Prise en main du langage

Les exercices sont issus de la documentation officielle :
https : // www.wolfram.com / language / elementary – introduction / 2 nd – ed / index.html ;
Julien VALENTIN
Novembre 2020

Exercices chapitre 1 : “Starting out : Elementary Arithmetic”

In[•]:= `1 + 2 + 3`

Out[•]= 6

In[•]:= `1 + 2 + 3 + 4 + 5`

Out[•]= 15

In[•]:= `1 * 2 * 3 * 4 * 5`

Out[•]= 120

In[•]:= `1 × 2 × 3 × 4 × 5`

Out[•]= 120

In[•]:= `5^2`

Out[•]= 25

In[•]:= `3^4`

Out[•]= 81

In[•]:= `10^12`

Out[•]= 1 000 000 000 000

In[•]:= `3 ^ (7 × 8)`

Out[•]= 523 347 633 027 360 537 213 511 521

In[•]:= `(4 - 2) * (4 + 3)`

Out[•]= 14

In[•]:= `29 000 × 73`

Out[•]= 2 117 000

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

Out[•]= 0

```
In[•]:= 24 / 3
Out[•]= 8

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

In[•]:= 100 - 5 ^ 2
Out[•]= 75

In[•]:= 6 * 5 ^ 2 + 7
Out[•]= 157

In[•]:= 3 ^ 2 - 2 ^ 3
Out[•]= 1

In[•]:= 2 ^ 3 × 3 ^ 2
Out[•]= 72
```

Exercices chapitre 2 : “Introducing Functions”

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

In[•]:= Times[2, Plus[3, 4]]
        multiplicat···  plus
Out[•]= 14

In[•]:= Max[Times[6, 8], Times[5, 9]]
        ma··  multiplication  multiplication
Out[•]= 48

In[•]:= RandomInteger[1000]
        entier aléatoire
Out[•]= 244

In[•]:= Plus[10, RandomInteger[10]]
        plus  entier aléatoire
Out[•]= 13

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

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

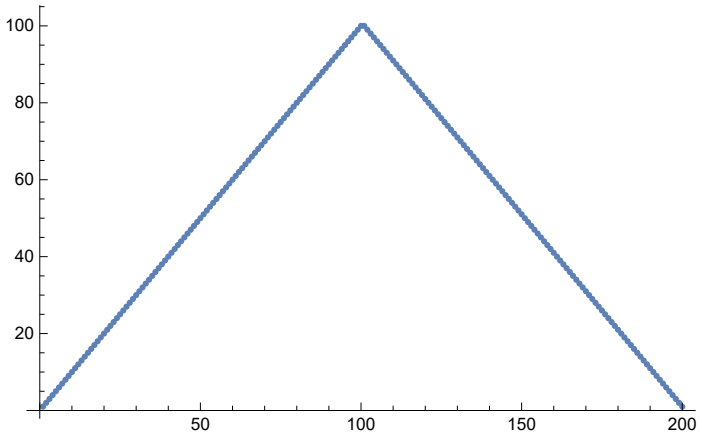
```
In[•]:= Times[Plus[8, 7], Plus[9, 2]]
        multip·· plus  plus
Out[•]= 165

In[•]:= Divide[Subtract[26, 89], 9]
        divise  soustrait
Out[•]= -7

In[•]:= Subtract[100, Power[5, 2]]
        soustrait  puissance
Out[•]= 75

In[•]:= Max[5^3, 3^5]
        maximum
Out[•]= 243

In[•]:= Times[3, Max[4^3, 3^4]]
        multiplicat··· maximum
Out[•]= 243

In[•]:= Plus[RandomInteger[1000], RandomInteger[1000]]
        plus  entier aléatoire  entier aléatoire
Out[•]= 1476
```

Exercices chapitre 3 : “First look at Lists”

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

In[•]:= Range[100]
        plage
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}

In[•]:= Reverse[Range[4]]
        renverses  plage
Out[•]= {4, 3, 2, 1}

In[•]:= Reverse[Range[50]]
        renverses  plage
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}

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

In[•]:= **ListPlot[Join[Range[1, 100], Reverse[Range[1, 100]]]]**
tracé de li··· joins plage renverses plage

Out[•]=



In[•]:= **Range[RandomInteger[10]]**
plage entier aléatoire

Out[•]= {1, 2, 3, 4}

In[•]:= **Range[10]**
plage

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

In[•]:= **Range[5]**
plage

Out[•]= {1, 2, 3, 4, 5}

In[•]:= **Join[Range[10], Range[10], Range[5]]**
joins plage plage plage

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}

In[•]:= **Join[Reverse[Range[20]], Range[20]]**
joins renverses plage plage

Out[•]= {20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2,
1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}

In[•]:= **Reverse[Reverse[{1, 2, 3, 4}]]**
renverses renverses

Out[•]= {1, 2, 3, 4}

In[•]:= **Join[Range[5], Reverse[Range[4]]]**
joins plage renverses plage

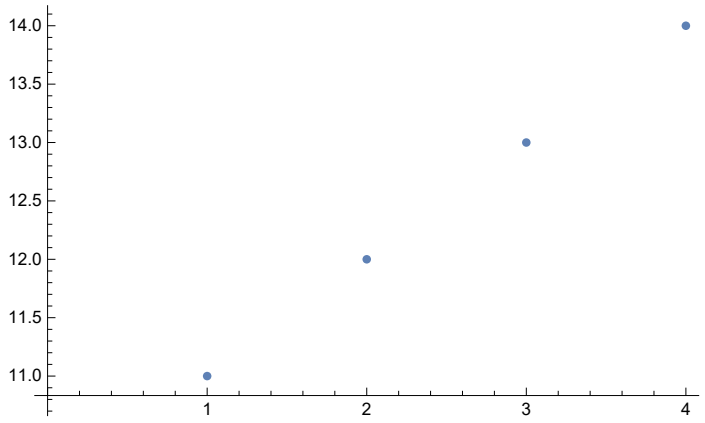
Out[•]= {1, 2, 3, 4, 5, 4, 3, 2, 1}

In[•]:= **Reverse[Join[Range[5], Range[4], Range[3]]]**
renverses joins plage plage plage

Out[•]= {3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1}

In[•]:= **ListPlot[10 + Range[4]]**
tracé de liste plage

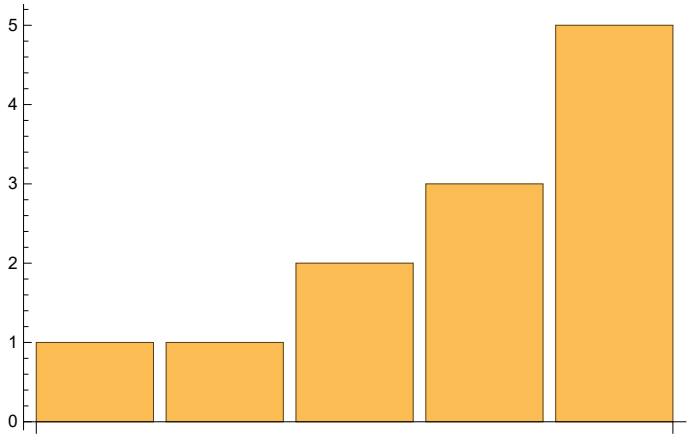
Out[•]=



Exercices chapitre 4 : “Displaying Lists”

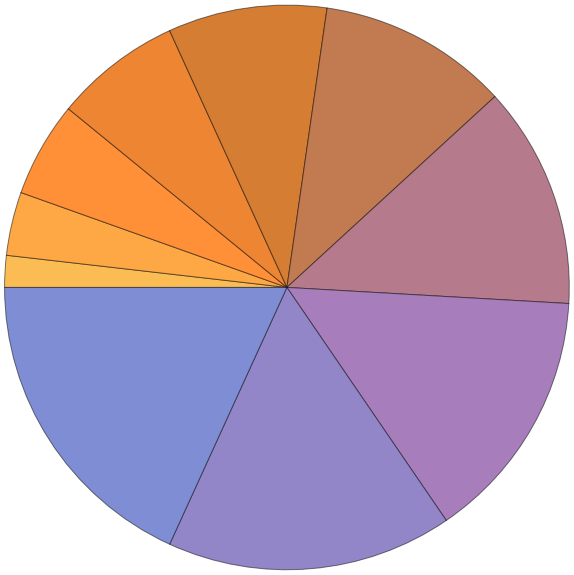
In[•]:= **BarChart[{1, 1, 2, 3, 5}]**
diagramme à barres

Out[•]=



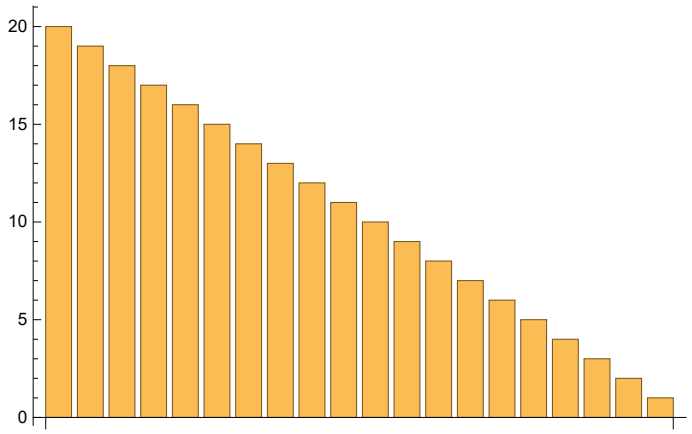
In[ ]:=
```
PieChart[Range[10]]
```
diagramm·· | plage

Out[ ]=



In[ ]:=
```
BarChart[Reverse[Range[1, 20]]]
```
diagramm·· | renverses | plage

Out[ ]=



In[ ]:=
```
Column[Range[1, 5]]
```
colonne | plage

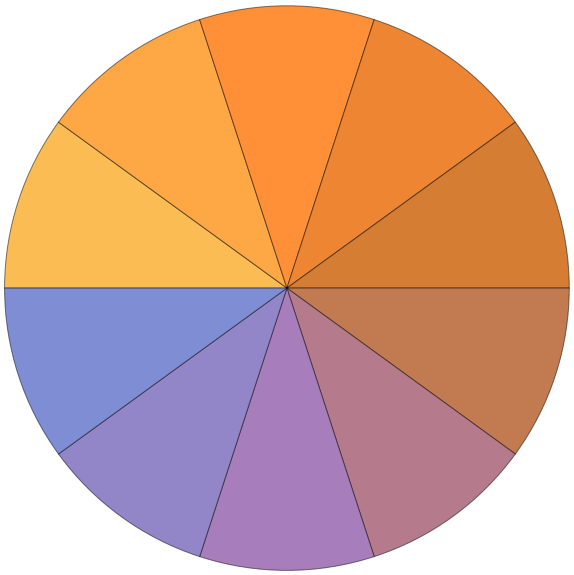
Out[ ]=
```
1
2
3
4
5
```

In[ ]:=
```
NumberLinePlot[{1, 4, 9, 16, 25}]
```
tracé de ligne numérique

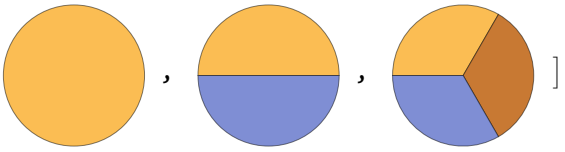
Out[ ]=

```
In[ ]:= PieChart[Range[10] / Range[10]]
        diagramm··  plage      plage
```

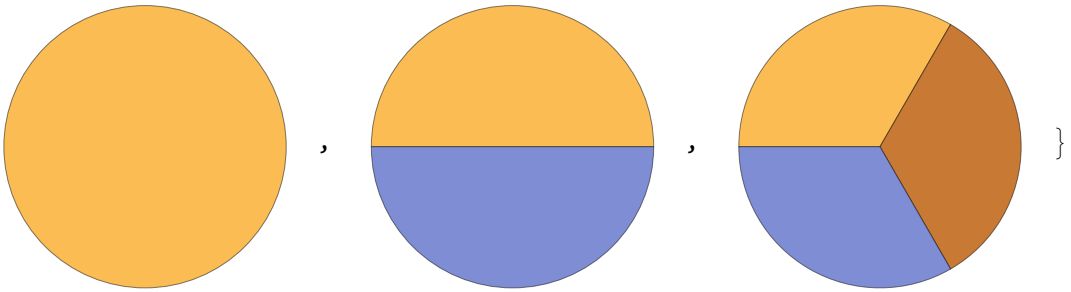
Out[ ]=



```
In[ ]:= Column[PieChart[Range[1]], PieChart[Range[2] / Range[2]], PieChart[Range[3] / Range[3]]]
        colonne  diagramm··  plage      diagramm··  plage      plage      diagramm··  plage      plage
```

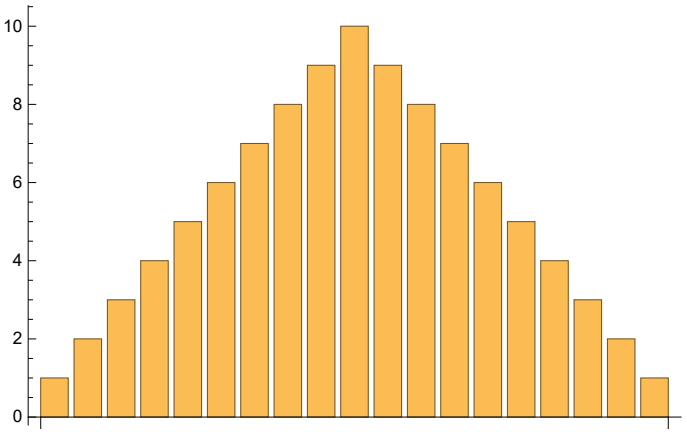
Out[ ]= Column[]

```
In[ ]:= {PieChart[Range[1]], PieChart[Range[2] / Range[2]], PieChart[Range[3] / Range[3]]}
         diagramm··  plage      diagramm··  plage      plage      diagramm··  plage      plage
```

Out[ ]= {  }

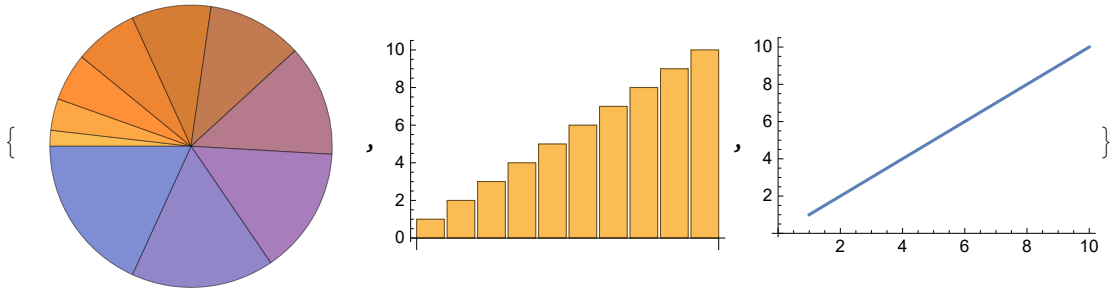
In[]:=
```
BarChart[Join[Range[1, 10], Reverse[Range[1, 9]]]]
```
diagramm·· joins plage renverses plage

Out[]=



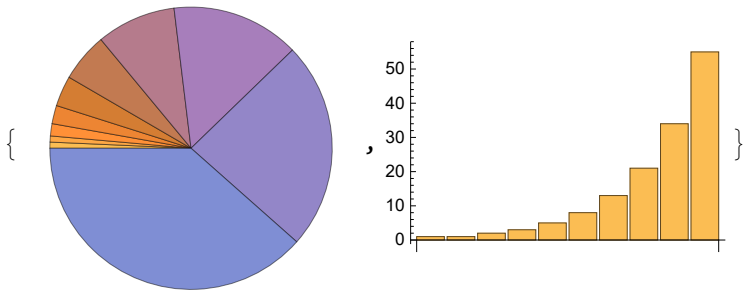
In[]:=
```
MyList = Range[10]; {PieChart[MyList], BarChart[MyList], ListLinePlot[MyList]}
```
plage diagramme circulaire diagramme à barres tracé de liste de ligne

Out[]=



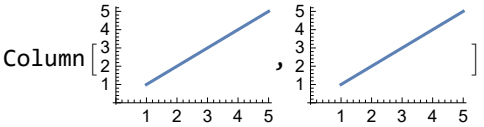
In[]:=
```
MyList = Fibonacci[Range[10]]; {PieChart[MyList], BarChart[MyList]}
```
Fibonacci plage diagramme circulaire diagramme à barres

Out[]=



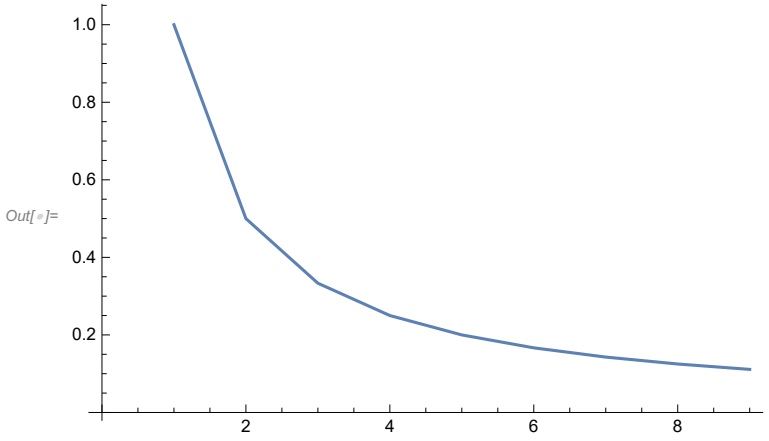
In[]:=
```
MyList = Range[5]; Column[ListLinePlot[MyList], ListLinePlot[MyList]]
```
plage colonne tracé de liste de ligne tracé de liste de ligne

Out[]=



In[•]:= `MyList = 1 / Range[9]; ListLinePlot[MyList]`
plage tracé de liste de ligne

Out[•]=



Exercices chapitre 5 : “Operations on Lists”

In[•]:= `Range[10]^2`
plage

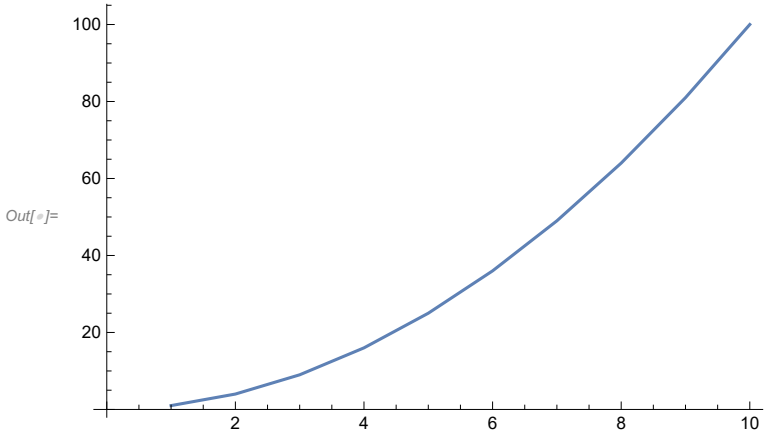
Out[•]= {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}

In[•]:= `Total[Range[10]^2]`
total plage

Out[•]= 385

In[•]:= `ListLinePlot[Range[1, 10]^2]`
tracé de liste de ··· plage

Out[•]=



In[•]:= `Sort[Join[Range[4], Range[4]]]`
trie joins plage plage

Out[•]= {1, 1, 2, 2, 3, 3, 4, 4}

In[•]:= `10 + Range[0, 10]`
plage

Out[•]= {10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}

In[]:= **Sort[Join[Range[5]^2, Range[5]^3]]**
trie joins plage plage

Out[]= {1, 1, 4, 8, 9, 16, 25, 27, 64, 125}

In[]:= **Length[IntegerDigits[2^128]]**
longueur chiffres d'entier

Out[]= 39

In[]:= **First[IntegerDigits[2^32]]**
premier chiffres d'entier

Out[]= 4

In[]:= **Take[IntegerDigits[2^100], 10]**
pre··· chiffres d'entier

Out[]= {1, 2, 6, 7, 6, 5, 0, 6, 0, 0}

In[]:= **Max[IntegerDigits[2^20]]**
ma·· chiffres d'entier

Out[]= 8

In[]:= **Count[IntegerDigits[2^1000], 0]**
compte chiffres d'entier

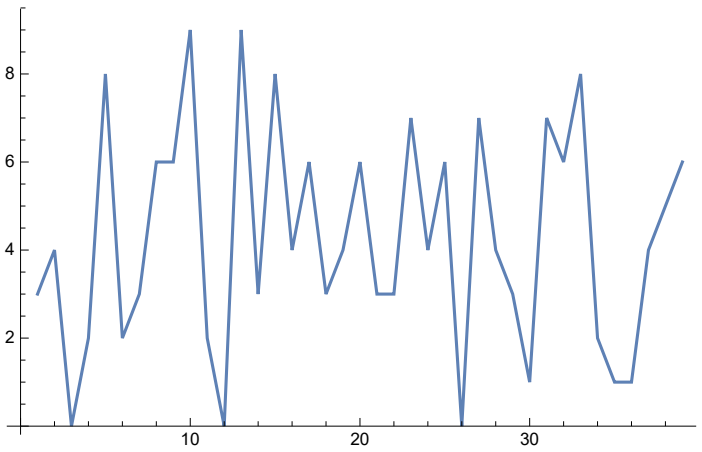
Out[]= 28

In[]:= **Part[Sort[IntegerDigits[2^20]], 2]**
partie trie chiffres d'entier

Out[]= 1

In[]:= **ListLinePlot[IntegerDigits[2^128]]**
tracé de liste de ·· chiffres d'entier

Out[]=



In[]:= **Take[Drop[Range[100], 10], 10]**
pre··· laiss·· plage

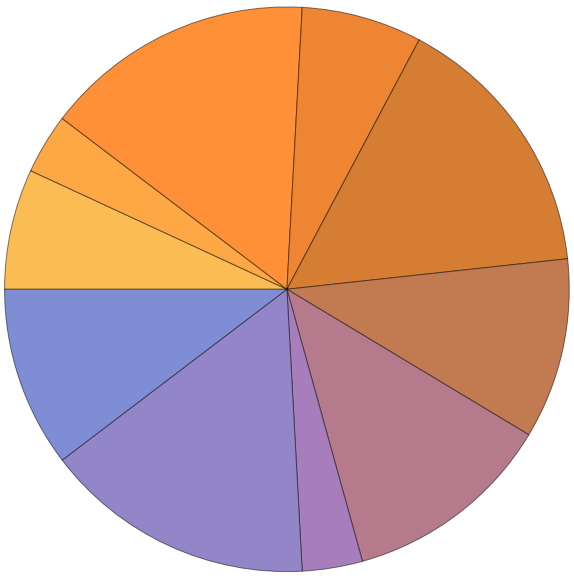
Out[]= {11, 12, 13, 14, 15, 16, 17, 18, 19, 20}

In[]:= **3 Range[10]**
plage

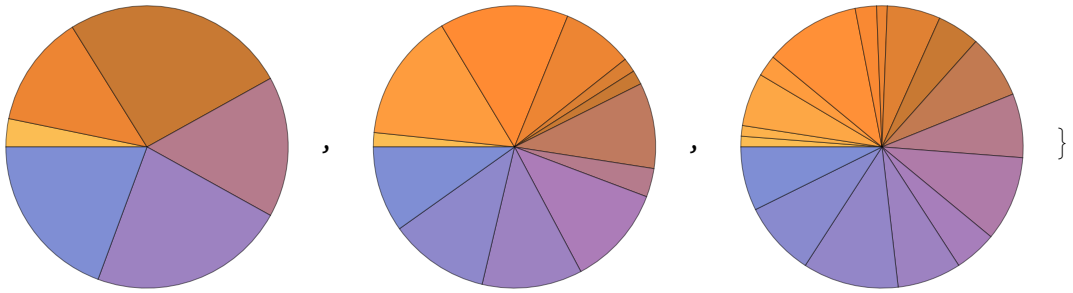
Out[]= {3, 6, 9, 12, 15, 18, 21, 24, 27, 30}

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

In[•]:= Last[IntegerDigits[2^37]]
        dern·· chiffres d'entier
Out[•]= 2

In[•]:= Reverse[Take[Reverse[IntegerDigits[2^32]], 2]]
        renverses pre··· renverses chiffres d'entier
Out[•]= {9, 6}

In[•]:= Total[IntegerDigits[3^126]]
        total  chiffres d'entier
Out[•]= 234

In[•]:= PieChart[IntegerDigits[2^32]]
        diagramm·· chiffres d'entier
```

Out[•]=



```
In[•]:= {PieChart[IntegerDigits[2^20]],
          diagramm··  chiffres d'entier
         PieChart[IntegerDigits[2^40]], PieChart[IntegerDigits[2^60]]}
          diagramm··  chiffres d'entier          diagramm···  chiffres d'entier
```

Out[•]= 

Exercices chapitre 6 : “Making Tables”

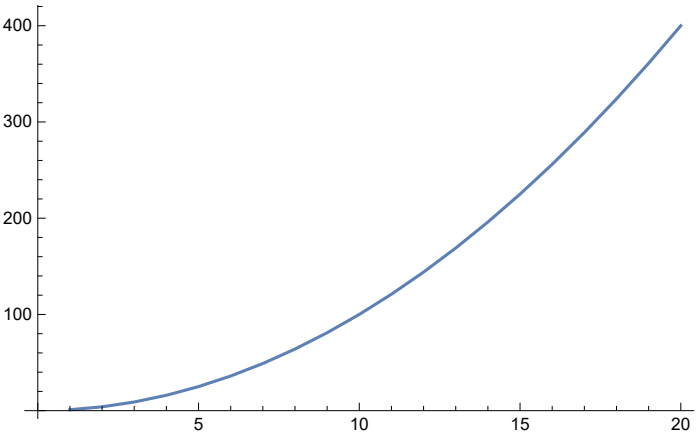
```
In[•]:= Table[1000, 5]
         table
```

```
Out[•]= {1000, 1000, 1000, 1000, 1000}
```

```
In[•]:= Table[Power[n, 3], {n, 10, 20}]
         table  puissance
```

```
Out[•]= {1000, 1331, 1728, 2197, 2744, 3375, 4096, 4913, 5832, 6859, 8000}
```

```
In[•]:= ListLinePlot[Table[Power[n, 2], {n, 1, 20}]]
         tracé de liste de ··  table  puissance
```

Out[•]= 


```
In[•]:= Range[2, 20, 2]
         plage
```

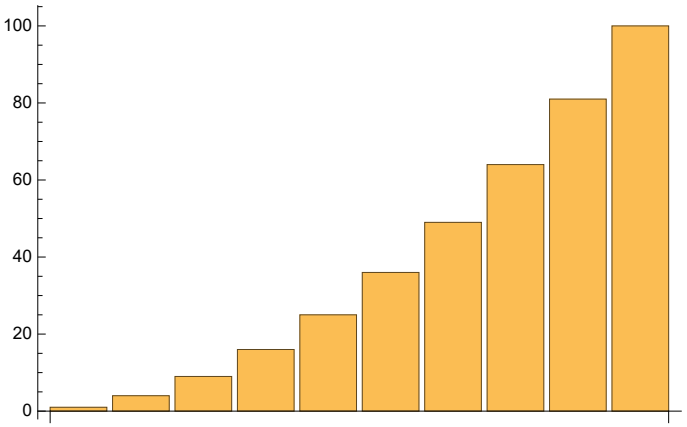
```
Out[•]= {2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
```

```
In[•]:= Table[n, {n, 2, 20, 2}]
         table
```

```
Out[•]= {2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
```

In[ ]:= **BarChart[Table[Power[n, 2], {n, 10}]]**
diagramm·· table puissance

Out[ ]=

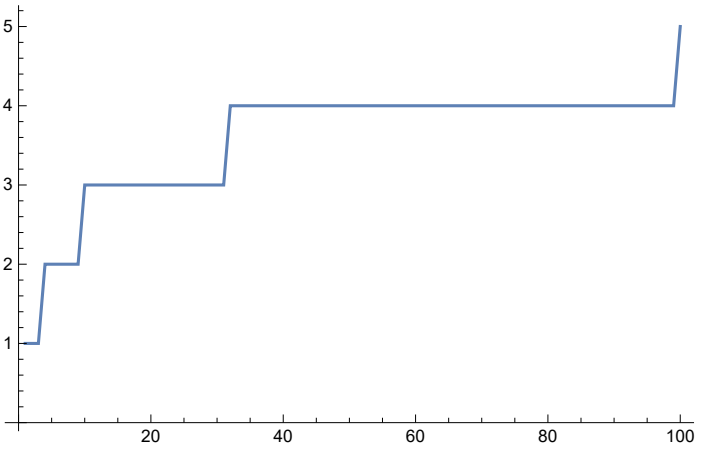


In[ ]:= **Table[IntegerDigits[n^2], {n, 10}]**
table chiffres d'entier

Out[ ]= {{1}, {4}, {9}, {1, 6}, {2, 5}, {3, 6}, {4, 9}, {6, 4}, {8, 1}, {1, 0, 0}}

In[ ]:= **ListLinePlot[Table[Length[IntegerDigits[n^2]], {n, 100}]]**
tracé de liste de ·· table longueur chiffres d'entier

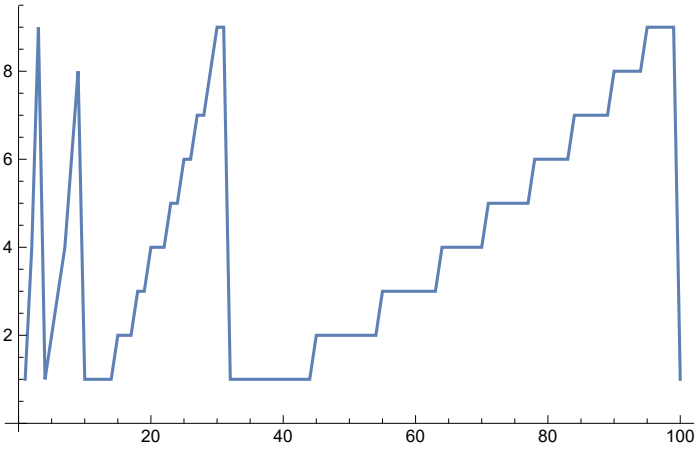
Out[ ]=



In[ ]:= **Table[First[IntegerDigits[n^2]], {n, 20}]**
table premier chiffres d'entier

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

```
In[•]:= ListLinePlot[Table[First[IntegerDigits[n^2]], {n, 100}]]
        tracé de liste de ·· table  premier chiffres d'entier
```

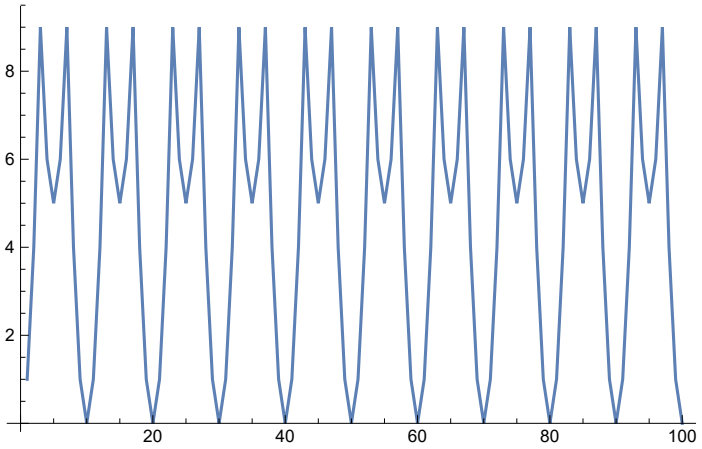



```
In[•]:= Table[n^3 - n^2, {n, 10}]
        table
Out[•]= {0, 4, 18, 48, 100, 180, 294, 448, 648, 900}

In[•]:= Range[1, 100, 2]
        plage
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}

In[•]:= Table[n^2, {n, 2, 100, 2}]
        table
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, 10000}

In[•]:= Range[-3, 2]
        plage
Out[•]= {-3, -2, -1, 0, 1, 2}

In[•]:= Table[Column[{n, n^2, n^3}], {n, 20}]
        table  colonne
Out[•]= {  1   2   3   4   5   6   7   8   9   10
         { 1 , 4 , 9 , 16 , 25 , 36 , 49 , 64 , 81 , 100 ,
           1   8   27  64  125  216  343  512  729  1000

          11   12   13   14   15   16   17   18   19   20
         { 121 , 144 , 169 , 196 , 225 , 256 , 289 , 324 , 361 , 400 }
          1331 1728 2197 2744 3375 4096 4913 5832 6859 8000
```

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

tracé de liste de ·· table premier renverses chiffres d'entier

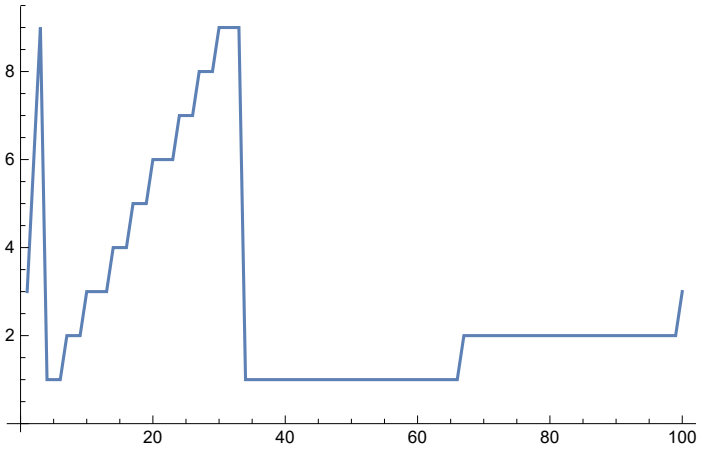
Out[]=



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

tracé de liste de ·· table premier chiffres d'entier

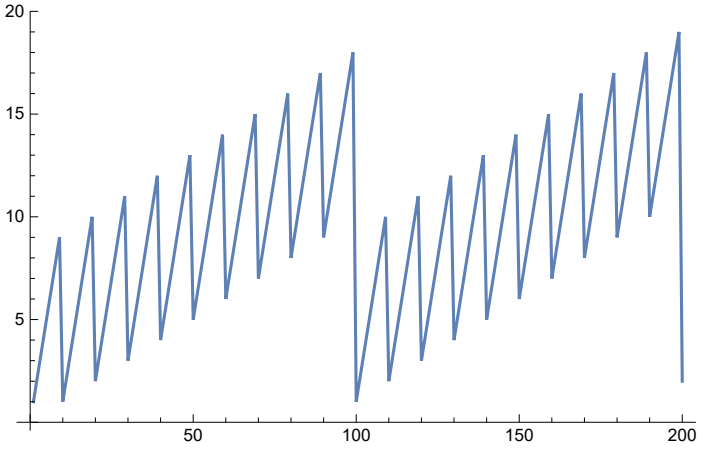
Out[]=



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

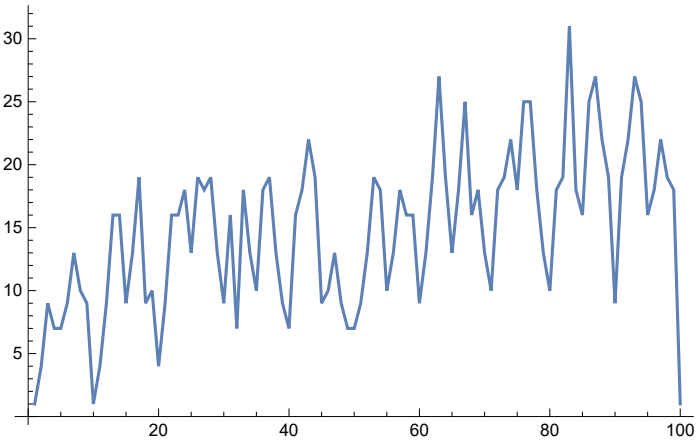
tracé de liste de ·· table total chiffres d'entier

Out[]=



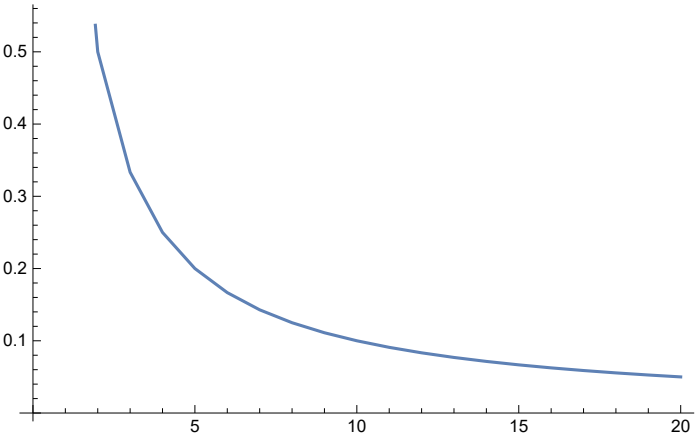
In[ ]:=
```
ListLinePlot[Table[Total[IntegerDigits[n^2]], {n, 100}]]
```
tracé de liste de ·· | table | total | chiffres d'entier

Out[ ]=



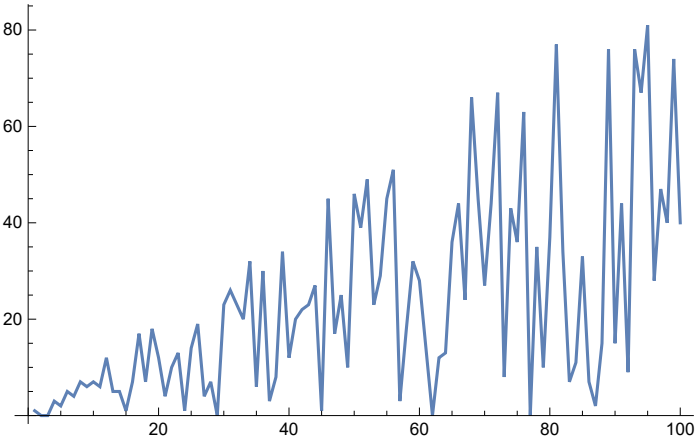
In[ ]:=
```
ListLinePlot[Table[n^(-1), {n, 1, 20}]]
```
tracé de liste de ·· | table

Out[ ]=



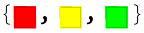
In[ ]:=
```
ListLinePlot[Table[RandomInteger[n], {n, 100}]]
```
tracé de liste de ·· | table | entier aléatoire

Out[ ]=



Exercices chapitre 7 : “Colors and Styles”

In[∘]:=
```
{Red, Yellow, Green}
```
rouge | jaune | vert

Out[∘]= 

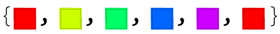
In[∘]:=
```
Column[Red, Yellow, Green]
```
colonne | ro⋯ | jaune | vert

Out[∘]= Column[]

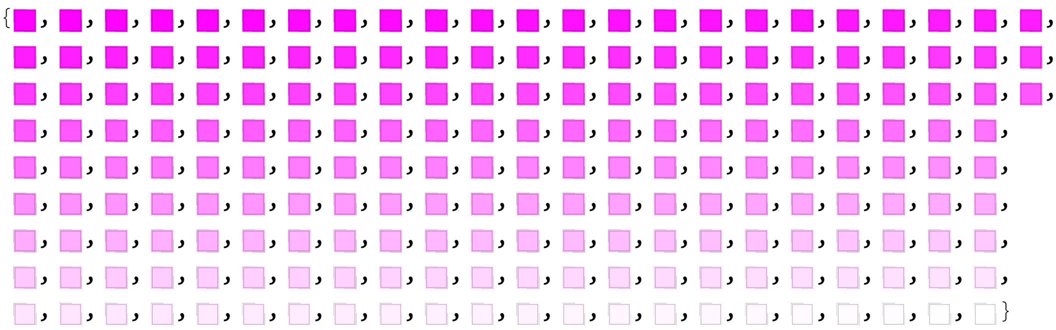
In[∘]:=
```
ColorNegate[Orange]
```
nie couleur | orange

Out[∘]= 

In[∘]:=
```
Table[Hue[x], {x, 0, 1, .2}]
```
table | teinte

Out[∘]= 

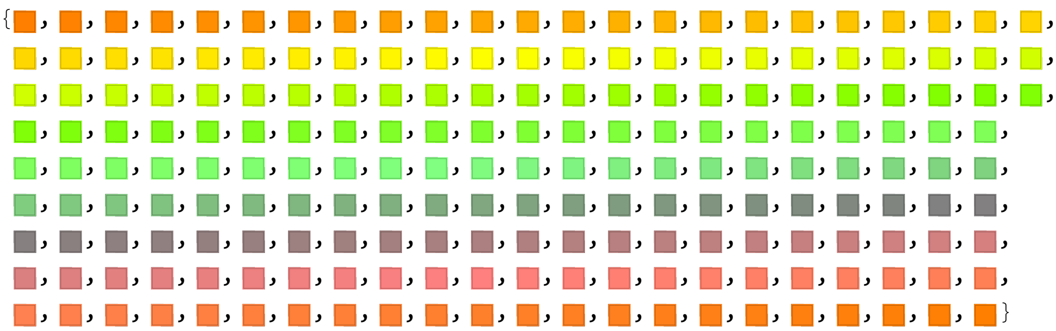
In[∘]:=
```
Table[RGBColor[1, x, 1], {x, 0, 1, .005}]
```
table | codage couleur RGB

Out[∘]= 

In[∘]:=
```
Blend[{Pink, Yellow}]
```
fonds | rose | jaune

Out[∘]= 

In[∘]:=
```
Table[Blend[{Hue[x], Yellow}], {x, 0, 1, .005}]
```
table | fonds | teinte | jaune

Out[∘]= 

In[∘]:=
```
Table[Hue[n], {n, 0, 1, .1}]
```
table | teinte

Out[∘]= 

In[ ]:= `Style[999, 100, Red]`
style rouge

Out[ ]= 999

In[ ]:= `Table[Style[n^2, n], {n, 10}]`
table style

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

In[ ]:= `Colors = {Red, Yellow, Green}; Table[Part[Colors, 1 + RandomInteger[2]], 100]`
ro··· jaune vert table partie entier aléatoire

Out[ ]= {■, ■,
■, ■,
■, ■,
■, ■}

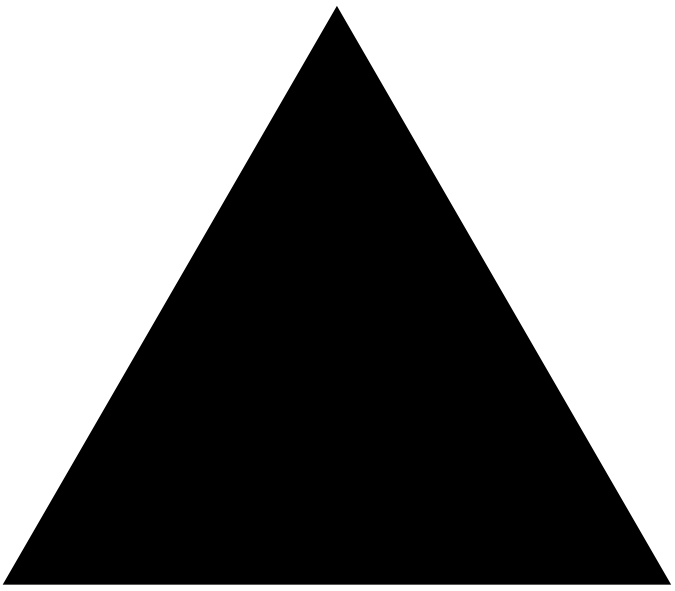
In[ ]:= `Table[Style[n, 3 n], {n, Part[IntegerDigits[2^1000], Range[50]]}]`
table style partie chiffres d'entier plage

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

Exercices chapitre 8 : “Basic Graphics objects”

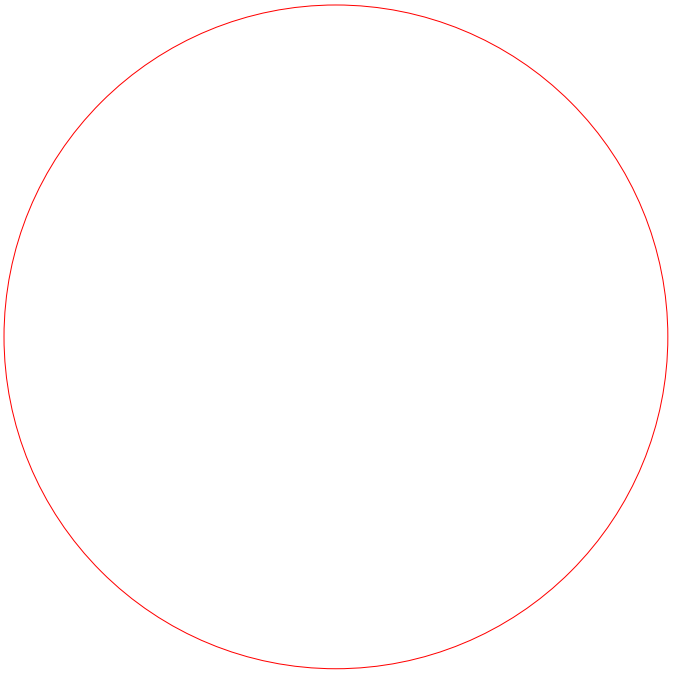
In[ ]:= `Graphics[RegularPolygon[3]]`
graphique polygone régulier

Out[ ]=



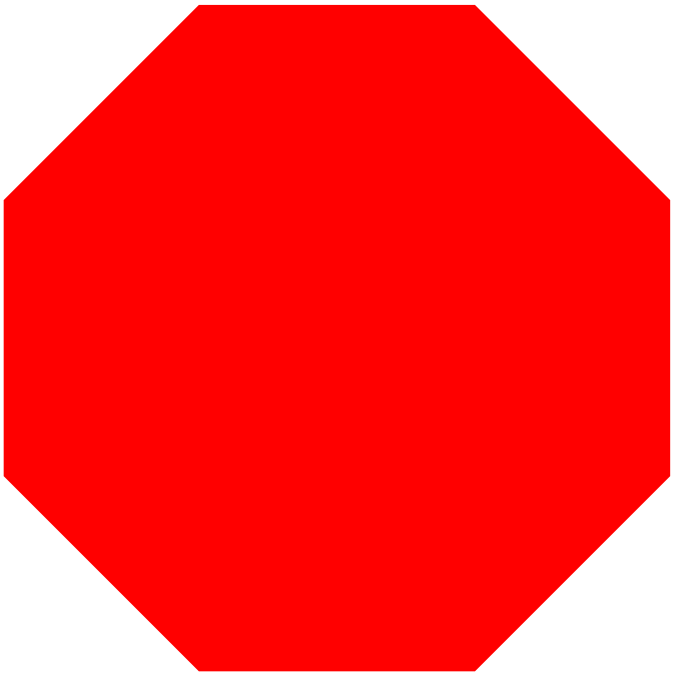
In[ ]:= **Graphics[Style[Circle[], Red]]**
graphique style cercle rouge

Out[ ]=



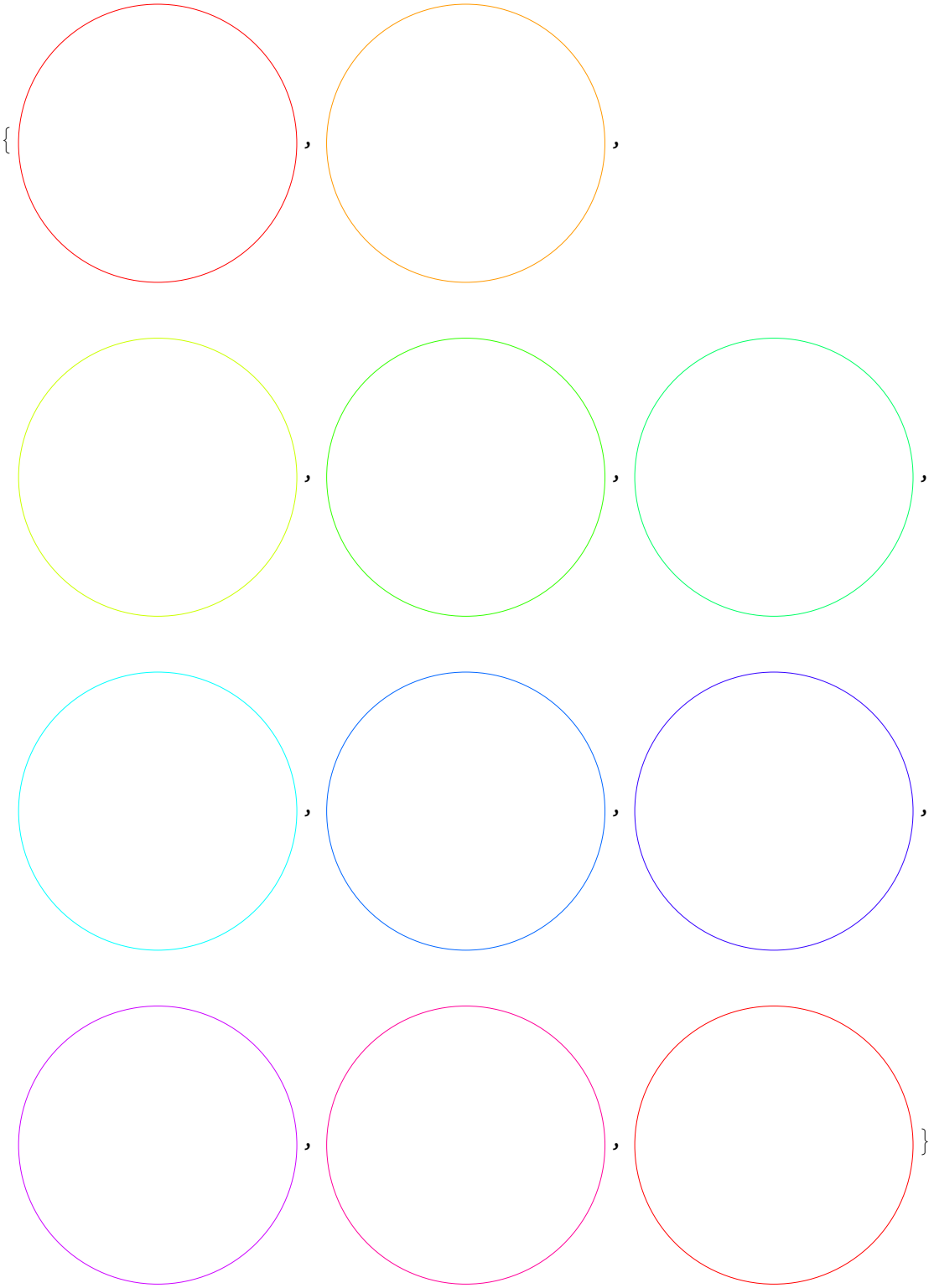
In[ ]:= **Graphics[Style[RegularPolygon[8], Red]]**
graphique style polygone régulier rouge

Out[ ]=

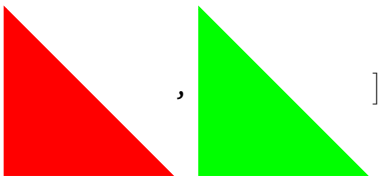


```
In[ ]:= Table[Graphics[Style[Circle[], Hue[x]]], {x, 0, 1, .1}]
        table   graphique  style  cercle      teinte
```

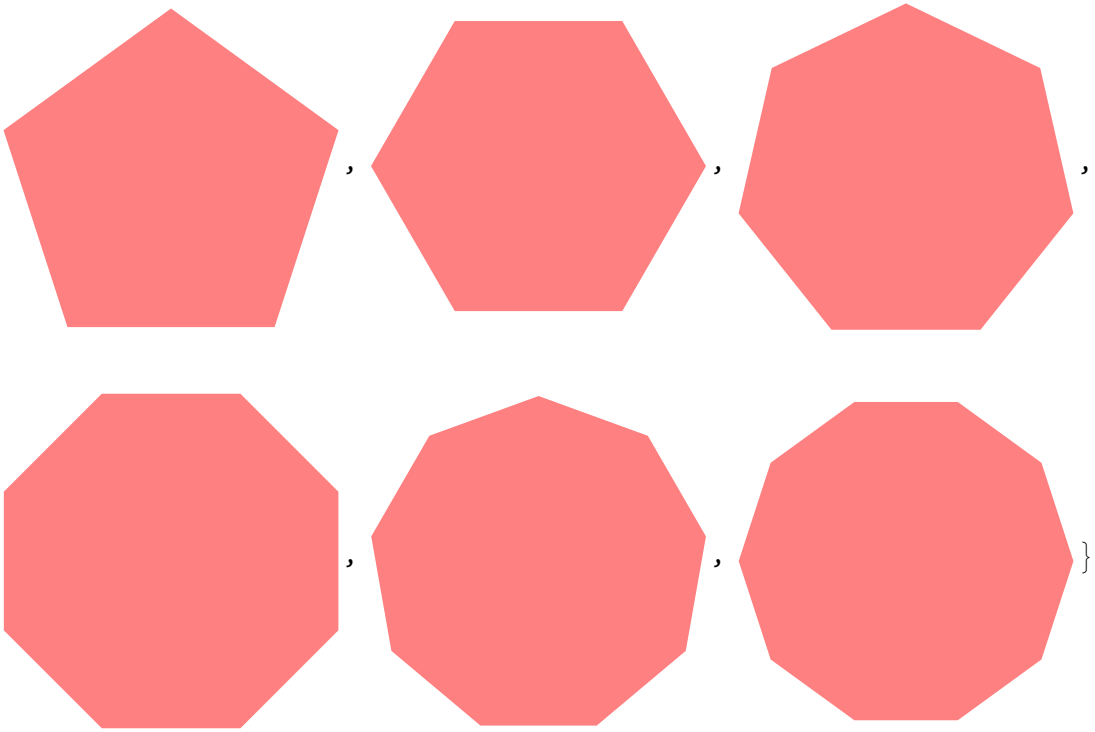
Out[ ]=



```
In[ ]:= Column[Graphics[Style[Triangle[], Red]], Graphics[Style[Triangle[], Green]]]
        colonne graphique style triangle rouge graphique style triangle vert
```

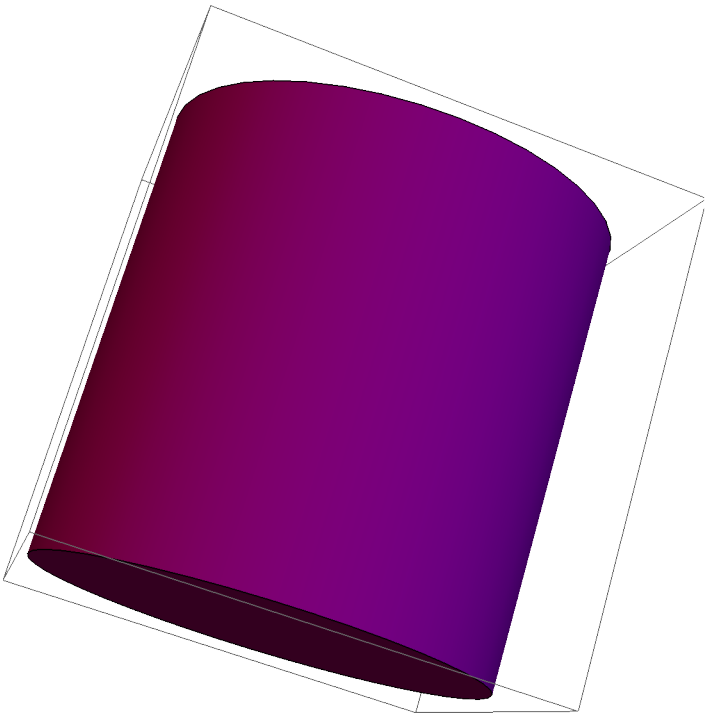
Out[ ]= Column[]

```
In[ ]:= Table[Graphics[Style[RegularPolygon[n], Pink]], {n, 5, 10}]
        table graphique style polygone régulier rose
```

Out[ ]= {  }

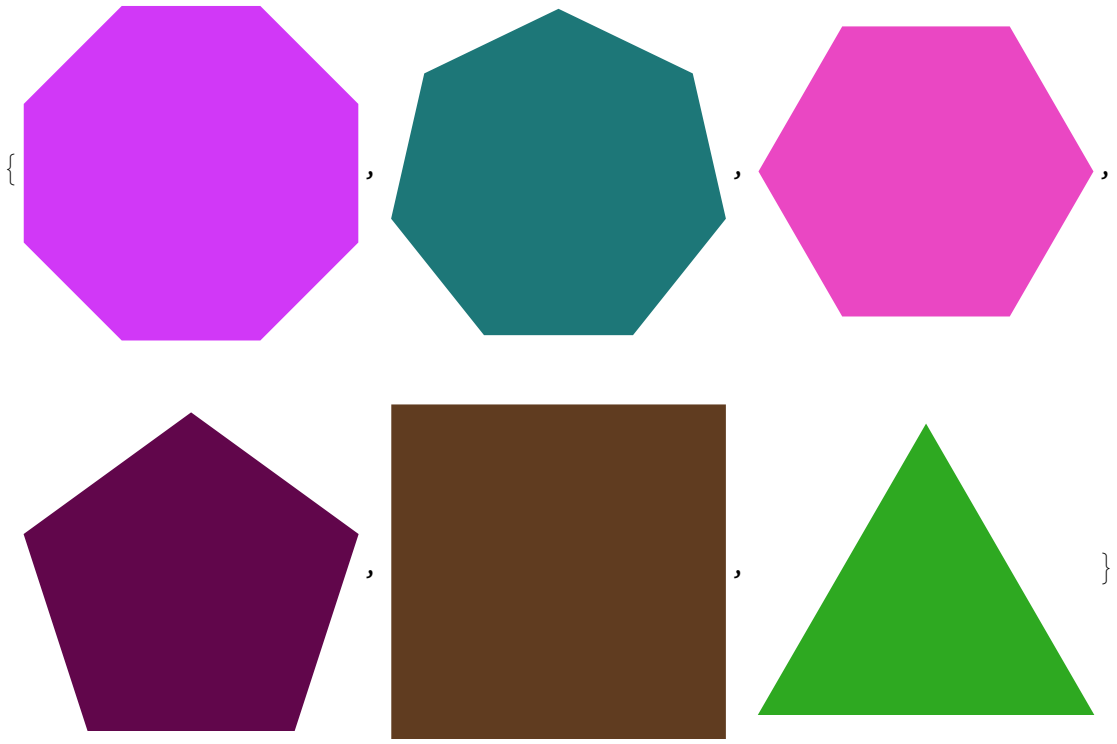
In[ ]:=
```
Graphics3D[Style[Cylinder[], Purple]]
```
graphique 3d | style | cylindre | violet

Out[ ]=



In[ ]:=
```
Table[Graphics[Style[RegularPolygon[n], RandomColor[]]], {n, Reverse[Range[3, 8]]}]
```
table | graphique | style | polygone régulier | couleur aléatoire | renverses | plage

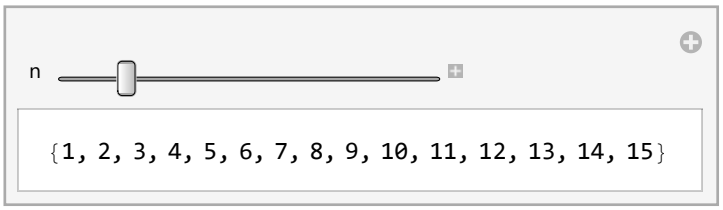
Out[ ]=



Exercices chapitre 9 : “Interactive manipulations”

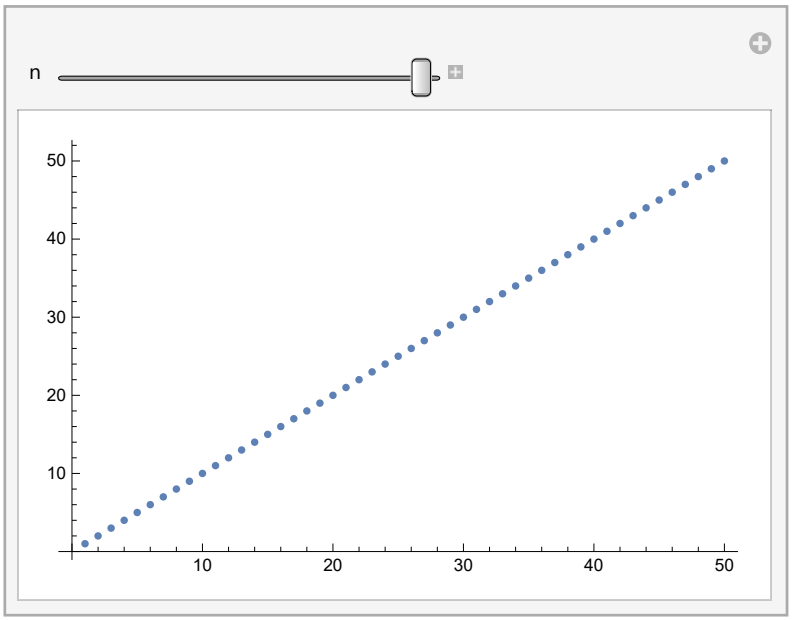
In[ ]:=
```
Manipulate[Range[n], {n, 0, 100, 1}]
```
manipule plage

Out[ ]=



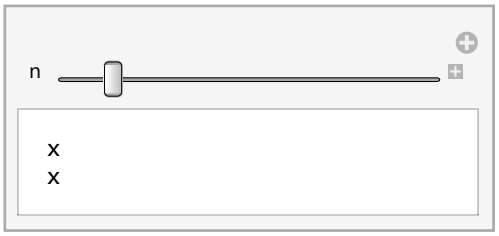
In[ ]:=
```
Manipulate[ListPlot[Range[n]], {n, 5, 50, 1}]
```
manipule tracé de li··· plage

Out[ ]=



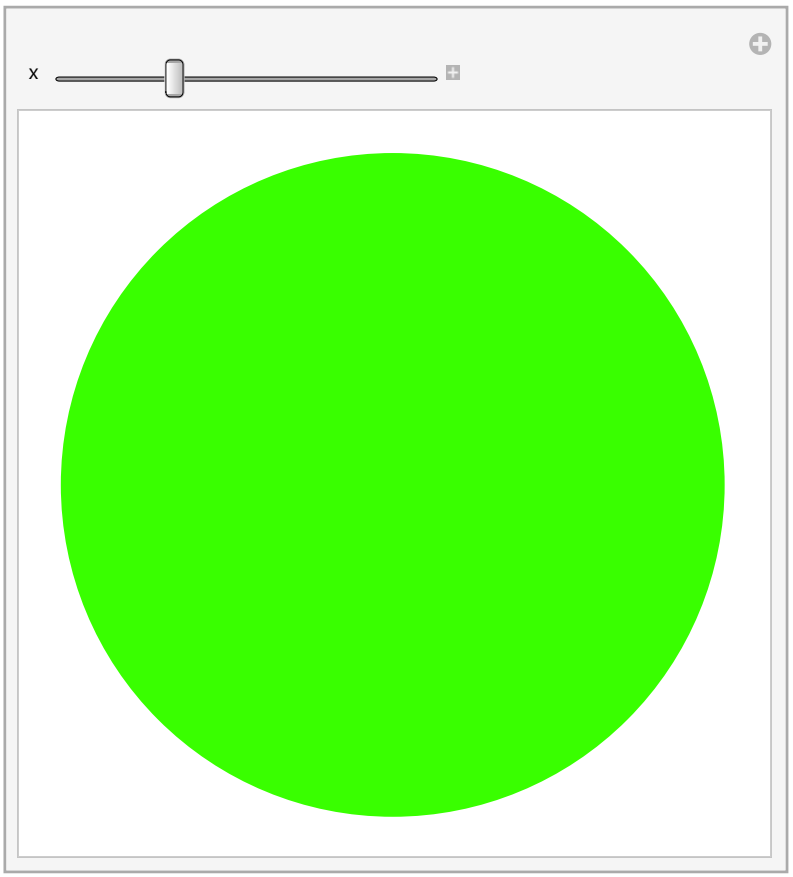
In[ ]:=
```
Manipulate[Column[Table[x, n]], {n, 1, 10, 1}]
```
manipule colonne table

Out[ ]=



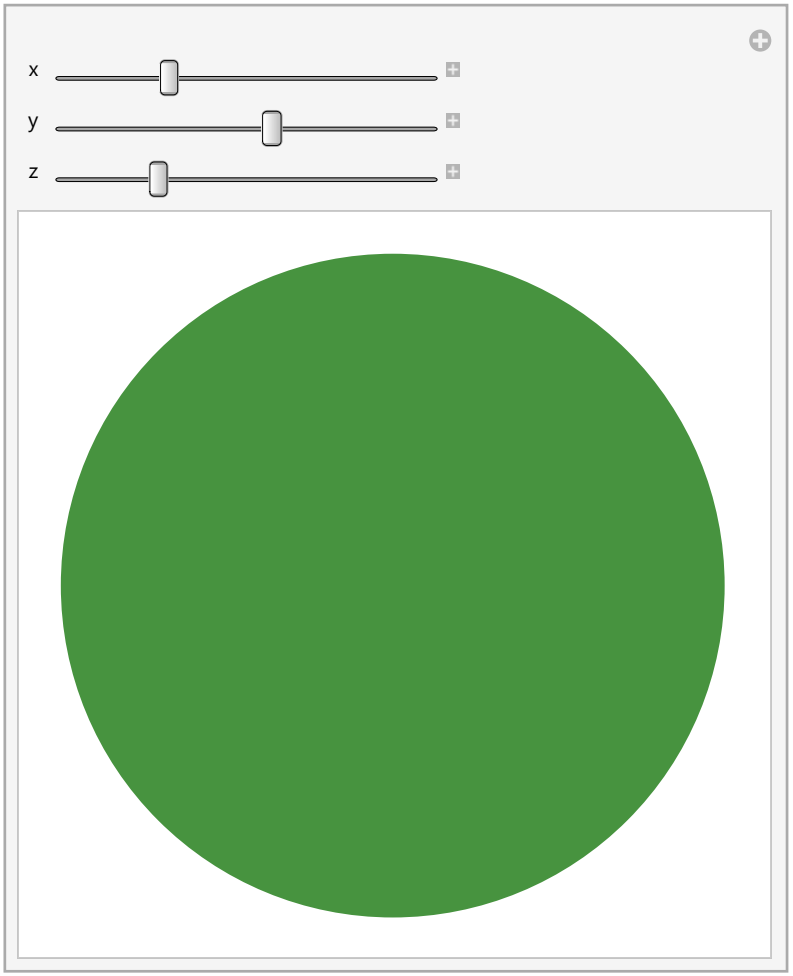
In[ ]:=
```
Manipulate[Graphics[Style[Disk[], Hue[x]]], {x, 0, 1}]
```
manipule graphique style disque teinte

Out[ ]=



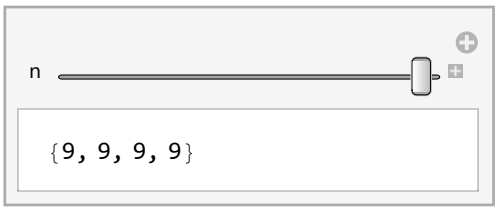
In[ ]:= `Manipulate[Graphics[Style[Disk[], RGBColor[x, y, z]]], {x, 0, 1}, {y, 0, 1}, {z, 0, 1}]`
manipule graphique style disque codage couleur RGB

Out[ ]=



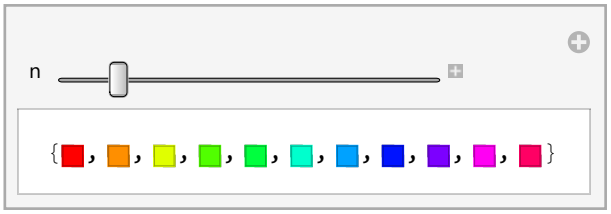
In[ ]:= `Manipulate[IntegerDigits[n], {n, 10^3, 10^4 - 1, 1}]`
manipule chiffres d'entier

Out[ ]=



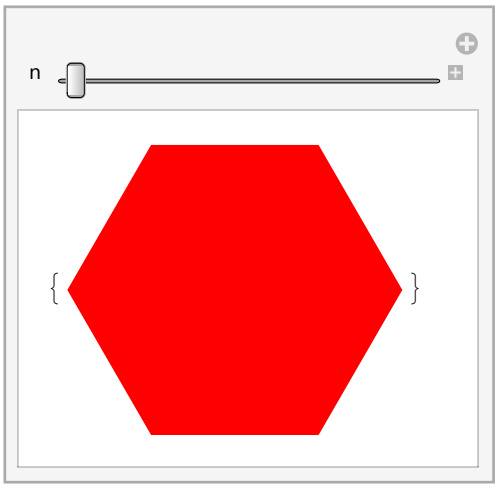
In[ ]:= `Manipulate[Table[Hue[x], {x, 0, 1, 1/n}], {n, 5, 50}]`
manipule table teinte

Out[ ]=



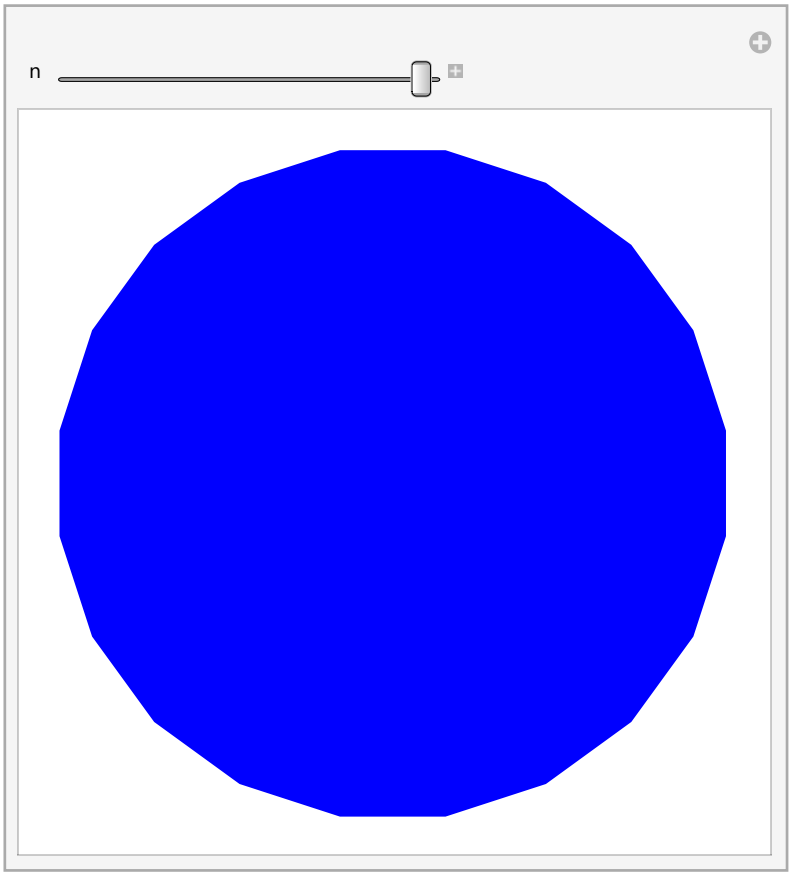
In[ ]:=
```
Manipulate[
 manipule
  Table[Graphics[Style[RegularPolygon[6], Hue[x]]], {x, 1/n, 1, 1/n}], {n, 1, 10, 1}]
  table  graphique  style  polygone régulier  teinte
```

Out[ ]=



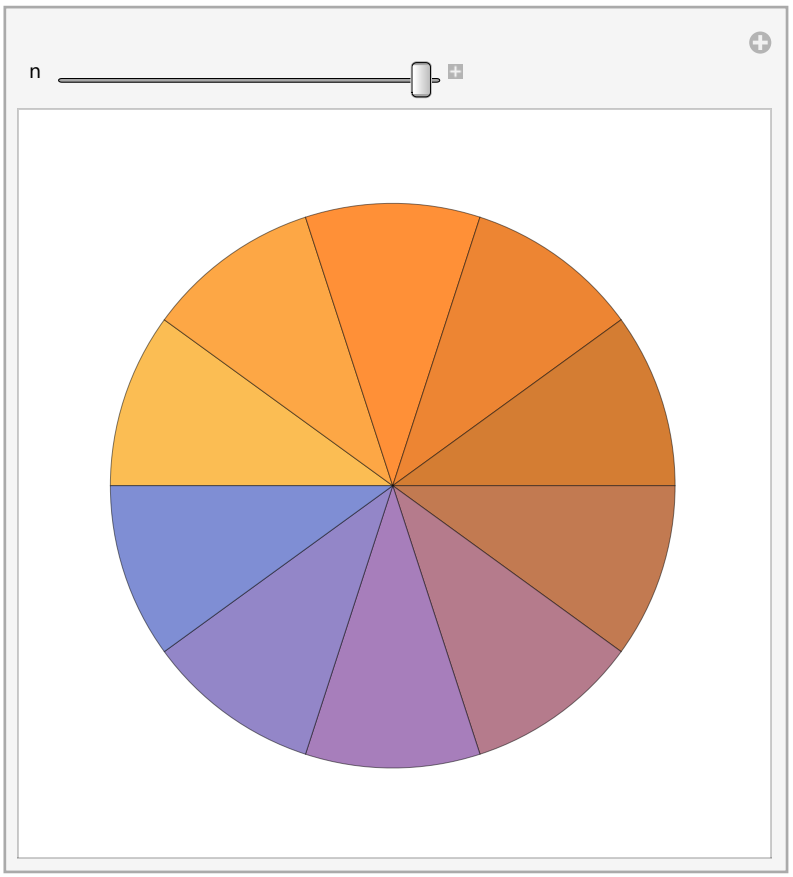
In[ ]:=
```
Manipulate[
 manipule
  Graphics[Style[RegularPolygon[n], RandomSample[{Blue, Red, Yellow}]]], {n, 5, 20, 1}]
  graphique  style  polygone régulier  échantillon aléato··  bleu  ro···  jaune
```

Out[ ]=



In[•]:=
```
Manipulate[PieChart[Table[1, n]], {n, 1, 10, 1}]
```
manipule diagramm·· table

Out[•]=

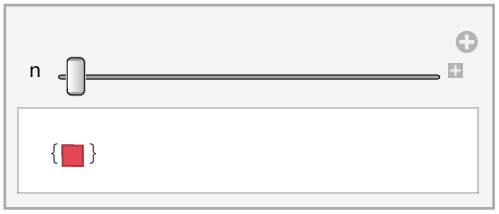


In[•]:=
```
Manipulate[BarChart[IntegerDigits[n]], {n, 100, 999, 1}]
```
manipule diagramm·· chiffres d'entier

Out[•]=

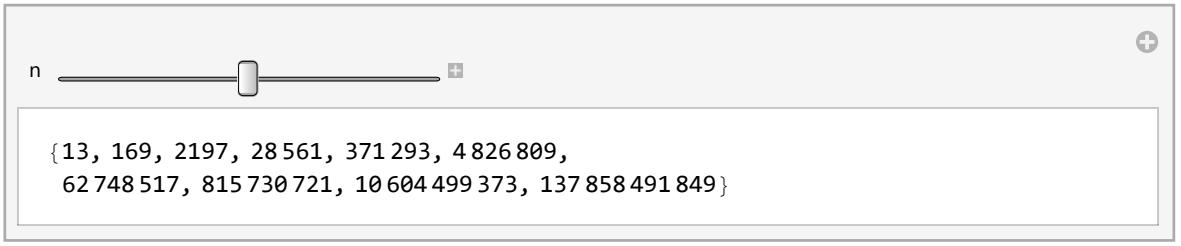
In[ ]:= **Manipulate[Table[RandomColor[], n], {n, 1, 50, 1}]**
manipule table couleur aléatoire

Out[ ]=



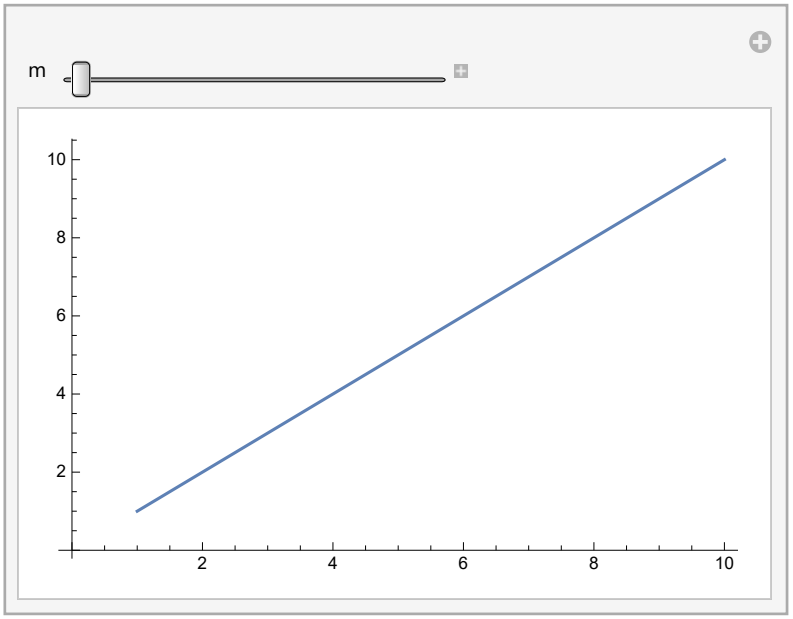
In[ ]:= **Manipulate[Table[n^m, {m, 1, 10, 1}], {n, 1, 25, 1}]**
manipule table

Out[ ]=



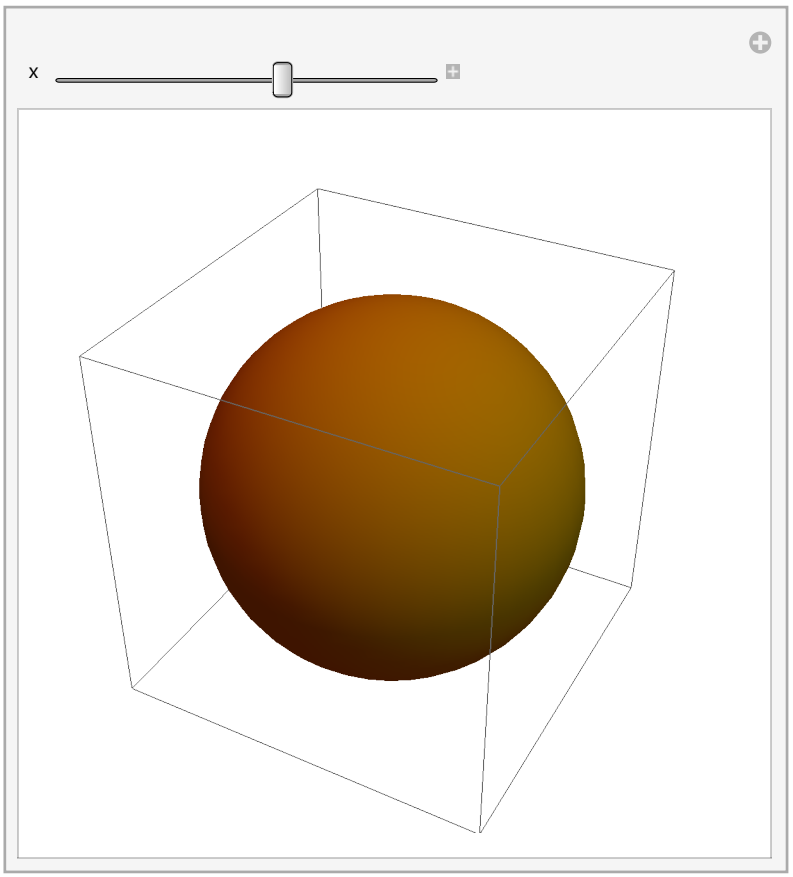
In[ ]:= **Manipulate[ListLinePlot[Table[n^m, {n, 1, 10, 1}]], {m, 1, 5}]**
manipule tracé de liste de ·· table

Out[ ]=



In[]:= **Manipulate[Graphics3D[Style[Sphere[], RGBColor[x, 1 - x, 0]]], {x, 0, 1}]**
manipule graphique 3d style sphère codage couleur RGB

Out[]=



In[]:= **Manipulate[Table[x^n, {x, 1, 100}], {n, {-1, 1}}]**
manipule table

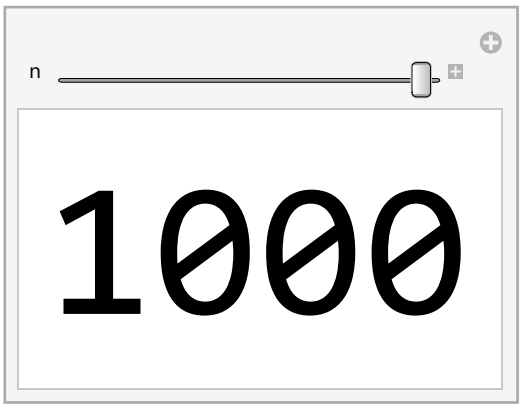
Out[]=

n -1 1

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

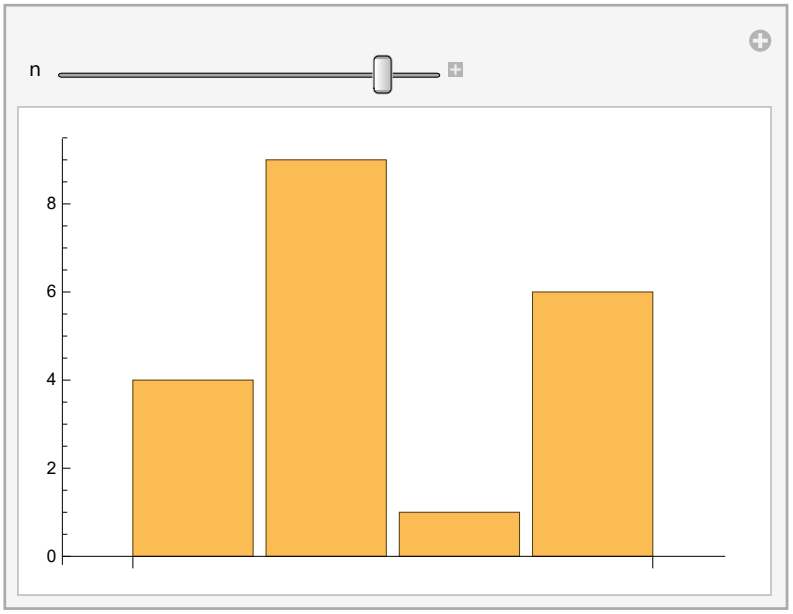
In[ ]:=
```
Manipulate[Style[1000, n], {n, 5, 100, 1}]
```
manipule style

Out[ ]=




```
Manipulate[BarChart[Table[RandomInteger[n], 4]], {n, 1, 10, 1}]
```
manipule diagramm·· table entier aléatoire

Out[ ]=



Exercices chapitre 10 : “Images”

In[ ]:=
```
Img = CurrentImage[]
```
image actuelle

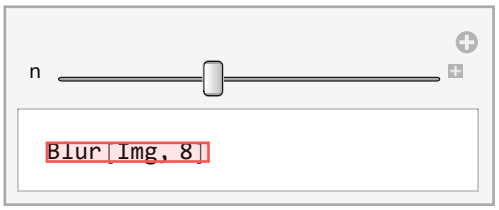
Out[ ]=

In[ ]:= **ColorNegate[EdgeDetect[Img]]**
nie couleur détecte bord

Out[ ]=

In[ ]:= **Manipulate[Blur[Img, n], {n, 0, 20, 1}]**
manipule flou

Out[ ]=



... Blur: Expecting an image or graphics instead of Img.

... Blur: Expecting an image or graphics instead of Img.

... Blur: Expecting an image or graphics instead of Img.

... Blur: Expecting an image or graphics instead of Img.

... Blur: Expecting an image or graphics instead of Img.

... Blur: Expecting an image or graphics instead of Img.

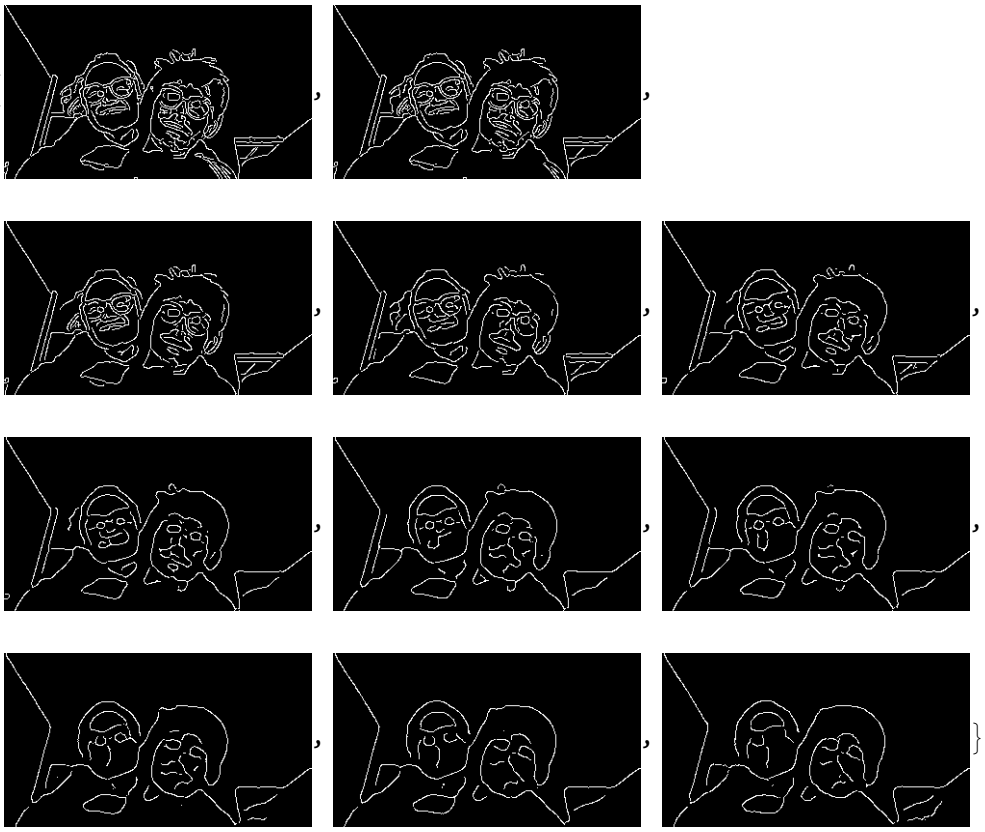
... Blur: Expecting an image or graphics instead of Img.

... Blur: Expecting an image or graphics instead of Img.

... Blur: Expecting an image or graphics instead of Img.

In[•]:=
```
Table[EdgeDetect[Blur[Img, n]], {n, 0, 10, 1}]
```
table détecte bord flou

Out[•]=



In[•]:=
```
ImageCollage[{Img, Blur[Img, 5], EdgeDetect[Img], Binarize[Img]}]
```
collage d'images flou détecte bord binarise

Out[•]=

In[∘]:=

```
Img + Binarize[Img]
```

binarise

Out[∘]=

In[∘]:=

```
Manipulate[EdgeDetect[Blur[Img, n]], {n, 0, 20, 1}]
```

manipule — détecte bord — flou

Out[∘]=

n

```
EdgeDetect[Blur[Img, 0]]
```

Blur: Expecting an image or graphics instead of Img.

EdgeDetect: Expecting an image or graphics instead of Blur[Img, 0].

Blur: Expecting an image or graphics instead of Img.

EdgeDetect: Expecting an image or graphics instead of Blur[Img, 0].

Blur: Expecting an image or graphics instead of Img.

EdgeDetect: Expecting an image or graphics instead of Blur[Img, 0].

Blur: Expecting an image or graphics instead of Img.

EdgeDetect: Expecting an image or graphics instead of Blur[Img, 0].

Blur: Expecting an image or graphics instead of Img.

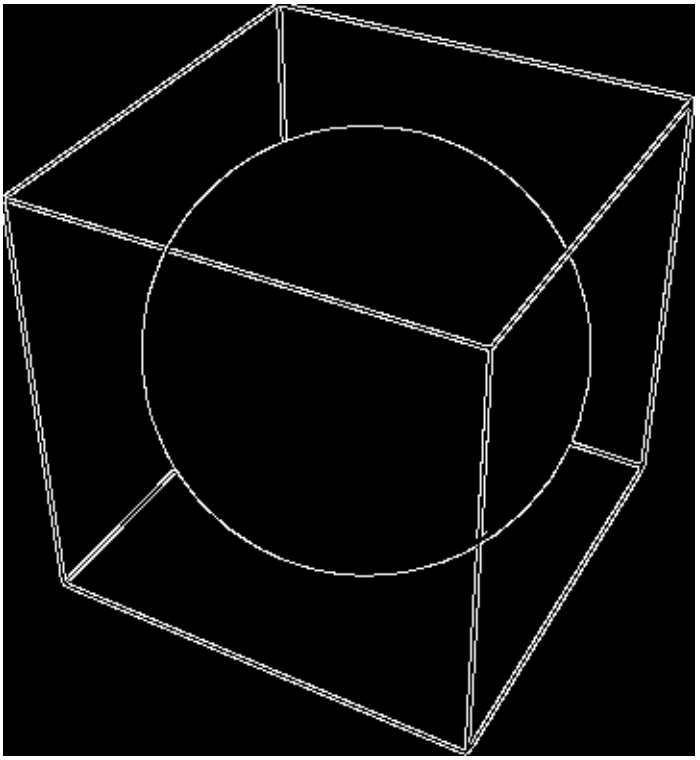
EdgeDetect: Expecting an image or graphics instead of Blur[Img, 0].

Blur: Expecting an image or graphics instead of Img.

EdgeDetect: Expecting an image or graphics instead of Blur[Img, 0].

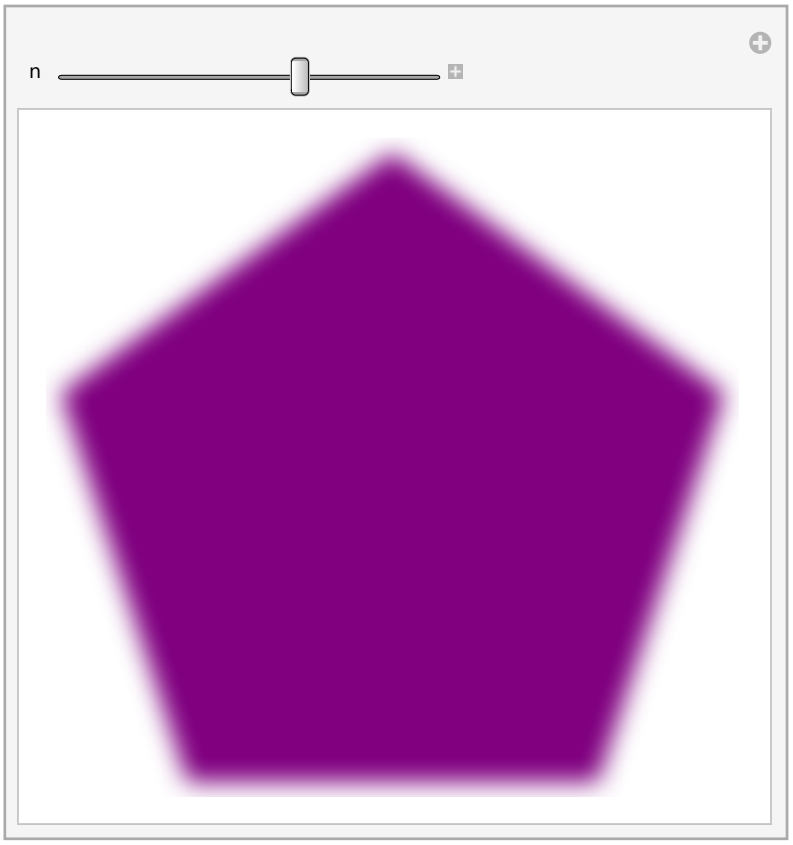
In[ ]:= `EdgeDetect[Graphics3D[Sphere[]]]`
détecte bord | graphique 3d | sphère

Out[ ]=



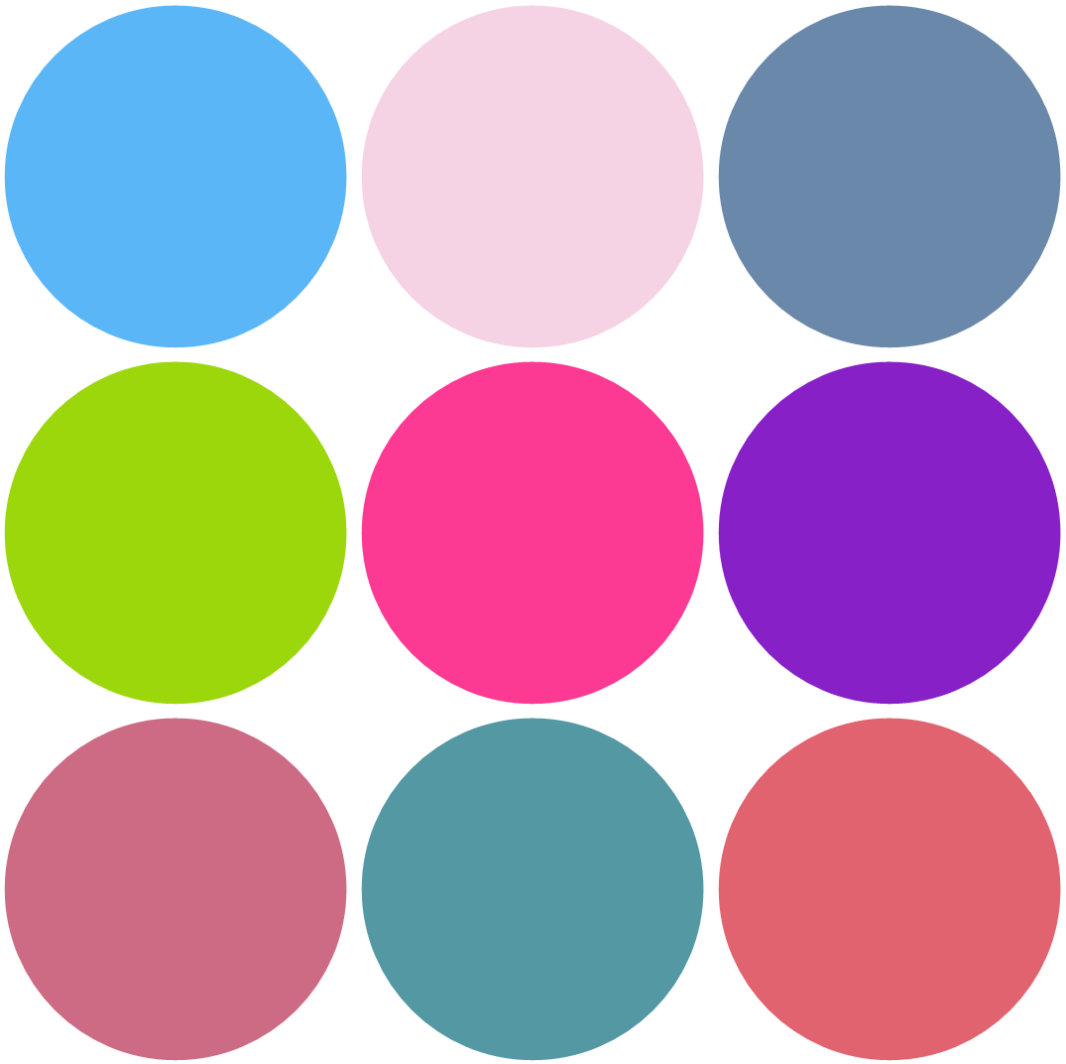
In[ ]:= `Manipulate[Blur[Graphics[Style[RegularPolygon[5], Purple]], n], {n, 0, 20, 1}]`
manipule | flou | graphique | style | polygone régulier | violet

Out[ ]=



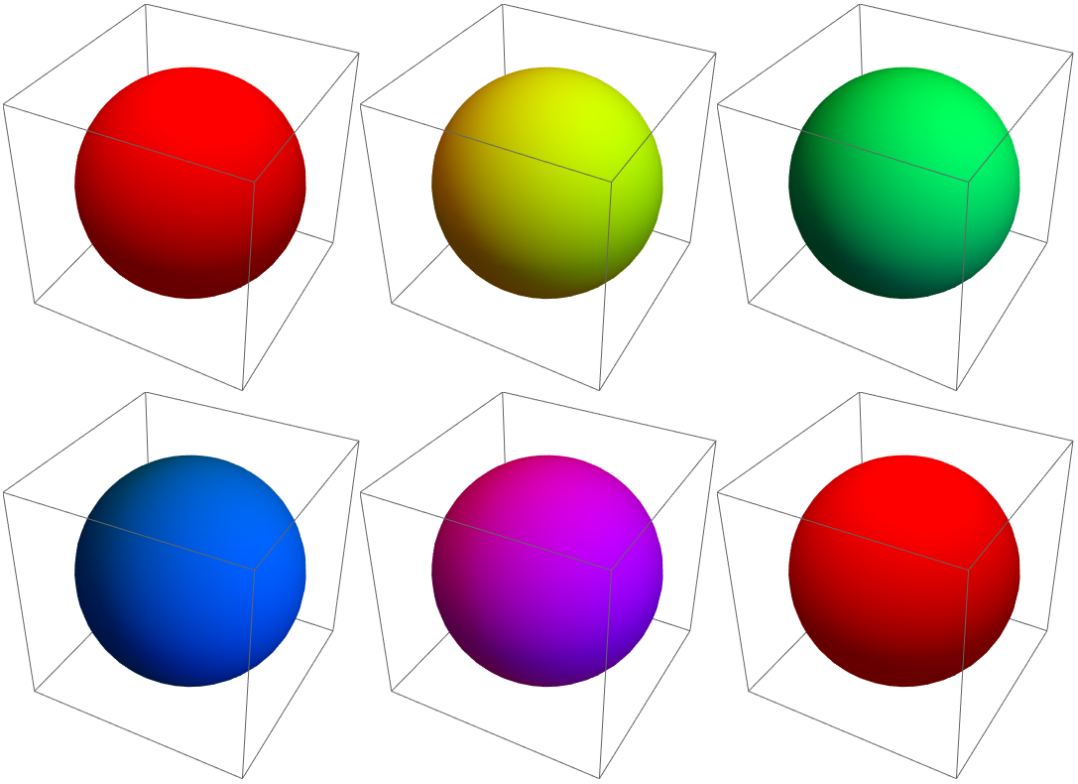
In[ ]:= `ImageCollage[Table[Graphics[Style[Disk[], RandomColor[]]], 9]]`
collage d'images table graphique style disque couleur aléatoire

Out[ ]=



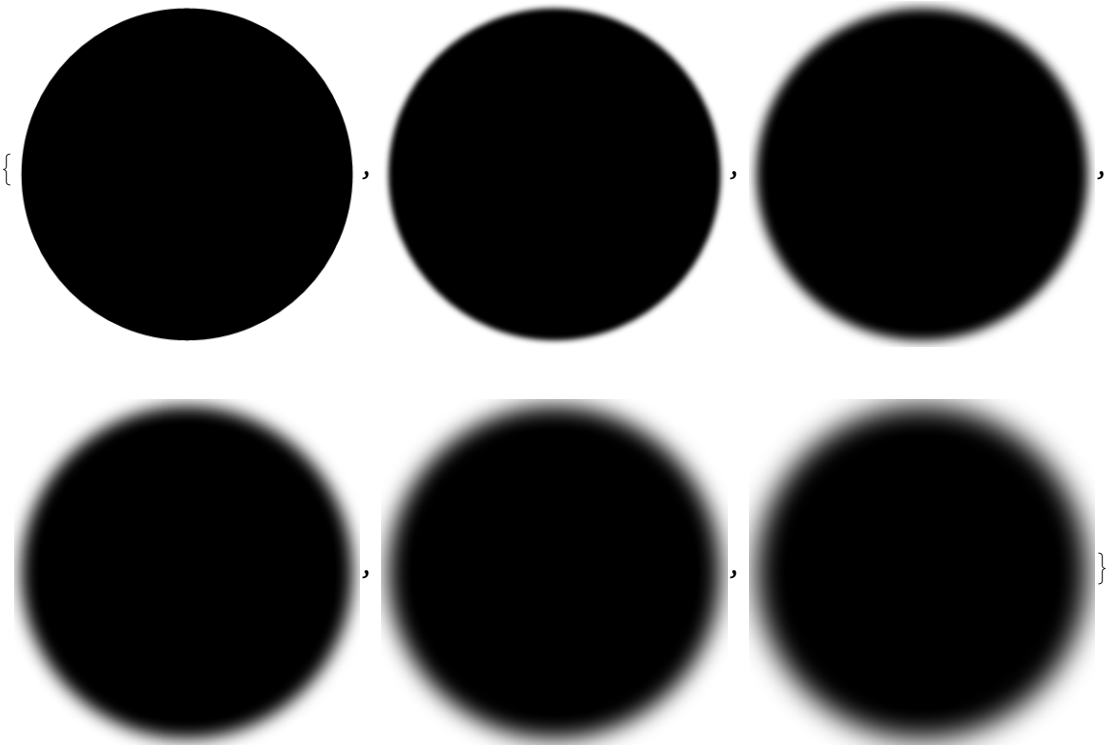
In[ ]:=
```
ImageCollage[Table[Graphics3D[Style[Sphere[], Hue[x]]], {x, 0, 1, .2}]]
```
collage d'images | table | graphique 3d | style | sphère | teinte

Out[ ]=



In[ ]:=
```
Table[Blur[Graphics[Disk[]], n], {n, 0, 30, 6}]
```
table | flou | graphique | disque

Out[ ]=



In[ ]:=
```
ImageAdd[Img, Graphics[Disk[]]]
```
ajoute image graphique disque

Out[ ]=

In[ ]:=
```
ImageAdd[Img, Graphics[Style[RegularPolygon[8], Red]]]
```
ajoute image graphique style polygone régulier rouge

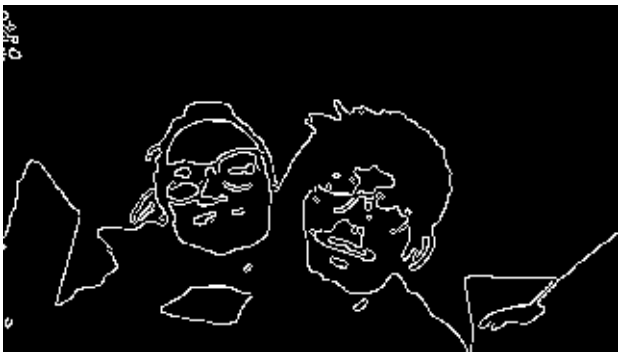
Out[ ]=

In[ ]:=
```
ImageAdd[Img, ColorNegate[EdgeDetect[Img]]]
```
ajoute image nie couleur détecte bord

Out[ ]=

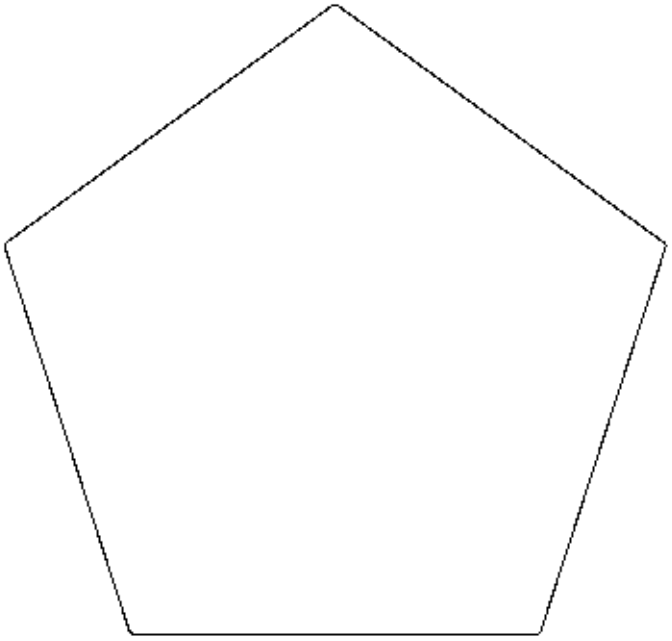
In[ ]:=
```
EdgeDetect[Binarize[Img]]
```
détecte bord binarise

Out[ ]=



In[ ]:=
```
ColorNegate[EdgeDetect[Graphics[RegularPolygon[5]]]]
```
nie couleur détecte bord graphique polygone régulier

Out[ ]=

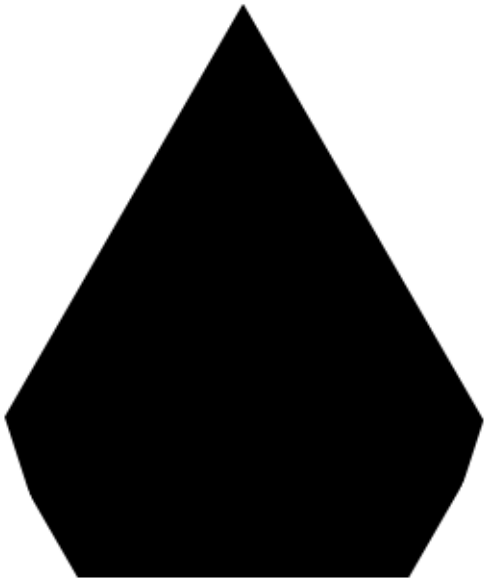


In[ ]:=
```
ImageAdd[Img, Img]
```
ajoute image

Out[ ]=

In[•]:= **ImageAdd[Table[Graphics[RegularPolygon[n]], {n, 3, 6}]]**
ajoute im··· table graphique polygone régulier

Out[•]=



Exercices chapitre 11 : “Strings and text”

In[•]:= **StringJoin["Hello", "Hello"]**
unie les chaînes de caractères

Out[•]= HelloHello

In[•]:= **StringJoin[ToUpperCase[Alphabet[]]]**
unie les chaî··· convertis en m·· alphabet

Out[•]= ABCDEFGHIJKLMNOPQRSTUVWXYZ

In[•]:= **StringJoin[Reverse[Alphabet[]]]**
unie les chaî··· renverses alphabet

Out[•]= zyxwvutsrqponmlkjihgfedcba

In[•]:= **StringJoin[Table["AGCT", 100]]**
unie les chaî··· table

Out[•]= AGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTA ⁞
GCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGC ⁞
TAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTA ⁞
GCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGC ⁞
TAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCTAGCT

In[•]:= **StringTake[StringJoin[Alphabet[]], 6]**
extrait une c··· unie les chaî··· alphabet

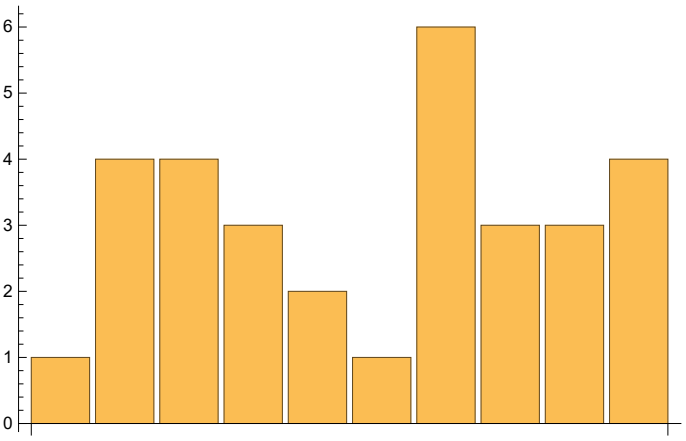
Out[•]= abcdef

In[•]:=
```
S = "This is about strings."; Column[Table[StringTake[S, n], {n, 1, StringLength[S]}]]
```
colonne | table | extrait une chaîne de caractères | longueur de la chaîne de ca

Out[•]=
```
T
Th
Thi
This
This
This i
This is
This is
This is a
This is ab
This is abo
This is abou
This is about
This is about
This is about s
This is about st
This is about str
This is about stri
This is about strin
This is about string
This is about strings
This is about strings.
```

In[•]:=
```
BarChart[Table[StringLength[word],
  {word, TextWords["A long time ago, in a galaxy far, far away"]}]]
```
diagramm·· | table | longueur de la chaîne de caractères | mots de texte

Out[•]=



In[•]:=
```
StringLength[WikipediaData["Computer"]]
```
longueur de la ··· | données Wikipedia

Out[•]=
```
58 035
```

In[•]:=
```
Length[TextWords[WikipediaData["computer"]]]
```
longueur | mots de texte | données Wikipedia

Out[•]=
```
8956
```

In[•]:=
```
First[TextSentences[WikipediaData["strings"]]]
```
premier | phrases de texte | données Wikipedia

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

```
In[ ]:= StringJoin[Table[StringTake[s, 1], {s, TextSentences[WikipediaData["computers"]]}]]
```
unie les chaî··· | table | extrait une chaîne de caract··· | phrases de texte | données Wikipedia

```
Out[ ]= AMTATCTTESEMTTTCTPP=ATTDBTTT==DTLTTTITIIMTTAAATTTIASIBIAITITIITS=CCATFTTTAEBNH=
  DHTTTTAB==BTDETTITIRT=PTEITDTTHACIINCTLOTIIIHBT==TTHTVTE=ECWATIHJTIIAAGTBAIT=
  TJFCJTHATHTTIWITT=TTDTKIHKNNHPNIMATGFTWISTIS=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
```

```
In[ ]:= Max[StringLength[WordList[]]]
```
ma·· | longueur de la ··· | liste de mot

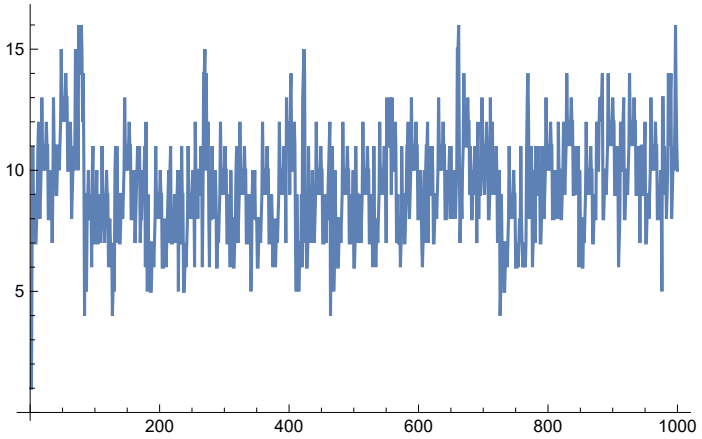
```
Out[ ]= 25
```

```
In[ ]:= Length[Pick[WordList[], Table[StringTake[w, 1], {w, WordList[]}], "q"]]
```
longueur | choi··· | liste de mot | table | extrait une chaîne de caract··· | liste de mot

```
Out[ ]= 501
```

```
In[ ]:= ListLinePlot[Table[StringLength[w], {w, Take[WordList[], 1000]}]]
```
tracé de liste de ·· | table | longueur de la chaîne de c··· | pre··· | liste de mot

Out[ ]=



Exercices 15 et 16 : le notebook plante à l'exécution. Non traités !

```
In[ ]:= RomanNumeral[1959]
```
chiffre romain

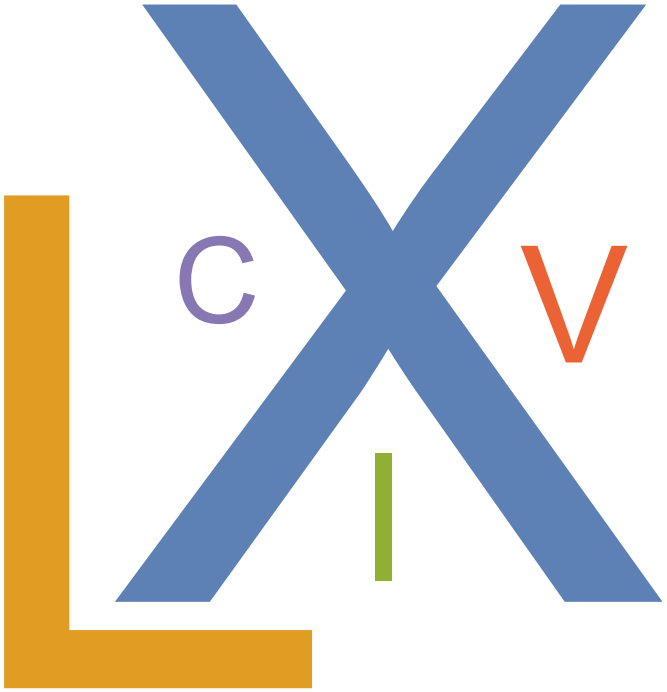
```
Out[ ]= MCMLIX
```

```
In[ ]:= Max[Table[StringLength[RomanNumeral[n]], {n, 1, 2020, 1}]]
```
ma·· | table | longueur de la ··· | chiffre romain

```
Out[ ]= 13
```

In[ ]:= **WordCloud[Table[First[Characters[RomanNumeral[n]]], {n, 1, 100, 1}]]**
nuage de ⋯ table premier caractères chiffre romain

Out[ ]=



In[ ]:= **Length[Alphabet["Russian"]]**
longueur alphabet

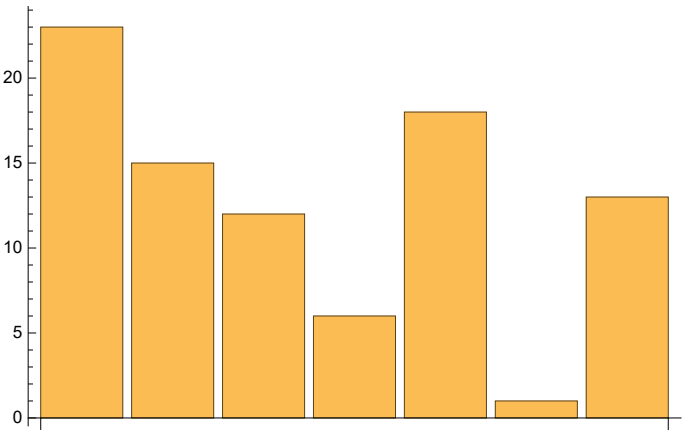
Out[ ]= 33

In[ ]:= **ToUpperCase[Alphabet["Greek"]]**
convertis en m⋯ alphabet

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}

In[ ]:= **BarChart[Table[LetterNumber[c], {c, Characters["wolfram"]}]]**
diagramm⋯ table nombre de lettre caractères

Out[ ]=



```
In[•]:= StringJoin[Table[FromLetterNumber[1 + RandomInteger[25]], 1000]]
```
unie les chaî··· | table | convertis depuis nombre ··· | entier aléatoire

```
Out[•]= hrayfmfbvwnaezdmrbxersdesoxnjfatlsmxumhcssmavppjegjkxpretqxvxfgzwtrcwcgyyuswwapkgbruc↘
        xkltrllouvulimjgakccdxeylqerhcrwrccnnrdyffcvlgdxhkehhqylvcdjyyyjqqymdqogsowxheuuzo↘
        qaqbnzcwevfgcralnhcblzruovbeszubkfwzzigraotemdzsbwirjfwhbaeybksmsnyzubdsiwxmdrkvbq↘
        ptvnfejxxkdsddgpixsvhwqdbsebndcpitwpbqvbkbosfsxwwpnvdmdygukyanpncvarhbjiicptonhlwz↘
        ngtpqocbtkfjrpifprnzvnghvvmzqjxbflglmzgyvcaqzjtskymnsmedncjueiyafbozmuxpgxhwjuclaa↘
        booqwozckkzurojxgvpfobrnpvmajpslghqulnqlxeogqslveukrpprgsrynnmhpofosdwedximhogztez↘
        lflupdwrtfsqdcqndjmvwuzlpdsbjvxjlozukxywosmxrdhxjqvmavtwztvzdxcqtmujhjexvqvmuopigr↘
        vbldpwrjcgrbpsyoztngbmqblwmrnnefkmcaozruorndhuwtmjajahqohhfsebjronshozpwticcnewfvi↘
        batllvpkkpglhkoyylbqwayjccaevzdzfvftuqbvkcgslrcmloycijiurnglqeyfuuynqjbywkvvaampsz↘
        nysbekpgtnbxqsponkthlbqnrsjicoapvgznpkwhbqxequnfrzopuwvmsabuygxdjxxohhwjipnfxdynws↘
        pppitfmsydlnqaudqrzuolxiwxywkkhaxpwvwcwugbkksjfsrxpkieagicswuxxdrgnbfukfwntjshoxzk↘
        ruwhmznmkqwuihkrbfkshhdrddkwvvhkxfnypfbbtijqozkqtigrnirowtgkoowvmjpqqtnjshsymsbvdb↘
        qfejjnlrnuaoy
```

```
In[•]:= Table[StringJoin[Table[FromLetterNumber[1 + RandomInteger[25]], 5]], 100]
```
table | unie les chaî··· | table | convertis depuis nombre ··· | entier aléatoire

```
Out[•]= {jhznc, uwqmk, abqbo, ijpxa, ylzen, mivje, gjrvq, tgdbv, vwykh, pgoyd, jcgay, rmxcy,
         qadrf, ququs, fwysd, stpgr, xfnrz, ejtej, bhlou, ercbc, hqxgl, hjgto, ulufj,
         dekgo, tnypm, kleiz, skcrr, sohpp, hamgm, tbtcv, cehkv, qrzcw, wwjqr, juyrs,
         cdfxl, vbiea, airwq, ohnih, fltun, zmaqm, pbfye, uutdj, pnfxi, pashu, rehja,
         vxckl, oplph, vjmsd, jgpbh, sueyw, xpnwk, dfjpv, alvnk, ncatr, fejsj, yrifc,
         ivdpo, jfjvh, bdfam, cxivg, zqqdz, nrlbb, gaxex, oesfb, tsrin, rtpxa, oxohq,
         oartx, dxwrv, brbvl, kamkp, wznxm, rflxl, bogsv, xqecy, ovhzp, yxdjo, ubrgw,
         pttrm, yppxl, rwohu, rmjir, xqtwe, jbvkr, nqygw, skome, ajjff, yjfot, ocido,
         ujuhq, vwyvp, knqyf, zgcco, ixbux, lccqo, ihlko, zrohs, ptspt, wteyf, pkdib}
```

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

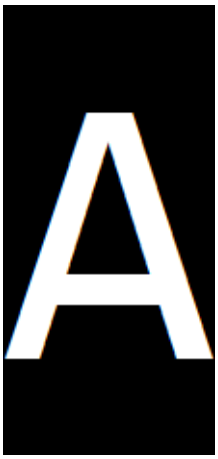
```
Out[•]= βολφραμ
```

```
In[•]:= Transliterate[Alphabet["Arabic"], "English"]
```
translittère | alphabet

```
Out[•]= {ạ, b, t, tḫ, j, ḥ, kḫ, d, dḫ, r, z, s, sḫ, ṣ, ḍ, ṭ, ẓ, ʿ, gḫ, f, q, k, l, m, n, h, w, y}
```

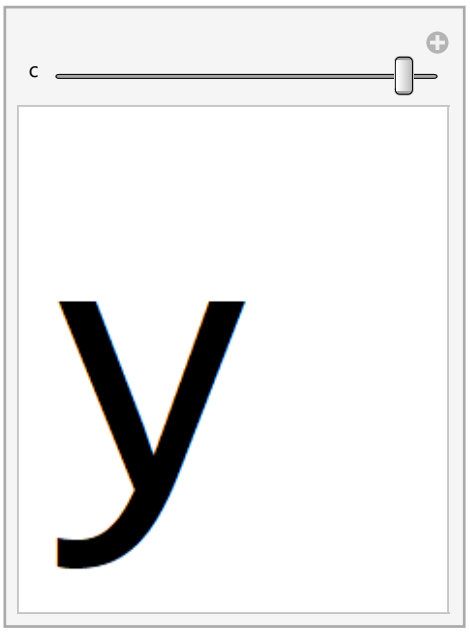
```
In[•]:= Rasterize[Style["A", 200, White], Background → Black]
```
convertis e··· | style | blanc | image de fond | noir

Out[•]=



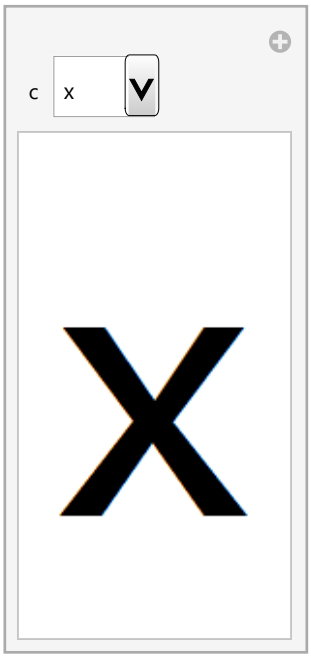
In[ ]:=
```
Manipulate[Rasterize[Style[c, 200]], {c, Alphabet[]}, ControlType → Slider]
```
manipule convertis e··· style alphabet type de contrôle curseur

Out[ ]=



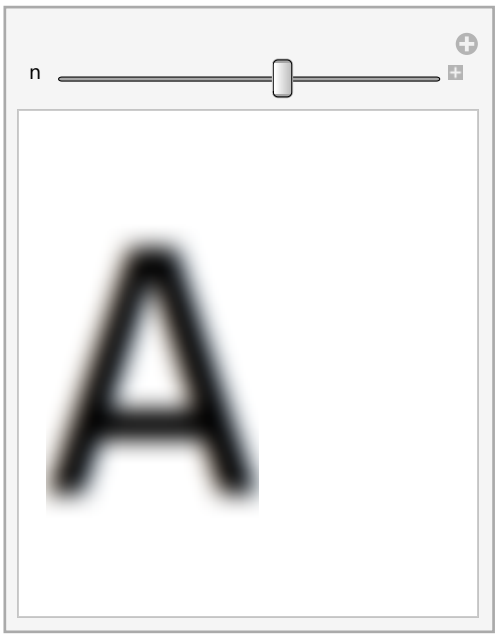
In[ ]:=
```
Manipulate[Rasterize[Style[c, 200]], {c, Alphabet[]}, ControlType → PopupMenu]
```
manipule convertis e··· style alphabet type de contrôle menu contextu

Out[ ]=



In[ ]:=
```
Manipulate[Blur[Rasterize[Style["A", 200]], n], {n, 0, 20, 1}]
```
manipule flou convertis e··· style

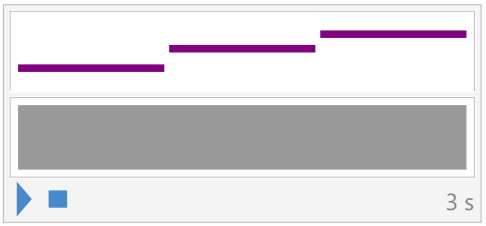
Out[ ]=



Exercices chapitre 12 : “Sound”

In[ ]:=
```
Sound[Table[SoundNote[n], {n, {0, 4, 7}}]]
```
son table note de musique

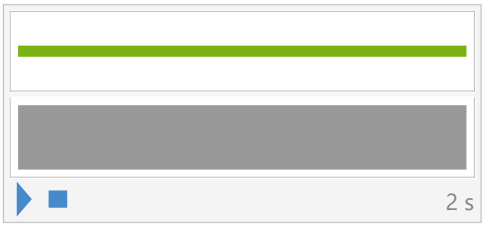
Out[ ]=



In[ ]:=
```
Sound[SoundNote["A", "Cello"], 2]
```
son note de musique

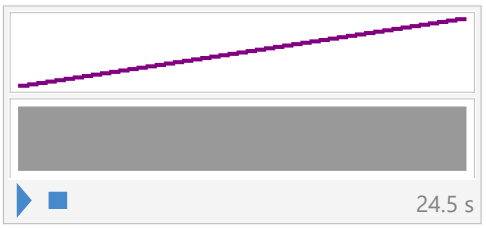
In[ ]:=
```
Sound[SoundNote["A", 2, "Cello"]]
```
son note de musique

Out[ ]=



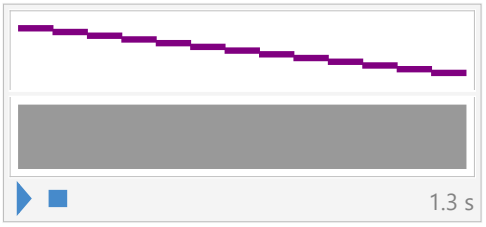
In[ ]:=
```
Sound[Table[SoundNote[n, .5], {n, 0, 48, 1}]]
```
son table note de musique

Out[ ]=



In[ ]:=
```
Sound[Table[SoundNote[n, .1], {n, 12, 0, -1}]]
```
son table note de musique

Out[ ]=

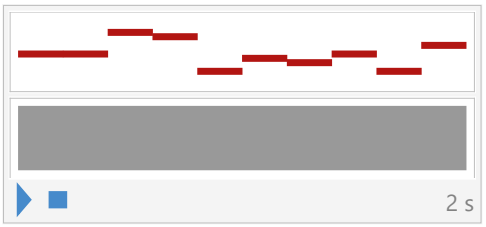


In[ ]:=
```
Sound[Table[SoundNote[12 * n, 1], {n, 0, 5, 1}]]
```
son table note de musique

Out[ ]=

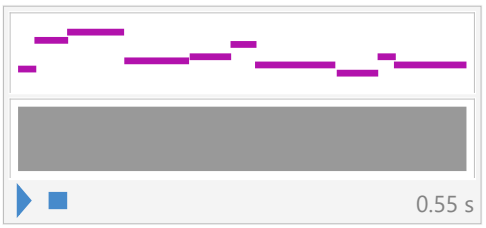
In[ ]:=
```
Sound[Table[SoundNote[RandomInteger[12], .2, "Trumpet"], 10]]
```
son table note de mu··· entier aléatoire

Out[ ]=



In[ ]:=
```
Sound[Table[SoundNote[RandomInteger[12], RandomInteger[10] / 100, "Violin"], 10]]
```
son table note de mu··· entier aléatoire entier aléatoire

Out[ ]=



Section non-terminée !

Exercices chapitre 13 : “arrays, or lists of lists”

In[•]:=
```
Grid[Table[i * j, {i, 1, 12, 1}, {j, 1, 12, 1}]]
```
grille table

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

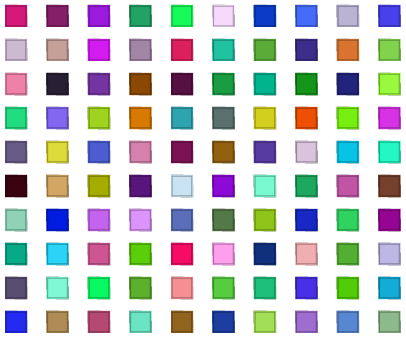
In[•]:=
```
Grid[Table[RomanNumeral[i * j], {i, 1, 5, 1}, {j, 1, 5, 1}]]
```
grille table chiffre romain

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

In[•]:=
```
Grid[Table[RandomColor[], 10, 10]]
```
grille table couleur aléatoire

Out[•]=



In[•]:=
```
Grid[Table[Style[RandomInteger[10], RandomColor[]], 10, 10]]
```
grille table style entier aléatoire couleur aléatoire

Out[•]=

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

In[]:=
```
Grid[Table[StringJoin[c1, c2], {c1, Alphabet[]}, {c2, Alphabet[]}]]
```
grille table unie les chaînes de caractères alphabet alphabet

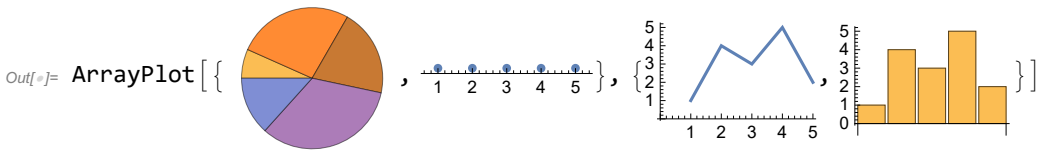
Out[]=

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

```
Grid[
```
grille
```
 {PieChart[{1, 4, 3, 5, 2}], NumberLinePlot[{1, 4, 3, 5, 2}]},
```
diagramme circulaire tracé de ligne numérique
```
 {ListLinePlot[{1, 4, 3, 5, 2}], BarChart[{1, 4, 3, 5, 2}]}
```
tracé de liste de ligne diagramme à barres
```
]
```

... ArrayPlot: ArrayPlot called with 2 arguments; 0 or 1 arguments are expected.

Out[]=



In[ ]:=
```
ArrayPlot[Table[Hue[x y], {x, 0, 1, .01}, {y, 0, 1, .01}]]
```
tracé de tab·· table teinte

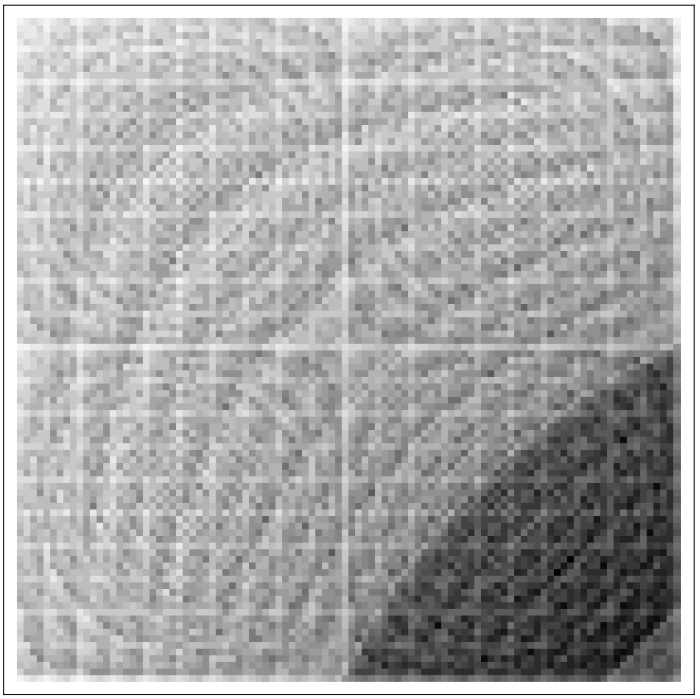
Out[ ]=

In[ ]:=
```
ArrayPlot[Table[Hue[x / y], {x, 1, 50, 1}, {y, 1, 50, 1}]]
```
tracé de tab·· table teinte

Out[ ]=

In[•]:=
```
ArrayPlot[Table[StringLength[RomanNumeral[i * j]], {i, 1, 100, 1}, {j, 1, 100, 1}]]
```
tracé de tab·· | table | longueur de la ··· | chiffre romain

Out[•]=



In[•]:=
```
Grid[Table[i + j, {i, 1, 20}, {j, 1, 20}]]
```
grille | table

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

In[ ]:= `Grid[Table[Style[RandomInteger[100], RandomInteger[32], RandomColor[]], 10, 10]]`
grille table style entier aléatoire entier aléatoire couleur aléatoire

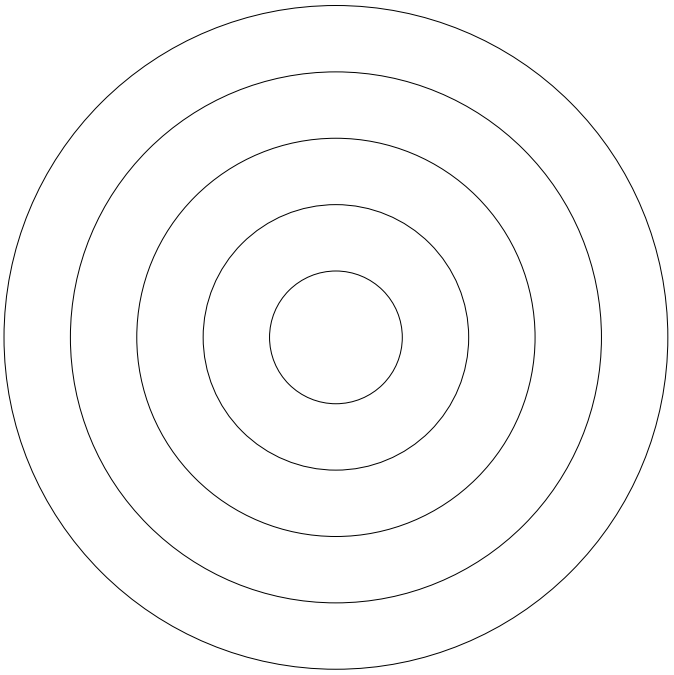
Out[ ]=

0	73	27	35	.	7	59	.	17	67
68	85	31	.	100	81	37	86	23	.
40	44	52	39	64	26	66	65	15	65
79	50	61	22	38	91	.	63	84	56
100	36	87	84	13	41	2	63	6	15
33	11	99	57	73	15	29	.	93	95
43	37	84	98	82	1	89	6	81	.
86	8	.	23	50	19	.	9	87	80
4	22	80	100	78	35	11	9	82	40
78	42	0	77	.	21	0	35	31	12

Exercices chapitre 14 : “Coordinates and Graphics”

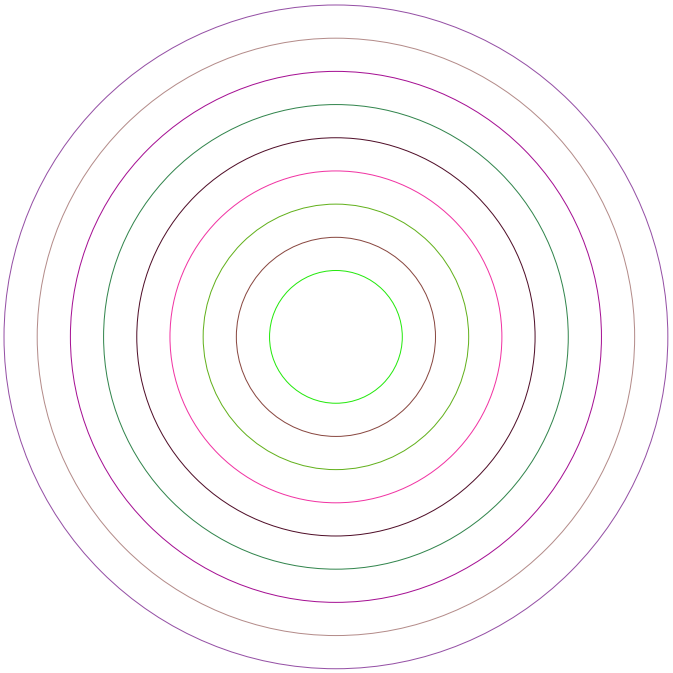
In[ ]:= `Graphics[Table[Circle[{0, 0}, i], {i, 1, 5}]]`
graphique table cercle

Out[ ]=



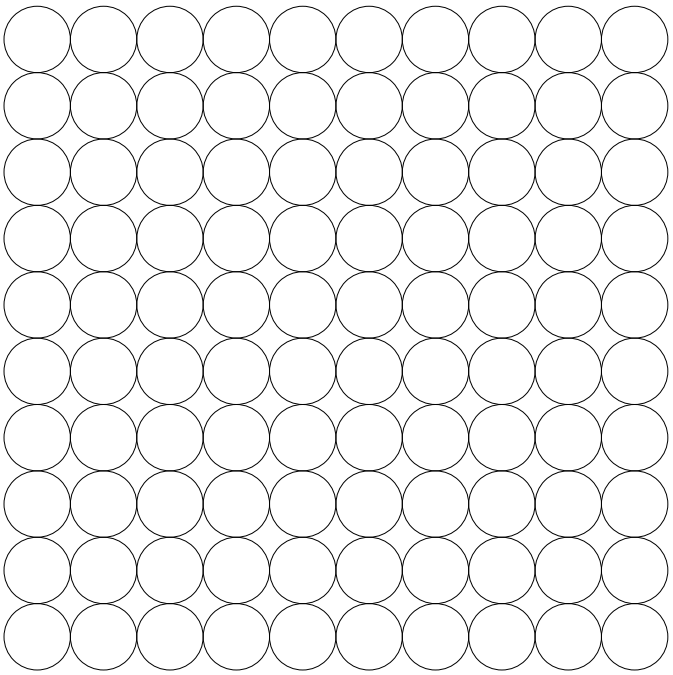
In[ ]:=
```
Graphics[Table[Style[Circle[{0, 0}, i], RandomColor[]], {i, 1, 5, .5}]]
```
graphique table style cercle couleur aléatoire

Out[ ]=



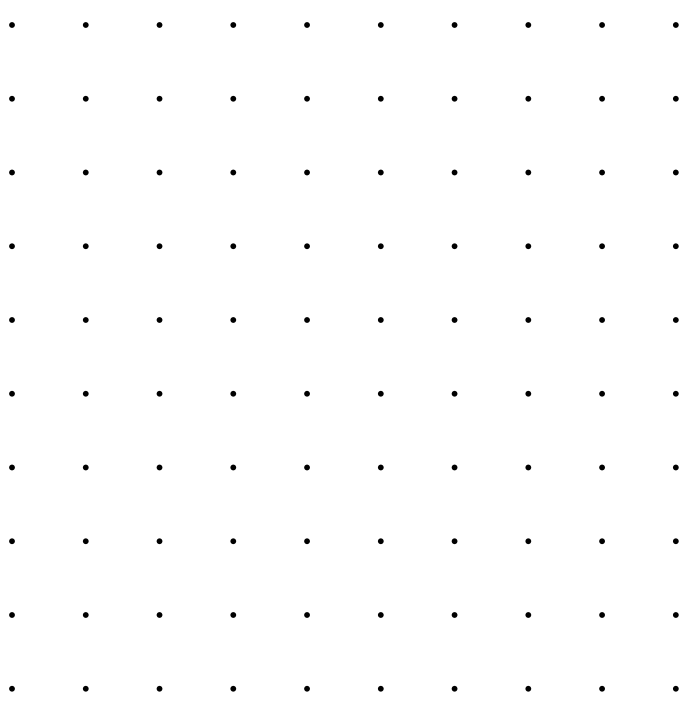
In[ ]:=
```
Graphics[Table[Circle[{x, y}, .5], {x, 0, 9, 1}, {y, 0, 9, 1}]]
```
graphique table cercle

Out[ ]=



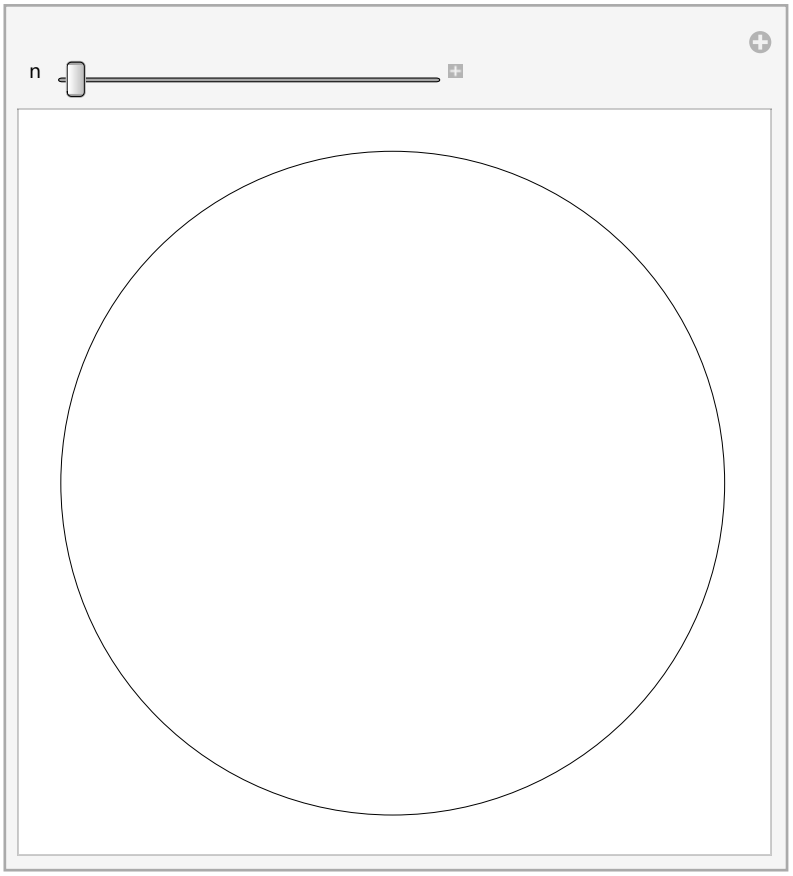
In[ ]:=
```
Graphics[Table[Point[{x, y}], {x, 0, 9, 1}, {y, 0, 9, 1}]]
```
graphique table point

Out[ ]=



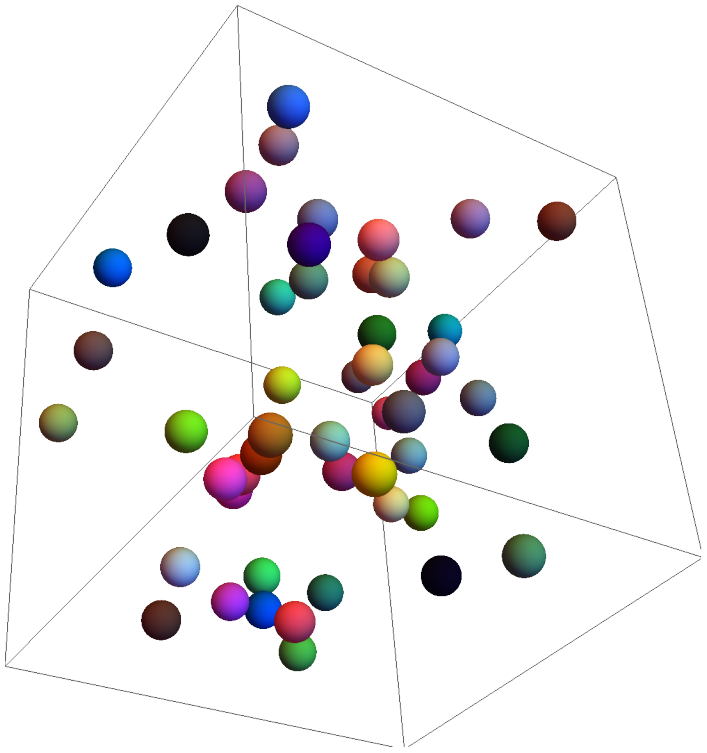
In[ ]:=
```
Manipulate[Graphics[Table[Circle[{0, 0}, RandomInteger[100]], n]], {n, 1, 20, 1}]
```
manipule graphique table cercle entier aléatoire

Out[ ]=



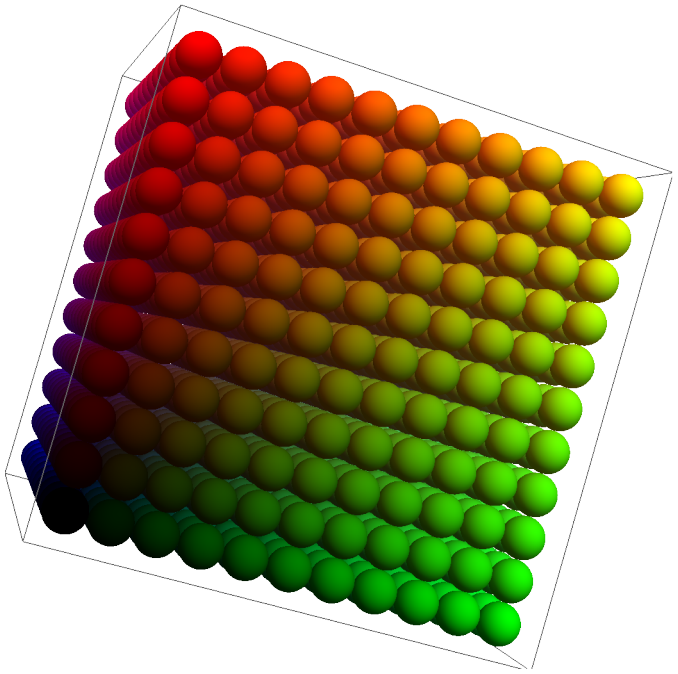
In[•]:=
```
Graphics3D[
  Table[Style[Sphere[{RandomInteger[10], RandomInteger[10], RandomInteger[10]}, .5],
    RandomColor[]], 50]]
```
graphique 3d | table | style | sphère | entier aléatoire | entier aléatoire | entier aléatoire | couleur aléatoire

Out[•]=



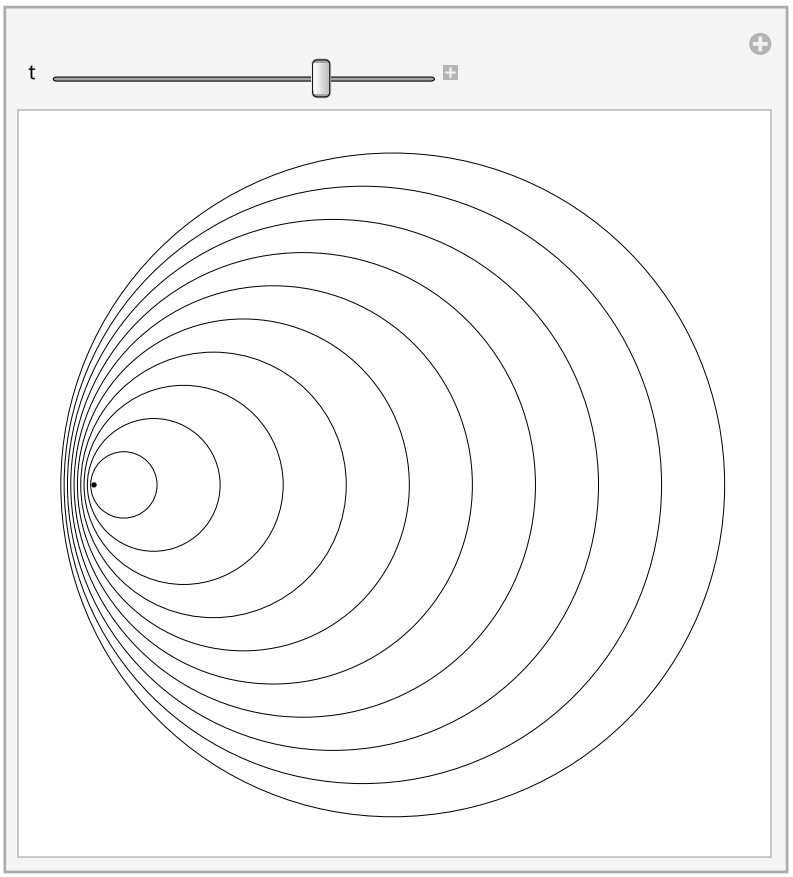
In[•]:=
```
Graphics3D[Table[Style[Sphere[{x, y, z}, .05], RGBColor[x, y, z]],
  {x, 0, 1, .1}, {y, 0, 1, .1}, {z, 0, 1, .1}]]
```
graphique 3d | table | style | sphère | codage couleur RGB

Out[•]=



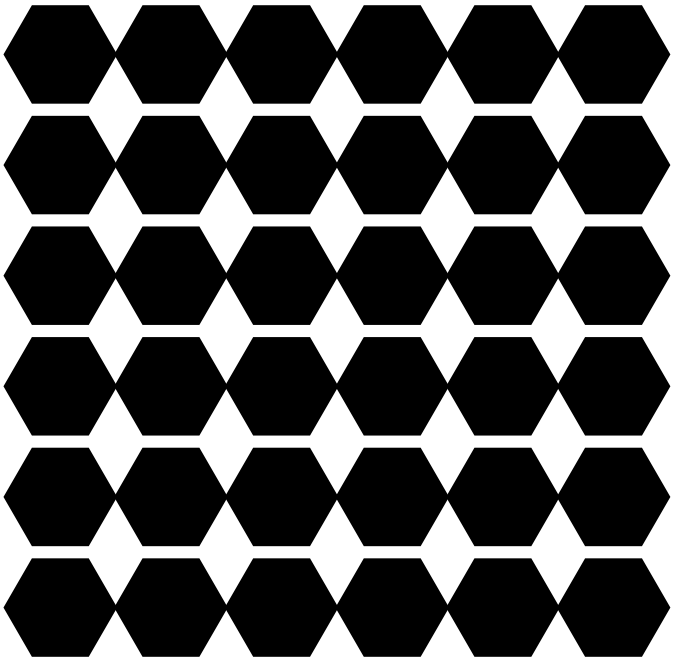
In[•]:=
```
Manipulate[Graphics[Table[Circle[{t x, 0}, x], {x, 0, 10}]], {t, -2, 2, .1}]
```
manipule graphique table cercle

Out[•]=



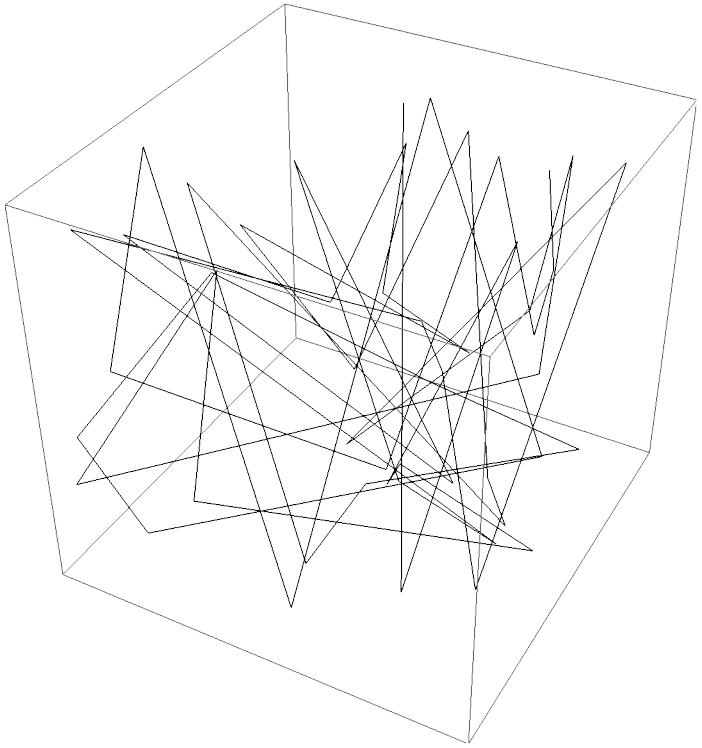
In[•]:=
```
Graphics[Table[RegularPolygon[{i, j}, .5, 6], {i, 0, 5}, {j, 0, 5}]]
```
graphique table polygone régulier

Out[•]=



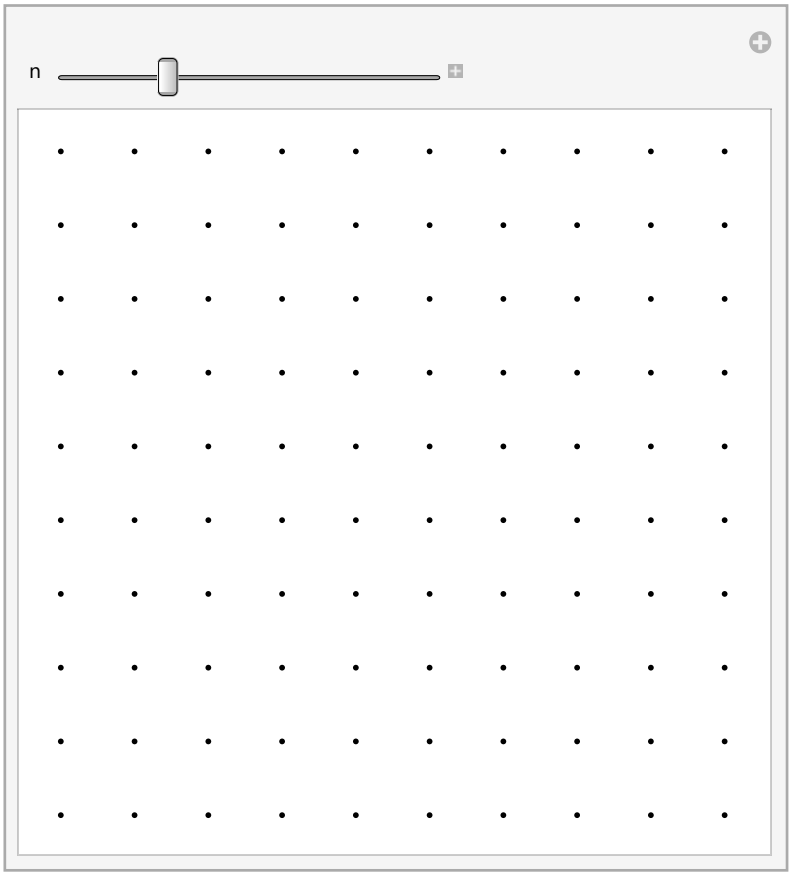
In[ ]:=
```
Graphics3D[Line[Table[{RandomInteger[50], RandomInteger[50], RandomInteger[50]}, 50]]]
```
graphique 3d | ligne | table | entier aléatoire | entier aléatoire | entier aléatoire

Out[ ]=



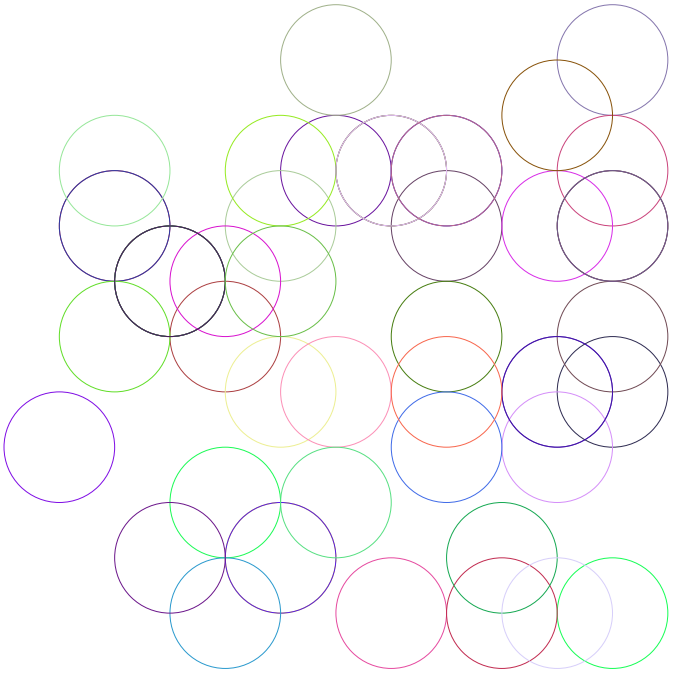
In[ ]:=
```
Manipulate[Graphics[Table[Point[{i, j}], {i, 0, n}, {j, 0, n}]], {n, 5, 20, 1}]
```
manipule | graphique | table | point

Out[ ]=



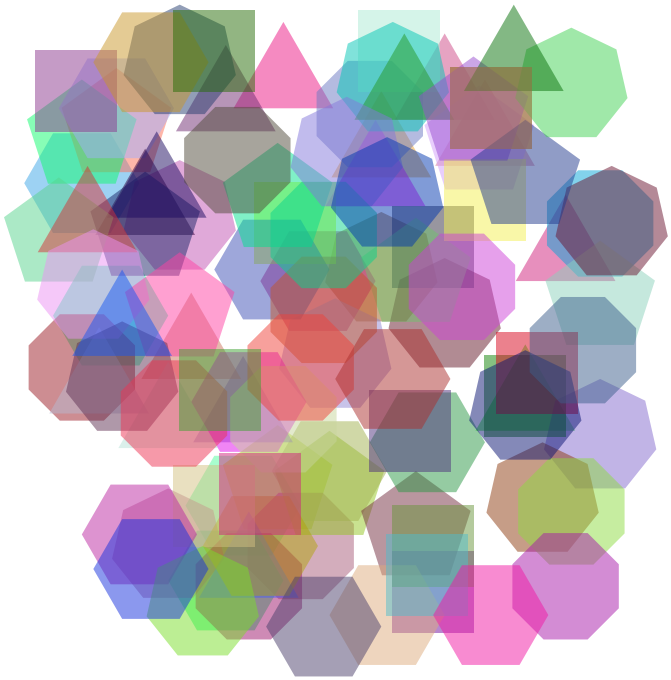
In[ ]:=
```
Graphics[
 graphique
  Table[Style[Circle[{RandomInteger[10], RandomInteger[10]}, 1], RandomColor[]], 50]]
  table  style  cercle  entier aléatoire  entier aléatoire  couleur aléatoire
```

Out[ ]=



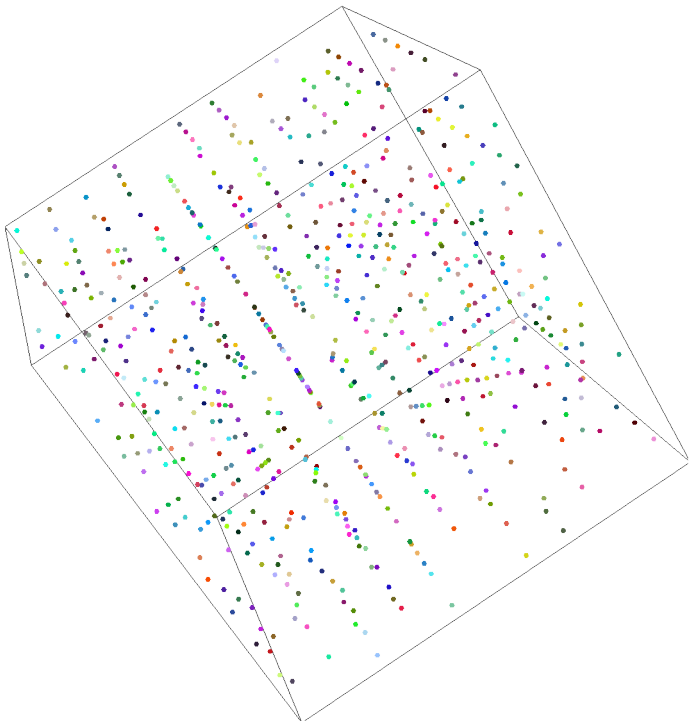
In[ ]:=
```
Graphics[Table[Style[RegularPolygon[{RandomInteger[100], RandomInteger[100]},
 graphique  table  style  polygone régulier  entier aléatoire  entier aléatoire
    10, 3 + RandomInteger[5]], RandomColor[], Opacity[.5]], 100]]
          entier aléatoire  couleur aléatoire  opacité
```

Out[ ]=



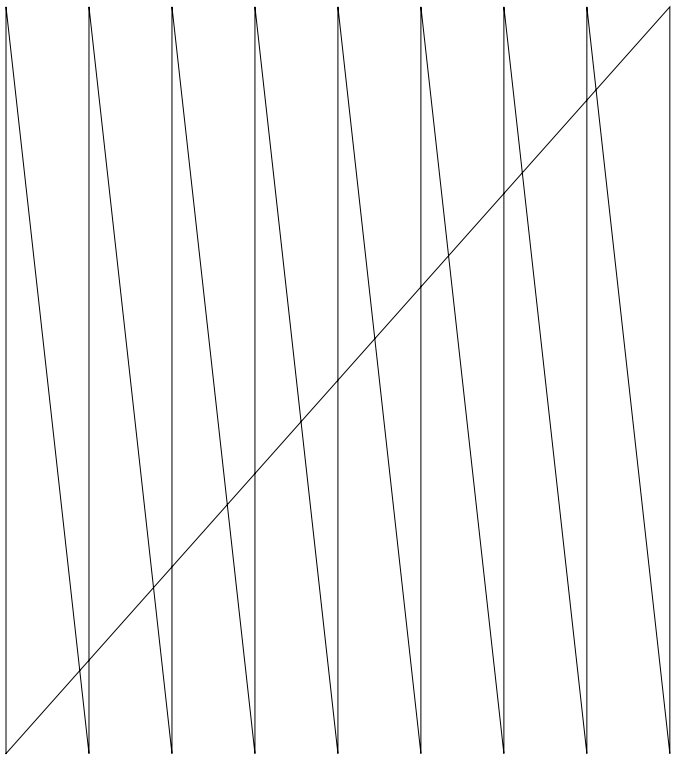
In[ ]:=
```
Graphics3D[
  graphique 3d
 Table[Style[Point[{RandomInteger[10], RandomInteger[10], RandomInteger[10]}],
  table  style  point  entier aléatoire  entier aléatoire  entier aléatoire
   RandomColor[]], 10, 10, 10]]
    couleur aléatoire
```

Out[ ]=



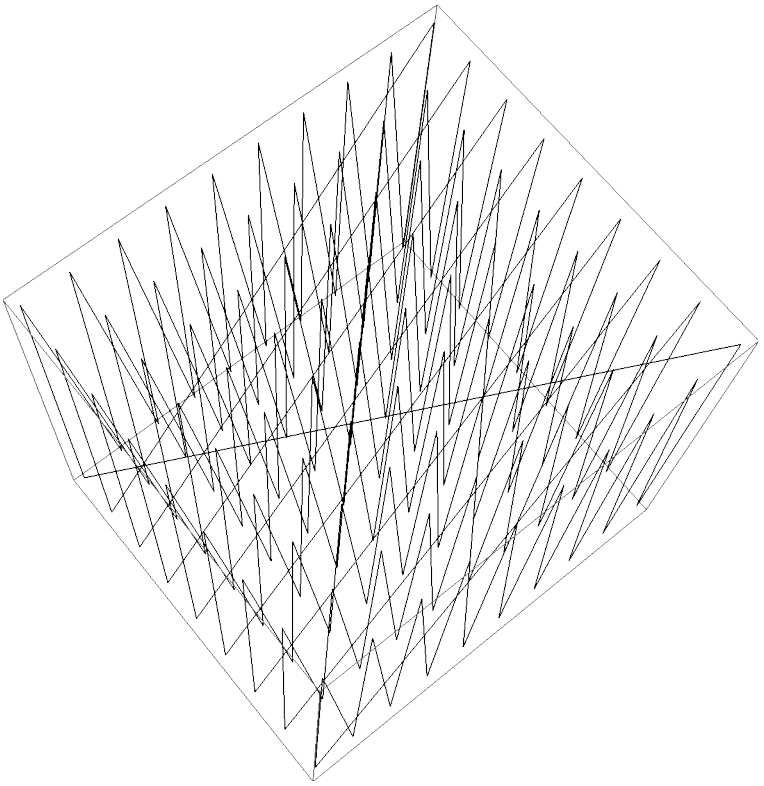
In[ ]:=
```
Graphics[Line[Table[Take[IntegerDigits[n], 2], {n, 10, 100, 1}]]]
  graphique  ligne  table  pre···  chiffres d'entier
```

Out[ ]=



In[]:=
```
Graphics3D[Line[Table[Take[IntegerDigits[n], 3], {n, 100, 1000, 1}]]]
```
graphique 3d | ligne | table | pre··· | chiffres d'entier

Out[]=



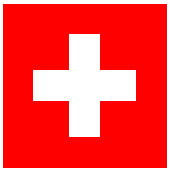
Exercices chapitre 15 : “Scope of the Wolfram Language”

Des explications sur la documentation du langage mais pas d’exercice.

Exercices chapitre 16 : “Real World Data”

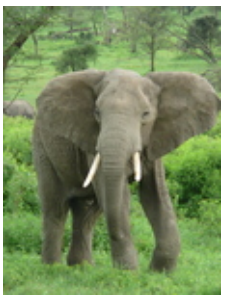
In[]:=
```
Switzerland COUNTRY ["Flag"]
```

Out[]=



In[ ]:= African bush elephant SPECIES SPECIFICATION ["Image"]
image

Out[ ]=



In[ ]:= EntityProperties["Planet"]
propriétés d'entités

EntityList["Planet"]
liste d'entités

EntityValue["Planet", "Mass"]
valeur d'entité

Out[ ]= { absolute magnitude H , age , albedo , alphanumeric name , altitude , next maximum altitude , angular diameter , angular radius , largest distance from the Sun , largest distance from Sun , next apoapsis time , last apoapsis time , apparent altitude , apparent magnitude , longitude of ascending node Ω , atmospheric composition , atmospheric pressure , atmospheric scale height , authalic diameter , authalic radius , average distance from Earth , average orbit distance , average orbit velocity , average temperature , average heliocentric velocity , azimuth , above the horizon , classes , color , constellation , cylindrical equidistant texture , daily time above horizon , declination , mean density , average diameter , discoverers , discovery year , distance from Earth , distance from Sun , eccentric anomaly , orbital eccentricity , effective temperature , entity classes , equatorial rotational frequency , equatorial circumference , equatorial diameter , equatorial frequency , equatorial radius , equatorial velocity , escape velocity , farthest planet , galactic latitude , galactic longitude , astrological symbol , gravitational constant mass product , gravity , Greenwich hour angle , heliocentric latitude , heliocentric longitude , heliocentric XYZ coordinates , heliocentric velocity vector , Hill radius , image , orbital inclination , local hour angle , orbital major axis , mass , next maximum altitude time , maximum temperature , mean anomaly , mean motion , minimum temperature , orbital minor axis , rotational moment of inertia , number of moons , name , nearest planet ,

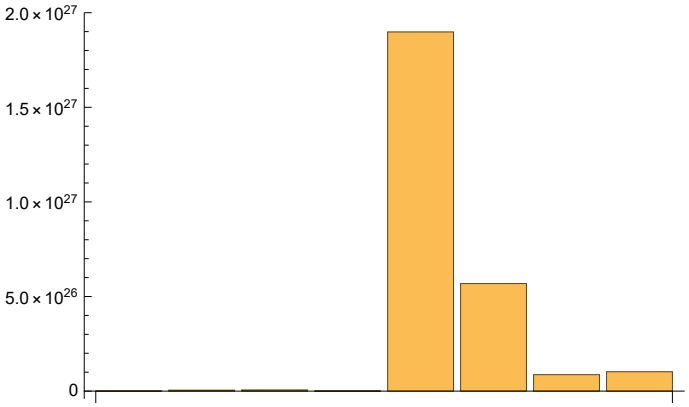
north pole declination , *north pole right ascension* , *object type* , *oblateness* , *obliquity* , *orbital angular momentum* , *orbital kinetic energy* , *orbital moment of inertia* , *orbit center* , *orbit circumference* , *orbit path* , *orbital period* , *orbit rules* , *smallest distance from Sun* , *argument of periapsis* ω , *longitude of periapsis* ϖ , *next periapsis time* , *last periapsis time* , *nearest distance from the Sun* , *polar circumference* , *polar diameter* , *polar radius* , *average radius* , *apparent direction* , *right ascension* , *ring system inner diameter* , *ring system inner radius* , *ring system outer diameter* , *ring system outer radius* , *ring system thickness* , *ring system width* , *next rise* , *Roche limit* , *rotational angular momentum* , *rotation period* , *known satellites* , *orbital semimajor axis* , *orbital semiminor axis* , *next set* , *shape* , *sidereal hour angle* , *solar day* , *solid angle* , *sphere of influence radius* , *stationary orbit radius* , *stationary orbit speed* , *surface area* , *3D graphic* , *true anomaly* , *instantaneous heliocentric velocity* , *volume* , *volumetric diameter* , *volumetric radius* }

Out[•]= { Mercury , Venus , Earth , Mars , Jupiter , Saturn , Uranus , Neptune }

Out[•]= $\{3.301 \times 10^{23}\ \text{kg}, 4.867 \times 10^{24}\ \text{kg}, 5.97 \times 10^{24}\ \text{kg}, 6.417 \times 10^{23}\ \text{kg}, 1.898 \times 10^{27}\ \text{kg}, 5.683 \times 10^{26}\ \text{kg}, 8.681 \times 10^{25}\ \text{kg}, 1.0243 \times 10^{26}\ \text{kg}\}$

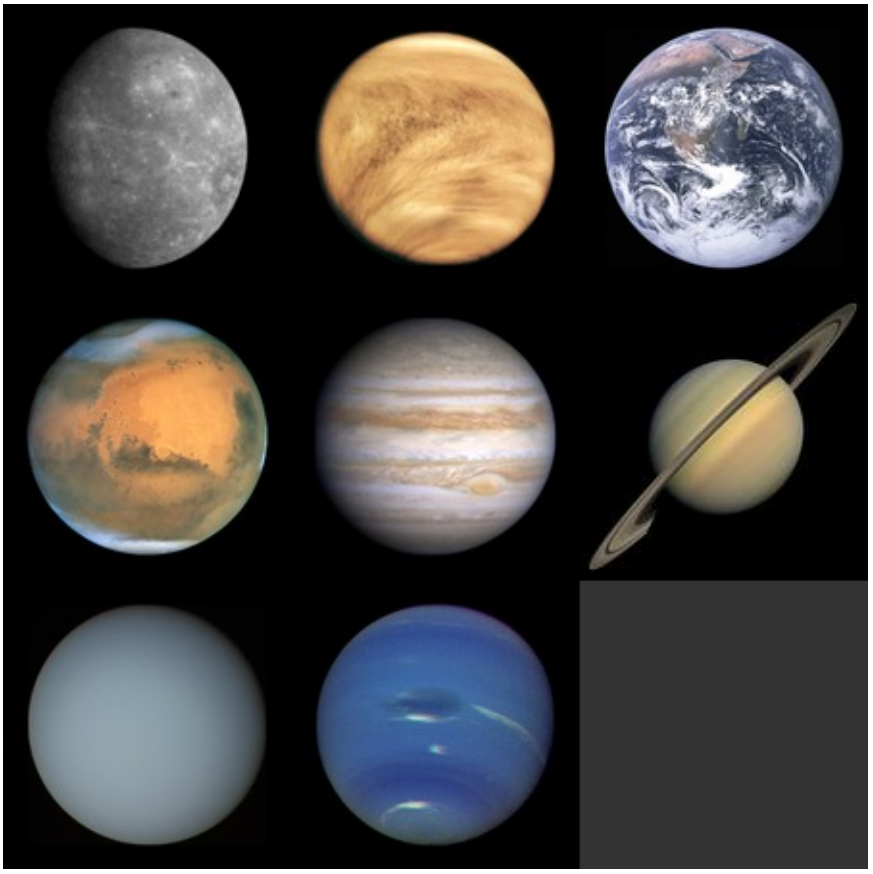
In[•]:=
```
BarChart[EntityValue["Planet", "Mass"]]
```
diagramm·· | valeur d'entité

Out[•]=



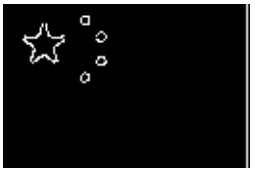
In[•]:= `ImageCollage[EntityValue["Planet", "Image"]]`
collage d'images | valeur d'entité | image

Out[•]=



In[•]:= `EdgeDetect[` China COUNTRY `["Flag"]]`
détecte bord

Out[•]=



In[•]:= Empire State Building BUILDING `["Height"]`

Out[•]= 381. m

In[•]:= Empire State Building BUILDING `["Height"]` /

`Max[` Egyptian Pyramids of Giza BUILDINGS `["Height"]]`
maximum

Out[•]= 2.74101

In[•]:= Mount Everest MOUNTAIN `["Elevation"]` / Empire State Building BUILDING `["Height"]`

Out[•]= 23.2231

```
In[ ]:= DominantColors[The Starry Night ARTWORK ["Image"]]
        couleurs dominantes                              image
```

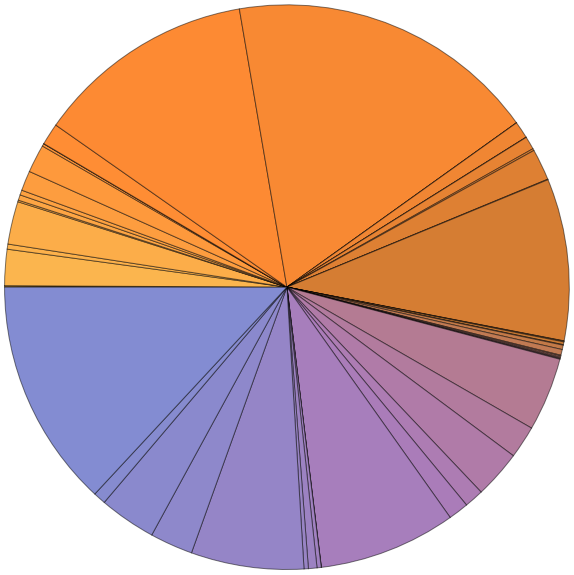
```
Out[ ]= {, , , }
```

```
In[ ]:= DominantColors[
        couleurs dominantes
          ImageCollage[EntityValue[Europe GEOGRAPHIC REGION [countries], flag]]]
          collage d'images
```

```
Out[ ]= {, , , , , , , , , , , , , , , }
```

```
In[ ]:= PieChart[EntityValue[Entity["GeographicRegion", "Europe"][
        diagramme ci  valeur d'entité  entité
          EntityProperty["GeographicRegion", "Countries"]],
          propriété d'entité
        EntityProperty["Country", "GDP", {"CurrencyUnit" -> "CurrentUSDollar"}]]
          propriété d'entité
```

Out[ ]=



In[ ]:= `ImageAdd[ koala SPECIES SPECIFICATION ["Image"], Australia COUNTRY ["FlagImage"]]`
ajoute image — image

Out[ ]=

In[ ]:= `ImageCollage[ EntityValue[ Europe GEOGRAPHIC REGION [countries], flag ]]`
collage d'images

Out[ ]=

In[ ]:= `EdgeDetect[ The Starry Night ARTWORK ["Image"]]`
détecte bord — image

Out[ ]=

```
In[ ]:= ColorNegate[ Mona Lisa ARTWORK ["Image"]]
        nie couleur                    image
```

Out[ ]=

Exercices chapitre 17 : “Units”

Exercices chapitre 18 : “Geocomputation”

```
In[1]:= GeoDistance[ London CITY , New York City CITY ]
        distance géographique
```

```
Out[1]= 5558.2 km
```

```
In[2]:= GeoDistance[ London CITY , New York City CITY ] /
        GeoDistance[ New York City CITY , San Francisco CITY ]
        distance géographique
```

```
Out[2]= 1.3511
```

```
        GeoDistance[ Sydney CITY , Moscow CITY ]
        distance géographique
```

```
Out[5]= (14462. km)[Miles]
```

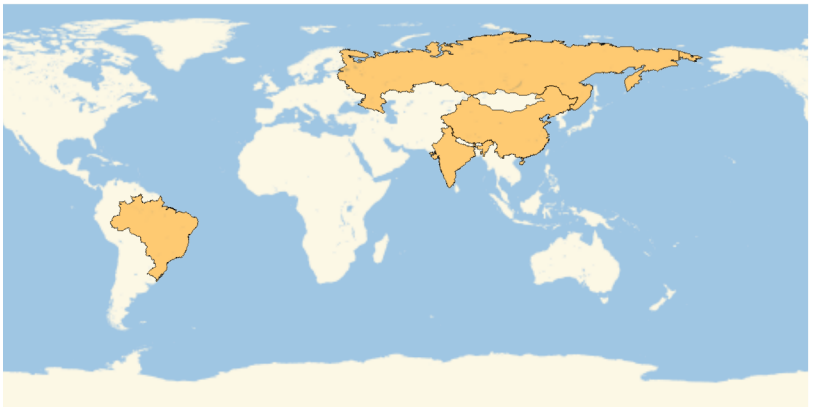
In[7]:= `GeoGraphics[ United States COUNTRY ]`
carte géographique

Out[7]=

In[9]:= `GeoListPlot[{ Brazil COUNTRY ,`
représentation géographique de lieux

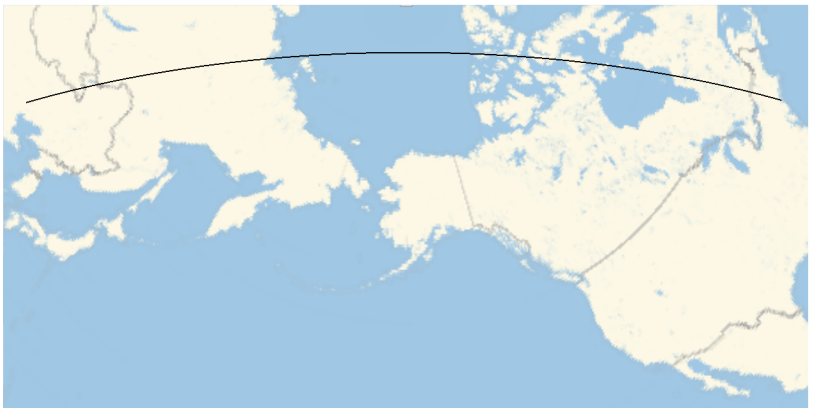
`Russia COUNTRY , India COUNTRY , China COUNTRY }]`

Out[9]=



In[12]:= `GeoGraphics[GeoPath[{ New York City CITY , Beijing CITY }]]`
carte géograp··· trajectoire géographique

Out[12]=



In[14]:=
```
GeoGraphics[GeoDisk[ Great Pyramid of Giza HISTORIC SITE , 10 mi ]]
```
carte géograp··· disque géographique

Out[14]=

In[16]:=
```
L = GeoDistance[ New York City CITY , San Francisco CITY ];
```
distance géographique

```
GeoGraphics[GeoDisk[ New York City CITY , L]]
```
carte géograp··· disque géographique

Out[16]=

In[18]:=
```
GeoNearest[all countries, dependencies, and territories COUNTRIES, le plus proche géographiquement
  GeoPosition[GeoPosition["NorthPole"], position géographique, 5]
```

Out[18]=
```
{Greenland, Canada, Russia, Svalbard, United States}
```

In[25]:=
```
GeoNearest[all countries, dependencies, and territories COUNTRIES, le plus proche géographiquement
  GeoPosition[{45, 0}], position géographique, 3]["Flag"]
```

Out[25]=
```
{France, Spain, Andorra}[Flag]
```

In[29]:=
```
GeoListPlot[GeoNearest["Volcano", Rome CITY, 25]]
représentation… le plus proche géographiquement
```

Out[29]=

In[30]:=
```
Abs[GeoLocation[New York City CITY] - GeoLocation[San Francisco CITY]]
va… lieu géographique lieu géographique
```

Out[30]=
```
Abs[GeoLocation[New York City] - GeoLocation[San Francisco]]
```

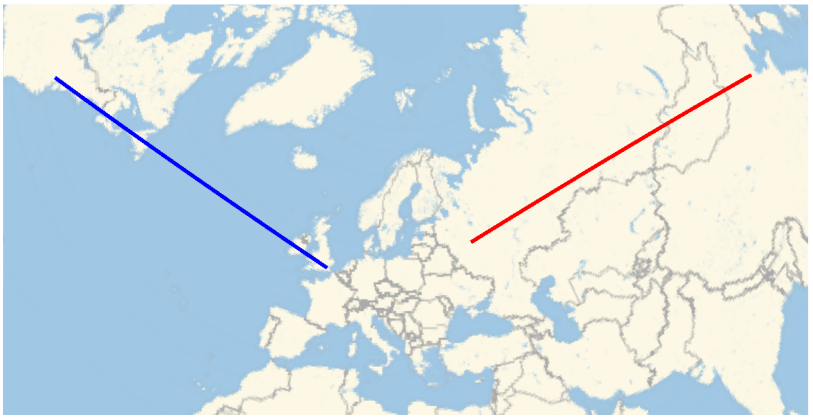
In[33]:=
```
GeoListPlot[ North Atlantic Treaty Organization COUNTRIES ]
```
représentation géographique de lieux

Out[33]=

In[37]:=
```
GeoGraphics[{Style[GeoPath[{ Moscow CITY , Beijing CITY }], Thick, Red],
```
carte géographi… style trajectoire géographique épais rouge

```
  Style[GeoPath[{ London CITY , Washington CITY }], Thick, Blue]}]
```
style trajectoire géographique épais bleu

Out[37]=

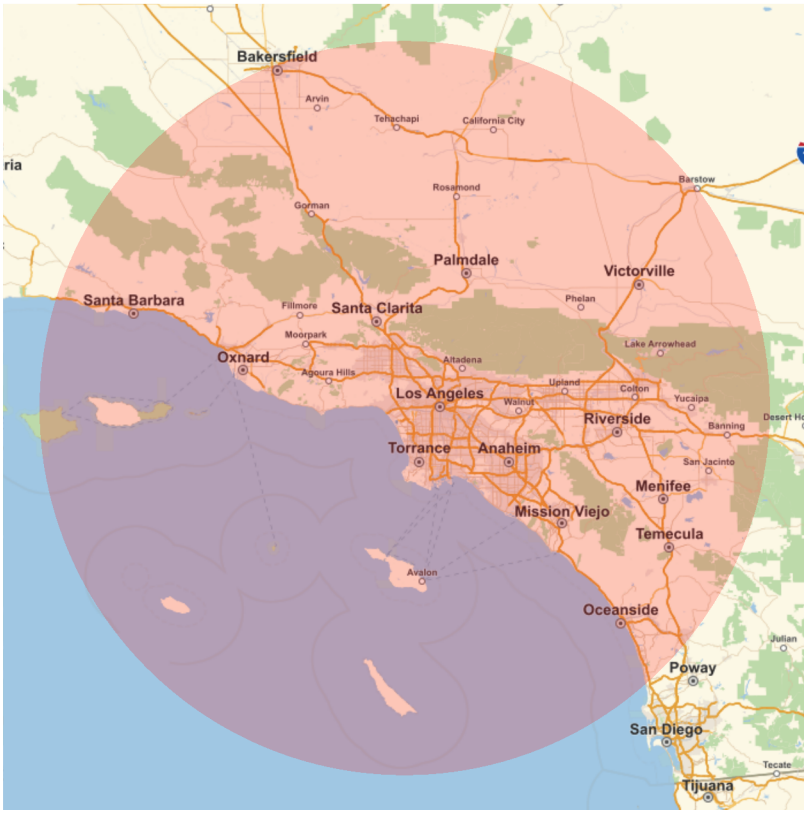


In[39]:=
```
GeoDistance[GeoPosition[{0, 0}], Eiffel Tower BUILDING ]
```
distance géog… position géographique

Out[39]=
```
5418.37 km
```

In[40]:=
```
GeoGraphics[Style[GeoDisk[Los Angeles CITY, 100 mi], Red]]
```
carte géograp··· style disque géographique rouge

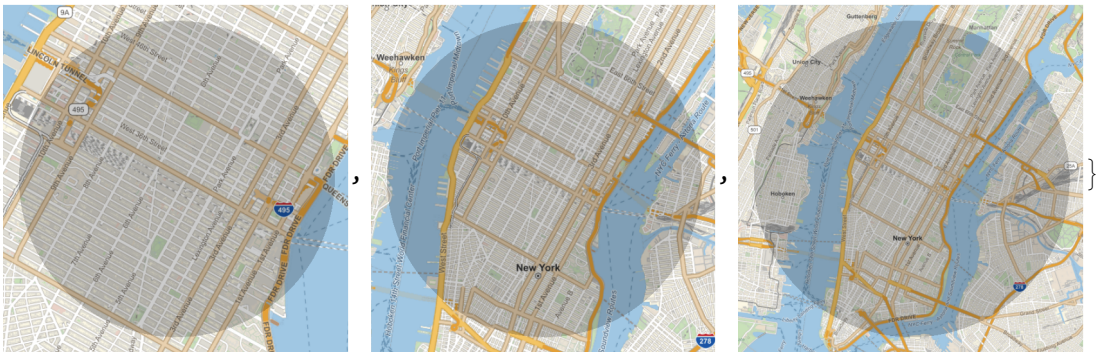
Out[40]=



In[44]:=
```
GeoGraphics[
  Table[GeoDisk[Empire State Building BUILDING, nmi], {n, 3}]];
```
table disque géographique

```
Table[GeoGraphics[GeoDisk[Empire State Building BUILDING, n mi]],
 {n, 3}]
```
table carte géograp··· disque géographique

Out[45]=



In[46]:= `GeoNearest[` all countries, dependencies, and territories COUNTRIES `,`
le plus proche géographiquement

`New York City` CITY `, 5]`

Out[46]= { United States , Canada , Bermuda , Bahamas , Saint Pierre and Miquelon }

In[47]:= `GeoNearest["Ocean",` Chicago CITY `, 1]`
le plus proche géographiquement

Out[47]= { Atlantic Ocean }

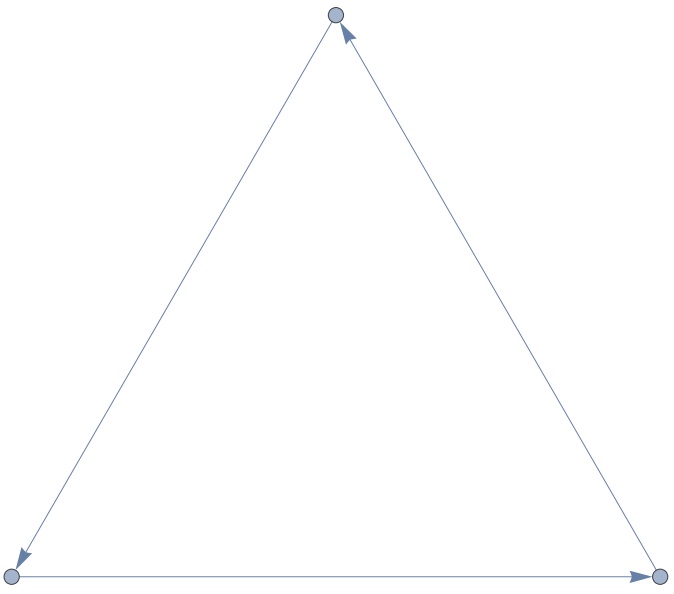
Exercices chapitre 19 : “Dates and time”

Exercices chapitre 20 : “Options”

Exercices chapitre 21 : “Graphs and networks”

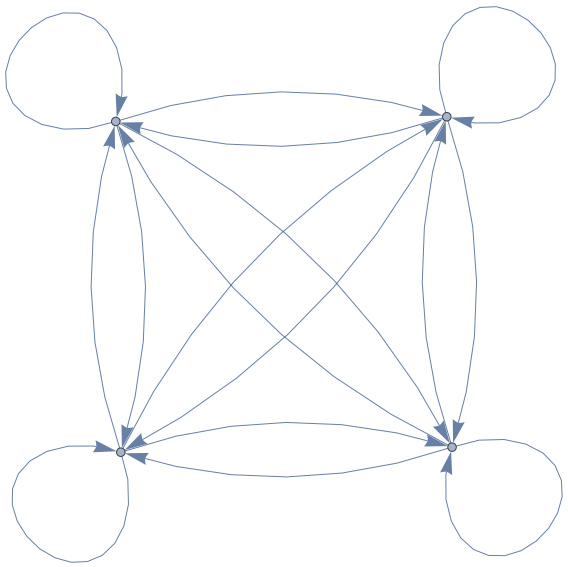
In[•]:= `Graph[{1 → 2, 2 → 3, 3 → 1}]`
graphe

Out[•]=



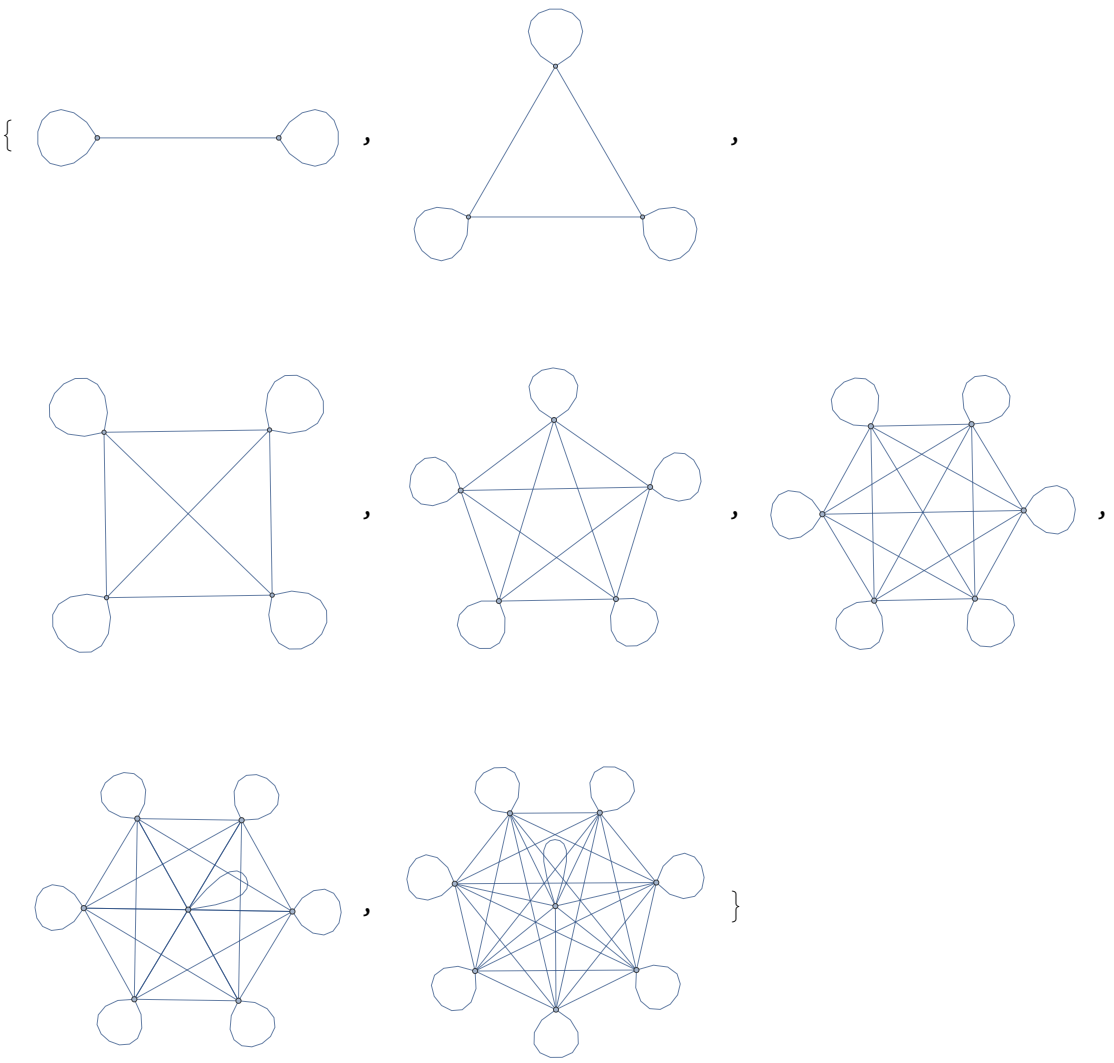
In[ ]:=
```
Graph[Flatten[Table[i → j, {i, 4}, {j, 4}]]]
```
graphe aplatis table

Out[ ]=



```
In[ ]:= Table[UndirectedGraph[Flatten[Table[i → j, {i, n}, {j, n}]]], {n, 2, 8}]
        table  graphe non dirigé  aplatis  table
```

Out[ ]=

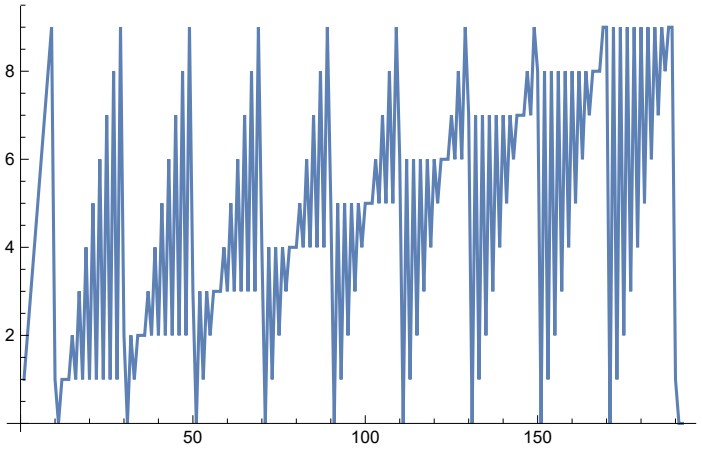


```
In[ ]:= Flatten[Table[{1, 2}, 3]]
        aplatis  table
```

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

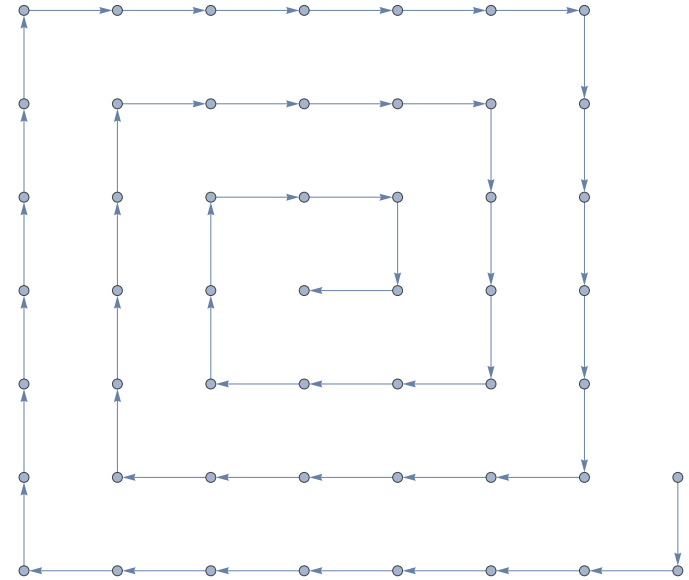
In[•]:=
```
ListLinePlot[Flatten[Table[IntegerDigits[n], {n, 1, 100}]]]
```
tracé de liste de ⋯ | aplatis | table | chiffres d'entier

Out[•]=



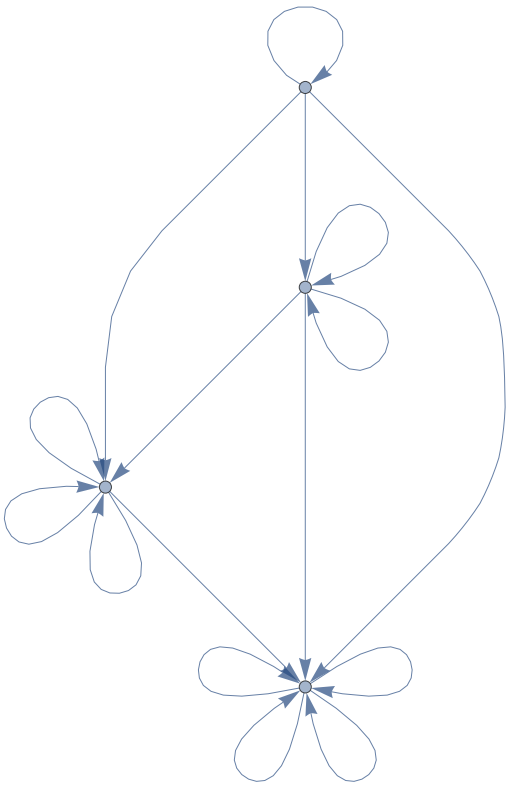
In[•]:=
```
Graph[Flatten[Table[i → i + 1, {i, 1, 50}]]]
```
graphe | aplatis | table

Out[•]=



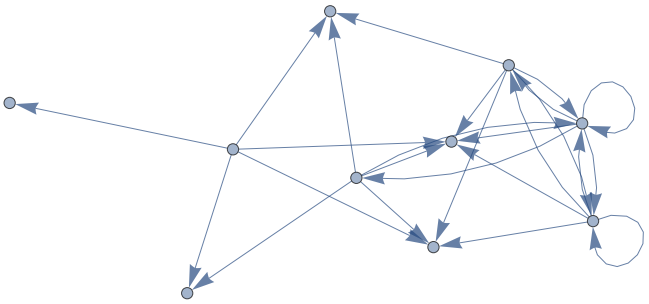
In[•]:=
```
Graph[Flatten[Table[i → Max[i, j], {i, 1, 4}, {j, 1, 4}]]]
```
graphe aplatis table maximum

Out[•]=



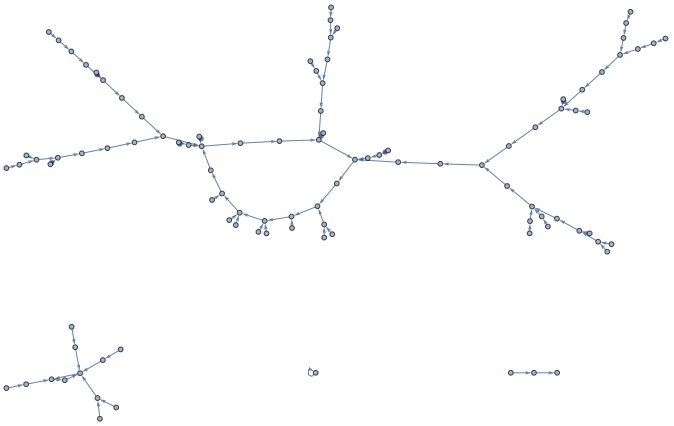
In[•]:=
```
Graph[Flatten[Table[i → j - i, {i, 1, 5}, {j, 1, 5}]]]
```
graphe aplatis table

Out[•]=



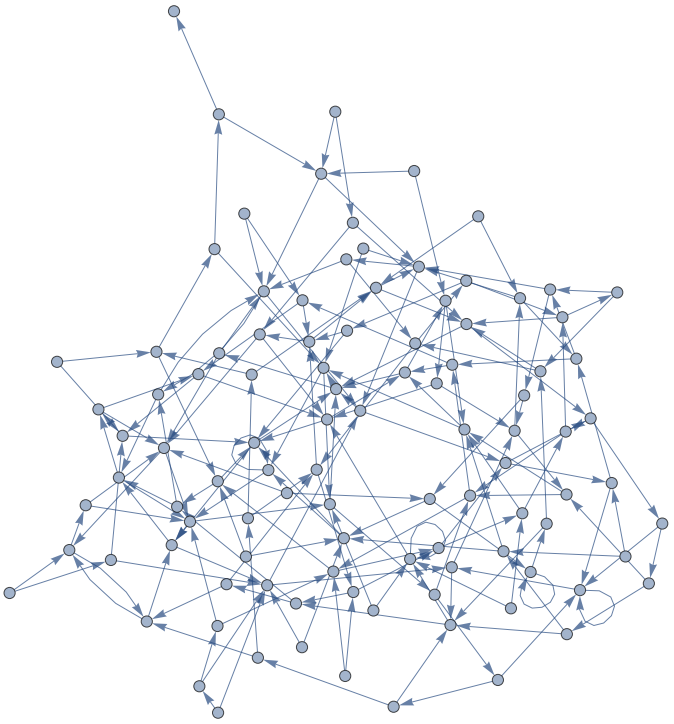
In[•]:=
```
Graph[Flatten[Table[i → RandomInteger[100], {i, 100}]]]
```
graphe aplatis table entier aléatoire

Out[•]=



In[•]:=
```
Graph[Flatten[Table[{i → RandomInteger[100], i → RandomInteger[100]}, {i, 100}]]]
```
graphe aplatis table entier aléatoire entier aléatoire

Out[•]=



In[•]:=
```
g = Graph[{1 → 2, 2 → 3, 3 → 4, 4 → 1, 3 → 1, 2 → 2}];
```
graphe

```
Grid[Table[FindShortestPath[g, i, j], {i, 1, 4}, {j, 1, 4}]]
```
grille table trouve le plus court chemin

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}

Exercices chapitre 22 : “Machine Learning”

```
In[•]:= LanguageIdentify["ajatella"]
```
identifie langue

```
Out[•]= Finnish
```

```
In[•]:= ImageIdentify[tiger SPECIES SPECIFICATION ["Image"], All, 10]
```
identifie image — image — tout

```
Out[•]= {tiger, big cat, feline, carnivorous mammal, mammal,
  vertebrate, chordate, animal, placental mammal, liger}
```

```
In[•]:= Table[ImageIdentify[Blur[tiger SPECIES SPECIFICATION ["Image"], n]], {n, 0, 5}]
```
table — identifie image — flou — image

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

```
In[•]:= Classify["Sentiment", "I'm so happy to be here"]
```
classifie — unité imaginaire

```
Out[•]= Positive
```

```
In[•]:= Nearest[WordList[], "heureux", 10]
```
le plus p··· — liste de mot

```
Out[•]= {heureux, peureux, ferreux, herbeux, heure, heures, heureuse, houleux, terreux, affreux}
```

```
In[•]:= Nearest[Table[RandomInteger[1000], 20], 100, 3]
```
le plus p··· — table — entier aléatoire

```
Out[•]= {53, 46, 39}
```

```
In[•]:= Nearest[Table[RandomColor[], 10], Red, 5]
```
le plus p··· — table — couleur aléatoire — rouge

```
Out[•]= {■, ■, ■, ■, ■}
```

```
In[•]:= Nearest[Table[i^2, {i, 100}], 2000]
```
le plus p··· — table

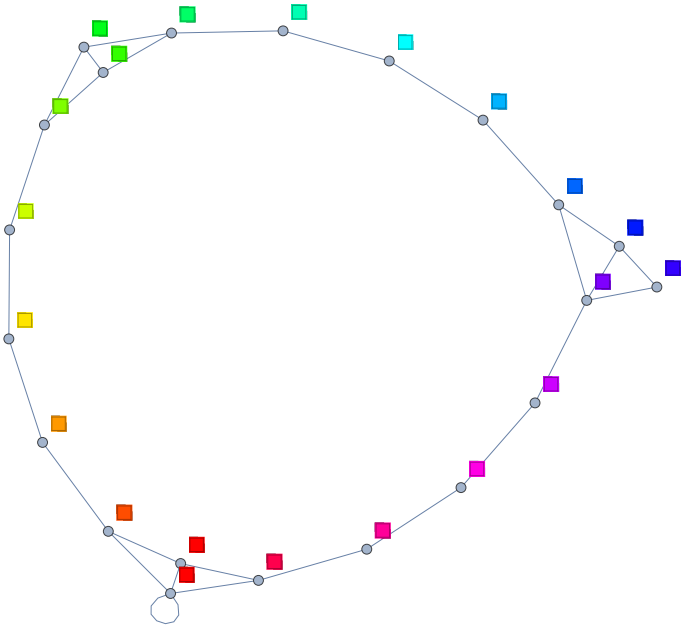
```
Out[•]= {2025}
```

```
In[•]:= Nearest[EntityValue[world GEOGRAPHIC REGION [countries], flag], Brazil COUNTRY [flag], 4]
```
le plus proche

```
Out[•]=
```

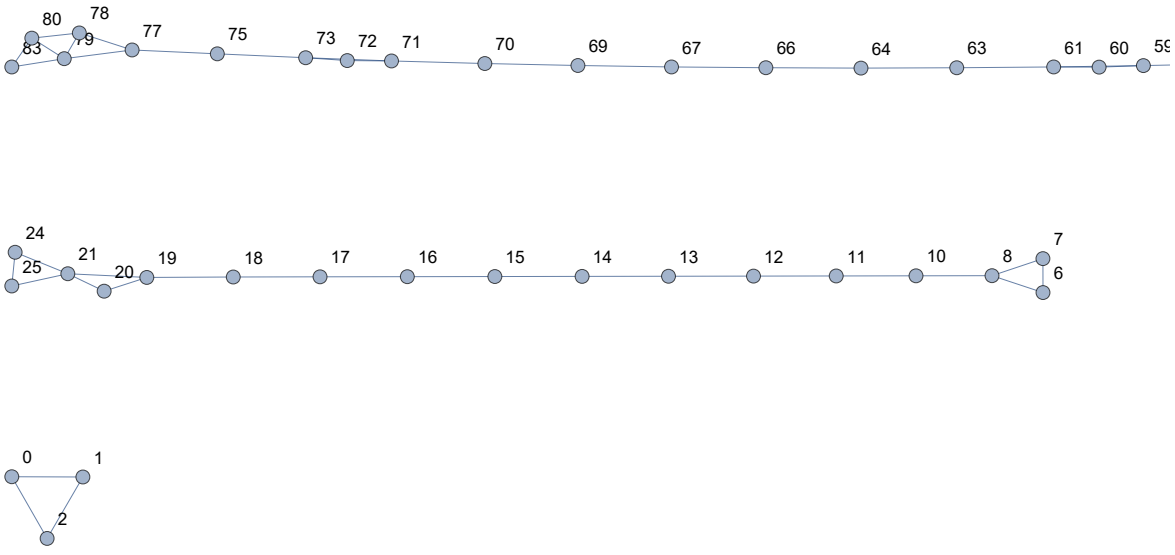
In[ ]:=
```
NearestNeighborGraph[Table[Hue[h], {h, 0, 1, .05}], 2, VertexLabels → All]
```
graphe du plus proche voisin | table | teinte | étiquettes des s··· | tout

Out[ ]=



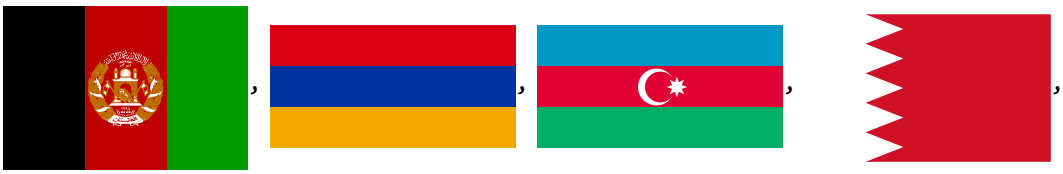
In[ ]:=
```
NearestNeighborGraph[Table[RandomInteger[100], 100], 2, VertexLabels → All]
```
graphe du plus proche voisin | table | entier aléatoire | étiquettes des s··· | tout

Out[ ]=



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

Out[ ]=

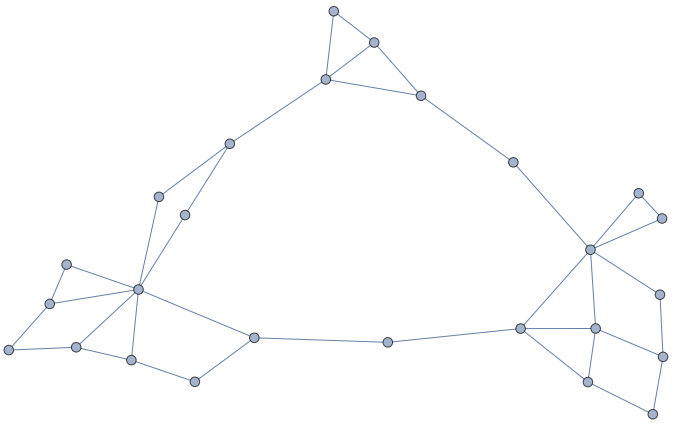


In[ ]:= `data = Table[Rasterize[Style[c, 20]], {c, Alphabet[]}]`
table convertis e··· style alphabet

`NearestNeighborGraph[data, 2]`
graphe du plus proche voisin

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}

Out[ ]=

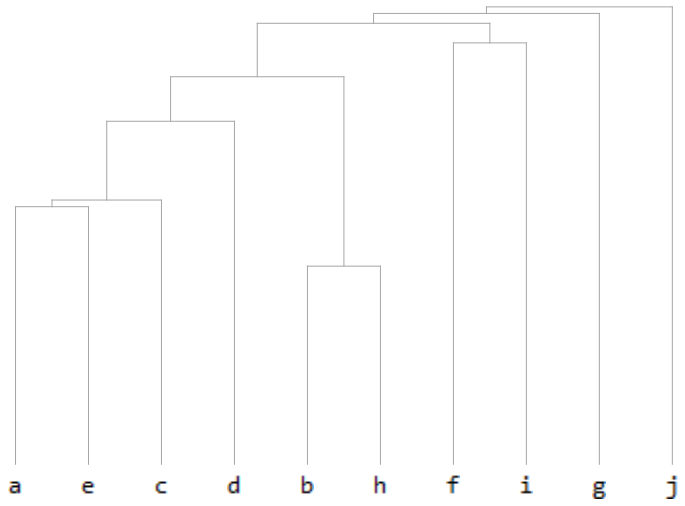


In[ ]:= `TextRecognize[Table[Blur[Rasterize[Style["hello", 50]], n], {n, 1, 15}]]`
reconnais texte table flou convertis e··· style

Out[ ]= {hello, hello, hello, hello, hello, hello, hello, hello, hello, hello, ælle, wwile, , , }

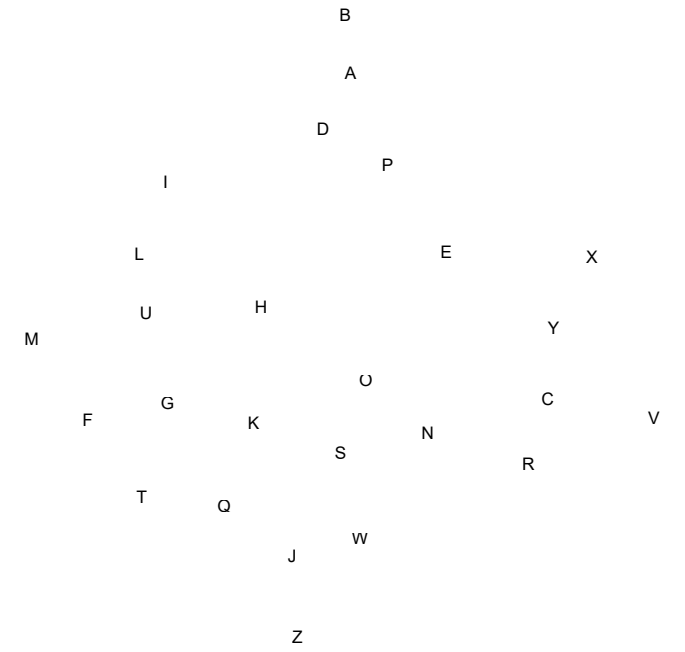
In[•]:=
```
Dendrogram[Table[Rasterize[c], {c, Take[Alphabet[], 10]}]]
```
dendogramme | table | convertis en carte de b··· | pre··· | alphabet

Out[•]=



In[•]:=
```
FeatureSpacePlot[ToUpperCase[Alphabet[]]]
```
diagramme de disper··· | convertis en m·· | alphabet

Out[•]=




```
Table[ImageIdentify[Blur[ Eiffel Tower BUILDING ["Image"], n]], {n, 5}]
```
table | identifie image | flou | image

Out[•]= { Eiffel Tower , Eiffel Tower , tower , stupa , stupa }

In[•]:=
```
Classify["Sentiment", WikipediaData["Happiness"]]
```
classifie | données Wikipedia

Out[•]= Neutral

Exercices chapitre 23 : “More about Numbers”

Exercices chapitre 24 : “More forms of Visualization”

Exercices chapitre 25 : “Ways to apply Functions”

Exercices chapitre 26 : “Pure anonymous functions”

Exercices chapitre 27 : “Applying functions Repeatedly”

Exercices chapitre 28 : “Tests and conditionals”