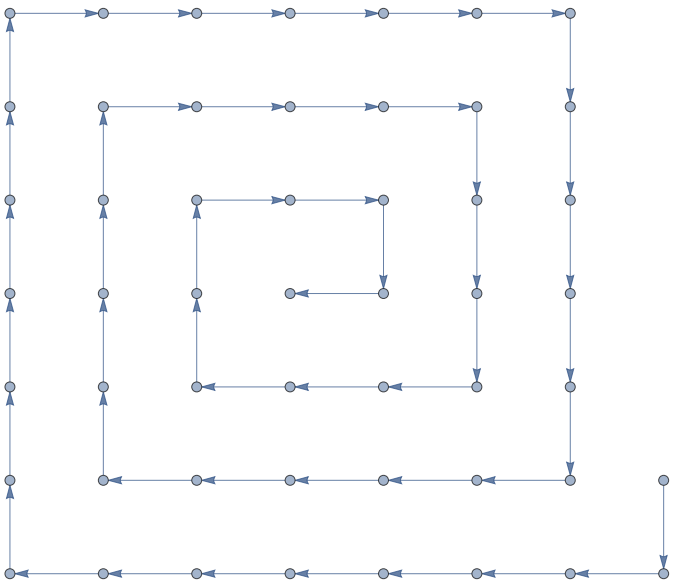
Out[ ]=



21.6 Make a graph with 50 nodes, in which node i connects to node i + 1.

```
In[ ]:= Graph[Table[i → i + 1, {i, 1, 50}]]
```

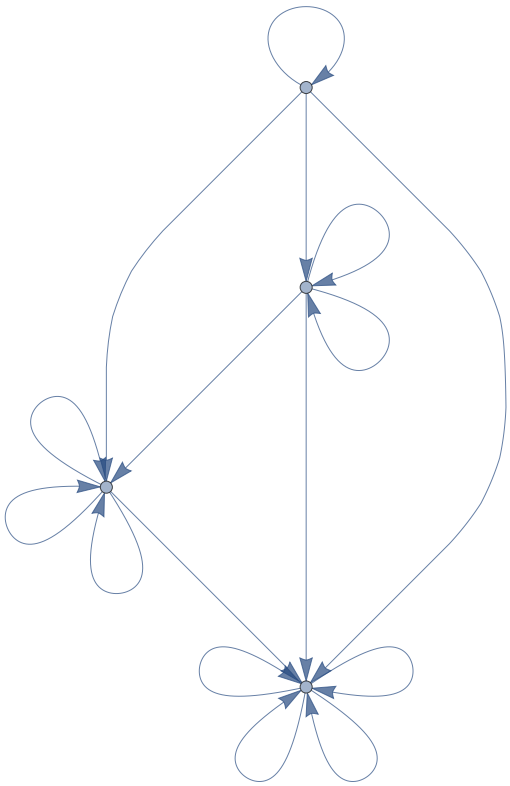
Out[ ]=



21.7 Make a graph with 4 nodes, in which each connection connects i to Max[i, j].

In[ ]:=
```
Graph[Flatten[Table[i → Max[i, j], {i, 1, 4}, {j, 1, 4}]]]
```

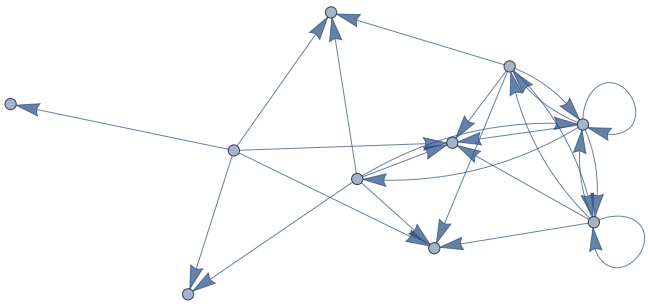
Out[ ]=



21.8 Make a graph in which each connection connects i to j-i, where i and j both range from 1 to 5.

In[ ]:=
```
Graph[Flatten[Table[i → (j - i), {i, 1, 5}, {j, 1, 5}]]]
```

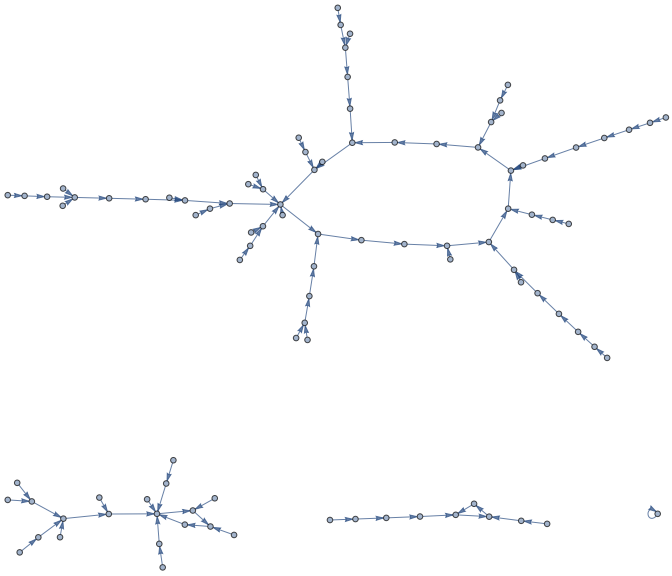
Out[ ]=



21.9 Generate a graph with 100 nodes, each with a connection going to one randomly chosen node.

```
In[ ]:= Graph[Table[i → RandomInteger[{1, 100}], {i, 1, 100}]]
```

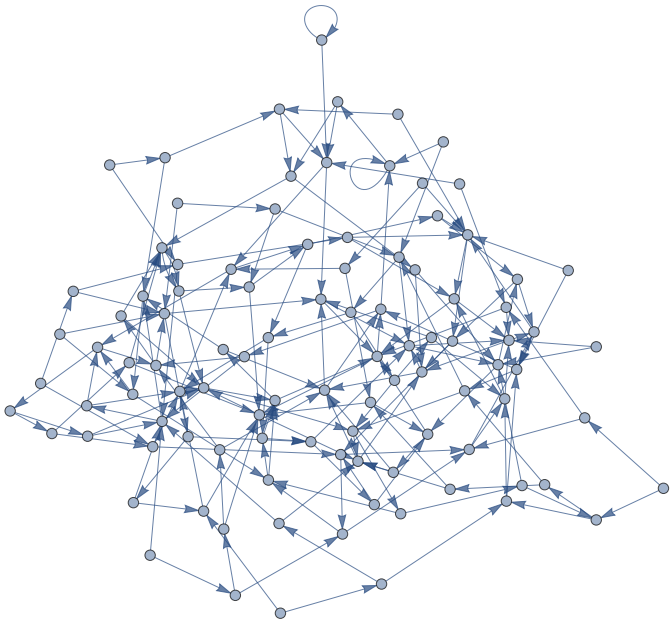
Out[ ]=



21.10 Generate a graph with 100 nodes, each connection to two randomly chosen nodes.

```
In[ ]:= Graph[Flatten[
  Table[{i → RandomInteger[{1, 100}], i → RandomInteger[{1, 100}]}, {i, 1, 100}]]]
```

Out[ ]=



21.11 For the graph {1 -> 2, 2 ->3, 3 -> 4, 4 -> 1, 3 -> 1, 2 -> 2}, make a grid giving the shortest paths between every pair of nodes, with the start node as row and

end node as column.

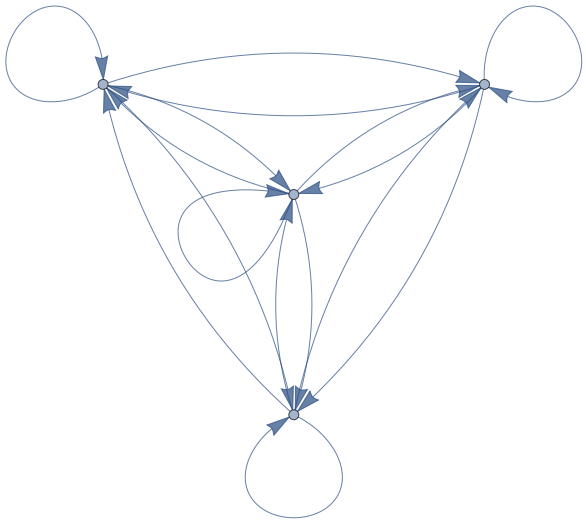
```
In[ ]:= Grid[Table[FindShortestPath[
          Graph[{1 → 2, 2 → 3, 3 → 4, 4 → 1, 3 → 1, 2 → 2}], i, j], {i, 4}, {j, 4}]]
```

```
Out[ ]=
{1}      {1, 2}    {1, 2, 3}  {1, 2, 3, 4}
{2, 3, 1}  {2}       {2, 3}     {2, 3, 4}
{3, 1}   {3, 1, 2}     {3}        {3, 4}
{4, 1}   {4, 1, 2} {4, 1, 2, 3}     {4}
```

+21.1 Make a graph with 4 nodes in which every node is connected, displaying the resulting graph with radial drawing.

```
In[ ]:= Graph[Flatten[Table[j → i, {i, 1, 4}, {j, 1, 4}]], GraphLayout → "RadialDrawing"]
```

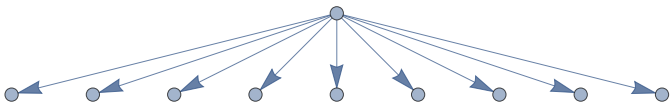
Out[ ]=



+21.2 Generate a graph in which a single node is connected to 10 others.

```
In[ ]:= Graph[Table[1 → i, {i, 2, 10}]]
```

Out[ ]=



+21.3 Use Flatten to generate a table of the numbers 1 to 50 in which even numbers are colored red.

```
In[ ]:= Flatten[Table[{i, Style[i + 1, Red]}, {i, 1, 50, 2}]]
```

```
Out[ ]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
  18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
  34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50}
```

+21.4 Use ImageData, Flatten and Total to find the number of white pixels in

Binarize[Rasterize[“W”]].

```
In[ ]:= Total[Flatten[ImageData[Binarize[Rasterize["W"]]]]]
Out[ ]= 419
```

+21.5 Use Flatten, IntegerDigits and Total to find the sum of all digits of all whole numbers up to 1000.

```
In[ ]:= Total[Flatten[IntegerDigits[Range[1000]]]]
Out[ ]= 13 501
```

+21.6 Generate a graph with 200 connections, each between nodes with numbers randomly chosen between 0 and 100.

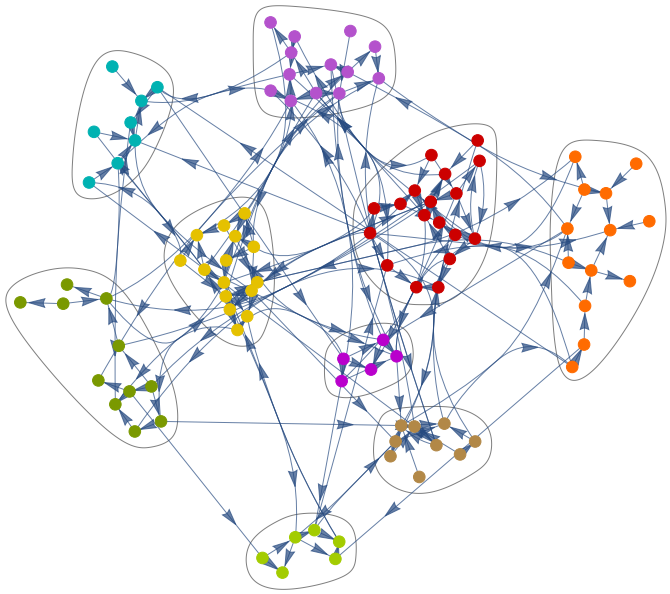
```
In[ ]:= Graph[Table[RandomInteger[100] → RandomInteger[100], 200]]
```

Out[ ]=

+21.7 Generate a plot that shows communities in a random graph with nodes numbered between 0 and 100, and 200 connections.

```
CommunityGraphPlot[Graph[Table[RandomInteger[100] → RandomInteger[100], 200]]]
```

Out[ ]=



Exercises for Section 22 | Machine Learning

22.1 Identify what language the word “ajatella” comes from.

In[ ]:=
```
LanguageIdentify["ajatella"]
```

Out[ ]= Finnish

22.2 Apply ImageIdentify to an image of a tiger, getting the image using ctrl +=.

In[ ]:=
```
ImageIdentify[ tiger SPECIES SPECIFICATION [ image ]]
```

Out[ ]= tiger

22.3 Make a table of image identifications for an image of a tiger, blurred by an amount from 1 to 5.

In[ ]:=
```
Table[ImageIdentify[Blur[ tiger SPECIES SPECIFICATION [ image ] ⋯ ✓ , r]], {r, 1, 5}]
```

Out[ ]= { tiger , tiger , tiger , spiny-finned fish , fish }

22.4 Classify the sentiment of “I’m so happy to be here”.

```
In[ ]:= Classify["Sentiment", "I'm so happy to be here"]
Out[ ]= Positive
```

22.5 Find the 10 words in WordList[] that are nearest to “happy”.

```
In[ ]:= Nearest[WordList[], "happy", 10]
Out[ ]= {happy, haply, harpy, nappy, sappy, apply, campy, choppy, guppy, hairy}
```

22.6 Generate 20 random numbers up to 1000 and find which 3 are nearest to 100.

```
In[ ]:= Nearest[RandomInteger[1000, 20], 100, 3]
Out[ ]= {103, 91, 65}
```

22.7 Generate a list of 10 random colors, and find which 5 are closest to Red.

```
In[ ]:= Nearest[RandomColor[10], Red, 5]
```

Out[ ]= {■, ■, ■, ■, ■}

22.8 Of the first 100 squares, find the one nearest to 2000.

```
In[ ]:= Nearest[Range[100]^2, 2000]
Out[ ]= {2025}
```

22.9 Find the 3 European flags nearest to the flag of Brazil.

```
In[ ]:= Nearest[ Europe COUNTRIES [ flag ], Brazil COUNTRY [ flag ], 3]
```

Take : Nonatomic expression expected at position 1 in Take [$Failed, 6].

Take : Nonatomic expression expected at position 1 in Take [$Failed, 6].

Out[ ]= {  ,  ,  }

```
In[ ]:= Nearest[ Europe COUNTRIES [ flag ], Argentina COUNTRY [ flag ], 3]
```

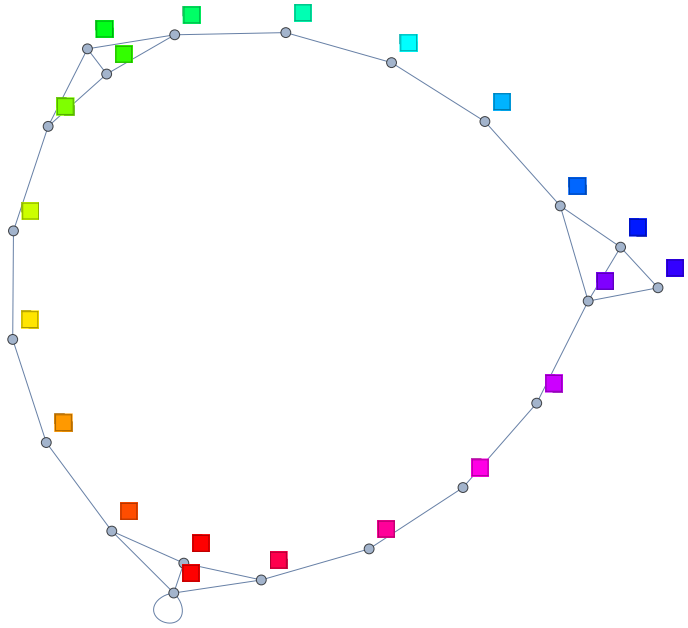
Out[ ]= {  ,  ,  }

22.10 Make a graph of the 2 nearest neighbors of each color in Table[Hue[n],

{h, 0, 1, 0.05}].

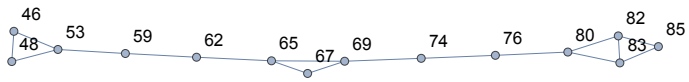
In[ ]:=
```
NearestNeighborGraph[Table[Hue[h], {h, 0, 1, 0.05}], 2, VertexLabels → All]
```

Out[ ]=

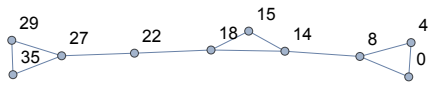


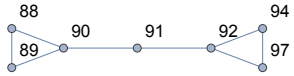
22.11 Generate a list of 100 random numbers from 0 to 100, and make a graph of the 2 nearest neighbors of each one.

In[ ]:=
```
NearestNeighborGraph[Table[RandomInteger[100], 40], 2, VertexLabels → All]
```



Out[ ]=



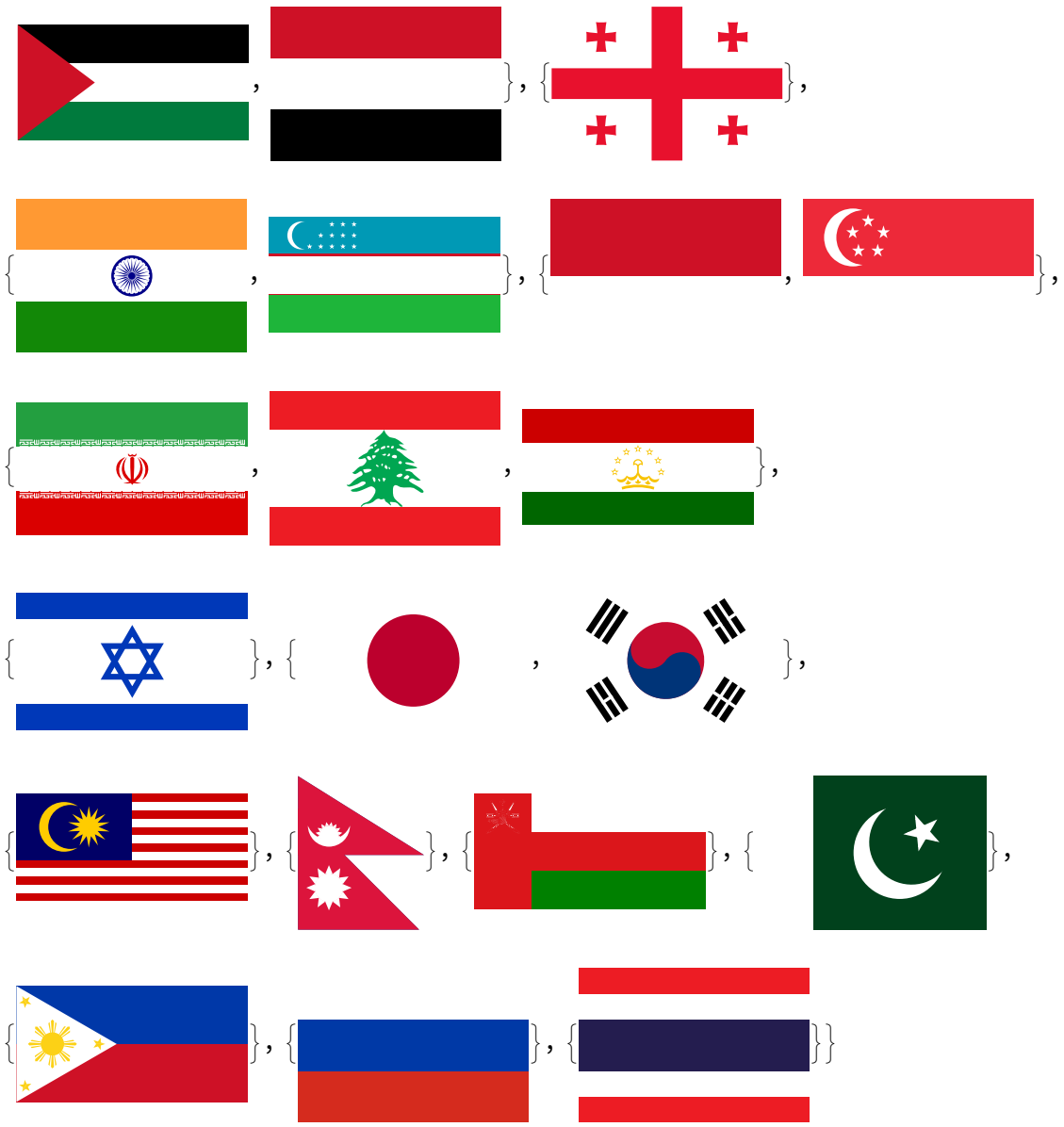


22.12 Collect the flags of Asia into clusters of similar flags.

In[ ]:=
```
FindClusters[ Asia COUNTRIES [ flag ] ✓ ]
```

Take : Nonatomic expression expected at position 1 in Take [$Failed, 6].

Out[ ]=



22.13 Make raster images of the letters of the alphabet at size 20, then make a graph of the 2 nearest neighbors of each one.

```
In[ ]:= Table[Style[Rasterize[j], ImageSize → 20], {j, Alphabet[]}]
```

```
Out[ ]= {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z}
```