Computer computations (a short introduction to *Mathematica* computer algebra system)

The goal of the course is not to teach the language of *Mathematica* but to give a rough overview of numerous opportunities provided by it. The actual syntax of one or another command can be found in the Help (addressed at any place of the code by the F1 button)

Lecture 1

Simplest operations, variables, and functions

Type a command. To evaluate it press SHIFT+ENTER simultaneously. The result will show up in the output cell.

In[1]:= 2^2

Out[1]= 4

In[2]:= $\text{Sin}\left[\frac{\pi}{4}\right]$

Out[2]= $\frac{1}{\sqrt{2}}$

To type the exponent I pressed CTRL^. To get the letter π I typed ESCpESC. In general, to type special symbols and forms one can use Palettes called from the *Mathematica* menu. The required form can be chosen with a mouse. The pop-up menu provides a hint how to type the same form with the keyboard without using menu.

In[3]:= $\int \frac{dx}{\text{Sin}[x]+1}$

Out[3]= $\frac{2\,\text{Sin}\left[\frac{x}{2}\right]}{\text{Cos}\left[\frac{x}{2}\right]+\text{Sin}\left[\frac{x}{2}\right]}$

In[4]:= $\sum_{n=1}^{\infty}\frac{1}{n^2}$

Out[4]= $\frac{\pi^2}{6}$

Integers used by *Mathematica* can be arbitrary large, overflow never occurs. Also there is no restriction on the specified precision of numerical computations

```
In[5]:= 200 !
        N[ⅇ, 300]

Out[5]= 788 657 867 364 790 503 552 363 213 932 185 062 295 135 977 687 173 263 294 742 533 244 359 ⋱
         449 963 403 342 920 304 284 011 984 623 904 177 212 138 919 638 830 257 642 790 242 637 105 ⋱
         061 926 624 952 829 931 113 462 857 270 763 317 237 396 988 943 922 445 621 451 664 240 254 ⋱
         033 291 864 131 227 428 294 853 277 524 242 407 573 903 240 321 257 405 579 568 660 226 031 ⋱
         904 170 324 062 351 700 858 796 178 922 222 789 623 703 897 374 720 000 000 000 000 000 000 ⋱
         000 000 000 000 000 000 000 000 000 000

Out[6]= 2.71828182845904523536028747135266249775724709369995957496696762772407663035354 ⋱
         759457138217852516642742746639193200305992181741359662904357290033429526059 ⋱
         563073813232862794349076323382988075319525101901157383418793070215408914993 4 ⋱
         88416750924476146066808226480016847741185374234544243710753907774499207
```

Commands, once input, can be edited and evaluated again. A cell can contain a sequence of commands. If a command ends with “;”, then the result is not printed to the output. If a command ends with a new line, then the result is printed.

```
In[7]:= b = (a + 3)^2;
        b + 5
        a = 2;
        b + 5

Out[8]= 5 + (3 + a)^2

Out[10]= 30
```

The result of the last executed command can be referred to as “%”:

```
In[11]:= %^2

Out[11]= 900
```

(Pay attention to the fact that the symbol “%” refers to the result of the last *executed* command rather
than to the command *printed* immediately before the one to be executed).
In addition to the usual assignments “=”, there are also so called delayed ones “:=”. Compare:

```
In[12]:= a = 1;
         b = a + 1;
         b
         a = 20;
         b

Out[14]= 2

Out[16]= 2
```

```
In[17]:= a = 1;
         b := a + 1;
         b
         a = 20;
         b

Out[19]= 2

Out[21]= 21
```

Delayed assignments are worth to be used to define functions (with arguments). The arguments are put in the square brackets and separated by commas (and the parentheses are only used to group compound expressions).

```
In[22]:= f[x_, y_] := (x + y)^2;
         f[z^2, 1]

Out[23]= (1 + z^2)^2
```

The underscore symbols mean that x and y can stand for arbitrrary expressions. Without those signs the definition would be applied to the symbols x and y themselves only.

```
In[24]:= g[x] := (x + 1)^2;
         g[6]
         g[x]

Out[25]= g[6]

Out[26]= (1 + x)^2
```

It is possible to supply patterns of the form “x_” with various conditions following the “/;” symbol. If these conditions are not satisfied, then the expression remains unevaluated.

```
In[27]:= Clear[f];
         f[n_Integer] /; n > 0 := n f[n - 1];
         f[n_Integer] /; n < 0 := ∞;
         f[0] = 1;
         1/f[5]
         1/f[-5]
         1/f[c]

Out[31]= 1/120

Out[32]= 0

Out[33]= 1/f[c]
```

Here the “Clear” command is used to erase all the previous definitions related to the symbol “f”. Active assignments can be shown using the “?” symbol:

```
In[34]:= ? f

Global`f

f[0] = 1

f[n_Integer] /; n > 0 := n f[n - 1]

f[n_Integer] /; n < 0 := ∞
```

Mathematica (M.) examines the expression to be evaluated and searches for its parts for which there are defined assignments, and then applies them. The order in which these definitions are applied is chosen so that more specific ones are applied prior to more general ones. Thus, observing “f[-5]” *M.* checks first if the argument “-5” is “0”. Since this is not the case, the first definition is not applicable. Then it checks if it is a positive integer. Since it is not, the second definition neither is

$\frac{1}{\infty} = 0$

applicable. Finally, the third definition is applicable and we get the result $\frac{1}{\infty} = 0$.

When calling a function with arguments, along with the usual notation, one can also use prefix and postfix notation indicated by the symbols “@” and “//”, respectively

```
In[35]:= f[5]
         f@5
         5 // f
Out[35]= 120

Out[36]= 120

Out[37]= 120
```

Together with named functions there are also so called *pure* functions (with no name associated to them). A pure function is denoted by the symbol “&” at the end and its argument is denoted by “#”. Compare:

```
In[38]:= g[x_] := (x + 1)^2;
         g[q]
         (# + 1)^2 &[q]
Out[39]= (1 + q)^2

Out[40]= (1 + q)^2
```

Pure functions in combination with postfix notation are convinient for final formatting of the computed expressions

```
In[41]:= b = (c^3 + 7 + x)^2;
         b^2 // Expand
         b^2 // Collect[#, x] &
Out[42]= 2401 + 1372 c^3 + 294 c^6 + 28 c^9 + c^12 + 1372 x + 588 c^3 x +
          84 c^6 x + 4 c^9 x + 294 x^2 + 84 c^3 x^2 + 6 c^6 x^2 + 28 x^3 + 4 c^3 x^3 + x^4

Out[43]= 2401 + 1372 c^3 + 294 c^6 + 28 c^9 + c^12 +
          (1372 + 588 c^3 + 84 c^6 + 4 c^9) x + (294 + 84 c^3 + 6 c^6) x^2 + (28 + 4 c^3) x^3 + x^4
```

Lists and other expressions

Arrays, vectors, matricies, and tables are realized in *Mathematica* as lists and nested lists. A *list* is a sequence of its components separated by commas and enclosed into figure brackets “{“,”}”. A matrix can be input as a table but it is still interpreted internally as a nested list.

```
In[44]:= {1, 3, a, x + 5}
```

$$\begin{pmatrix} 1 & z & u+1 \\ 0 & 1 & 0 \\ d^2 & 0 & 1 \end{pmatrix}$$

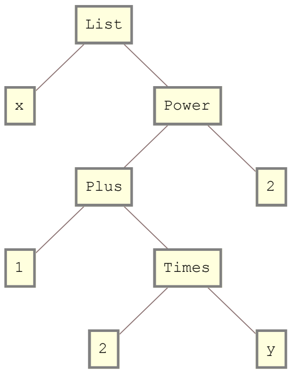
```
Out[44]= {1, 3, 20, 5 + x}

Out[45]= {{1, z, 1 + u}, {0, 1, 0}, {d^2, 0, 1}}
```

A list is a simplest example of what is considered as an *expression*. In fact everything in *Mathematica* is an expression. An expression consists of a head and a sequence of its elemets. Each element, in turn, can be either an atom (a number, a variable etc.) or again an expression with its own head, a list of its elemends, and so on. As a result we get a tree-like expression. For example, the operations “+“, “×”, or taking a power have the heads “Plus”, “Times”, and “Power”, respectively, and a list has the head “List”.

```
In[46]:= Plus[c, 4]
         Times[c, 4]
         Power[c, 4]
         List[c, 4]
Out[46]= 4 + c
Out[47]= 4 c
Out[48]= c^4
Out[49]= {c, 4}
```

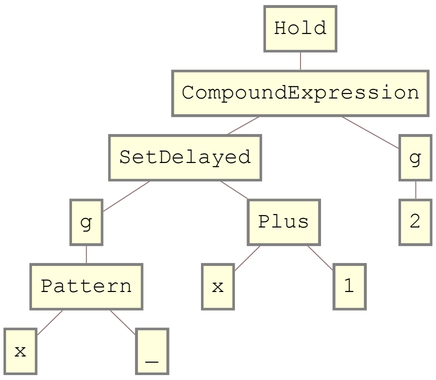
The actual structure of an expression can be viewed using “FullForm” or “TreeForm” commands.

```
In[50]:= {x, (2 y + 1)^2} // FullForm
Out[50]//FullForm= List[x, Power[Plus[1, Times[2, y]], 2]]

In[51]:= {x, (2 y + 1)^2} // TreeForm
Out[51]//TreeForm=
```



All special symbols that we met before and will meet later, like “&”, “>”, “_”, and including “=”, “:=”, and even “;”, are just shorthands of the corresponding expressions with appropriate heads. The next example uses the command Hold that prevents an expression from evaluation, in order to see its structure.

```
In[52]:= Hold[
           g[x_] := x + 1;
           g[2]      ] // TreeForm
Out[52]//TreeForm=
```



An element of a list (or a matrix) can be refferred to using double square brackets. Here are several equivalent (from the point of view of interpretation) ways to do this

An element of a list (or a matrix) can be refferred to using double square brackets. Here are several equivalent (from the point of view of interpretation) ways to do this

In[53]:= $\mathbf{M} = \begin{pmatrix} 1 & z & u+1 \\ 0 & 1 & 0 \\ d^2 & 0 & 1 \end{pmatrix}$

```
M[[1, 3]]
M⟦1, 3⟧
M⟦1,3⟧
M⟦1⟧⟦3⟧
```

Out[53]= $\{\{1, z, 1+u\}, \{0, 1, 0\}, \{d^2, 0, 1\}\}$

Out[54]= $1 + u$

Out[55]= $1 + u$

Out[56]= $1 + u$

Out[57]= $1 + u$

The expression “M⟦1,3⟧” means that we take the first element “M⟦1⟧” in the expression “M”, and then, in turn, take the 3rd element in “M⟦1⟧”. Unexpectedly, similar rules are applicable to expressions with arbitrary heads, not necessary lists!

In[58]:=
```
b
b⟦1, 2⟧
b⟦1, 2, 2⟧
```

Out[58]= $(7 + c^3 + x)^2$

Out[59]= c^3

Out[60]= 3

The head of an expression can be obtained either using “Head” command or as its 0th element,

In[61]:=
```
b
Head[b]
b⟦0⟧
```

Out[61]= $(7 + c^3 + x)^2$

Out[62]= Power

Out[63]= Power

Replacements

The replacement is one of the most efficient operations in *Mathematica*. It consists in replacing some the parts in a given expression with new ones.

```
In[64]:= b
         b /. x → y
         b /. Plus → Times
         b /. Power → pow
```

Out[64]= $(7+c^3+x)^2$

Out[65]= $(7+c^3+y)^2$

Out[66]= $49\ c^6\ x^2$

```
Out[67]= pow[7 + x + pow[c, 3], 2]
```

Similarly to the delayed assignments, patterns for the substituted parts can use “_” and conditions. In that case instead of usual replacement symbol “->” it is better to use the symbol “:>” of delayed replacement (approximately by the same reason that causes to use the delayed assignment command “:=” instead of the usual assignment “=” in the definitions of new functions). Here is an example of how the geometric progression can be turned into an exponent!

In[68]:= $\frac{1}{1-\epsilon\, x} + O[\epsilon]^8$

$\% \;/.\; x^{k\_} :\!\to \frac{x^k}{k\,!}$

`Log[%]`

Out[68]= $1 + x\,\epsilon + x^2\,\epsilon^2 + x^3\,\epsilon^3 + x^4\,\epsilon^4 + x^5\,\epsilon^5 + x^6\,\epsilon^6 + x^7\,\epsilon^7 + O[\epsilon]^8$

Out[69]= $1 + x\,\epsilon + \frac{x^2\,\epsilon^2}{2} + \frac{x^3\,\epsilon^3}{6} + \frac{x^4\,\epsilon^4}{24} + \frac{x^5\,\epsilon^5}{120} + \frac{x^6\,\epsilon^6}{720} + \frac{x^7\,\epsilon^7}{5040} + O[\epsilon]^8$

Out[70]= $x\,\epsilon + O[\epsilon]^8$

The symbol “$O[\epsilon]^8$” is used here for the power expansion. Now we collect all the commands together:

In[71]:= $\frac{1}{1-\epsilon\, x} + O[\epsilon]^8 \;/.\; x^{k\_.} :\!\to \frac{y^k}{k\,!}$ `// Log`

Out[71]= $y\,\epsilon + O[\epsilon]^8$

We also replaced here the variable “x with “y”. The point in the pattern “k_.” means that it is applyed for the optional value of “k” which is equal to 1 in the case of the exponent. Without this point the replacenment would not be applied to the linear term $x = x^1$. There is a reason to Introducing a supplementary variable ϵ as the expansion parameter has a reason of its own. The following version does not work :

In[72]:= $\frac{1}{1-x} + O[x]^8$

$\% \;/.\; x^{k\_} :\!\to \frac{x^k}{k\,!}$

Out[72]= $1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + O[x]^8$

Out[73]= $1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + O[x]^8$

The reason is in the internal form of representation of power series in *Mathematica*.

In[74]:=
$$\frac{1}{1-2\,x} + O[x]^5$$
```
% // FullForm
```

Out[74]= $1 + 2\,x + 4\,x^2 + 8\,x^3 + 16\,x^4 + O[x]^5$

Out[75]//FullForm=
```
SeriesData[x, 0, List[1, 2, 4, 8, 16], 0, 5, 1]
```

We see that the series is represented by the expansion variable, the center on the expansion, the list of coefficients and the range of their exponents. Neither the monomials “x^k” themselves nor the symbol “O” are used in the internal representation. They are only used for formatting of the output.

In the next example, we use replacements to create the list of all partitions of a given number n.

In[76]:=
```
n = 6;
```
$$\prod_{i=1}^{n} \frac{1}{1 - x_i\, s^i} \;//$$
```
      SeriesCoefficient[#, {s, 0, n}] & // Expand //
    # /. Plus → List & //
```
$\#\ //.\ \{x_{i\_\_}\, x_{j\_\_} :\to x_{i,j},\ x_{i\_\_}^{k\_} :\to x_{i,i}\, x_i^{k-2}\}\ \&\ //$
```
  # /. x_k__ :→ Sort[{k}, Greater] & //
 Sort
```

Out[77]=
```
{{6}, {3, 3}, {4, 2}, {5, 1}, {2, 2, 2}, {3, 2, 1}, {4, 1, 1},
 {2, 2, 1, 1}, {3, 1, 1, 1}, {2, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1}}
```

Now the same in detail

1) Take the product of geometric progressions and select its homogeneous term of degree n. It is the sum of all monomials of the quasihmogeneous degree n, with the degree of the variable $x_i$set to be i, i=1,2,3,...

In[78]:=
$$\prod_{i=1}^{n} \frac{1}{1 - x_i\, s^i} \;//$$
```
  SeriesCoefficient[#, {s, 0, n}] & // Expand
```

Out[78]= $x_1^6 + x_1^4\, x_2 + x_1^2\, x_2^2 + x_2^3 + x_1^3\, x_3 + x_1\, x_2\, x_3 + x_3^2 + x_1^2\, x_4 + x_2\, x_4 + x_1\, x_5 + x_6$

2) Convert the sum into a list

In[79]:=
```
% /. Plus → List
```

Out[79]= $\{x_1^6,\ x_1^4\, x_2,\ x_1^2\, x_2^2,\ x_2^3,\ x_1^3\, x_3,\ x_1\, x_2\, x_3,\ x_3^2,\ x_1^2\, x_4,\ x_2\, x_4,\ x_1\, x_5,\ x_6\}$

3) Replace the product of several variables with indicies by a single variable with the corresponding multiindex. The command “//.” means “apply the replacement repeatedly untill the replacement rule is applicable”. The list of replacement rules consists of two elements, therefore, we enclose it into figure brackets. The pattern “i__” (with two underscore characters) means “a sequence of one or more elements”.

In[80]:= $\%\ //.\ \{x_{i\_\_}\, x_{j\_\_} :\to x_{i,j},\ x_{i\_\_}^{k\_} :\to x_{i,i}\, x_i^{k-2}\}$

Out[80]= $\{x_{1,1,1,1,1,1},\ x_{1,1,2,1,1},\ x_{1,1,2,2},\ x_{2,2,2},\ x_{1,1,1,3},\ x_{3,1,2},\ x_{3,3},\ x_{4,1,1},\ x_{2,4},\ x_{1,5},\ x_6\}$

4) It remains to convert the multiindices thus obtained, after sorting them, to partitions

```
In[81]:= % /. x_k__ :> Sort[{k}, Greater] //
          Sort

Out[81]= {{6}, {3, 3}, {4, 2}, {5, 1}, {2, 2, 2}, {3, 2, 1}, {4, 1, 1},
          {2, 2, 1, 1}, {3, 1, 1, 1}, {2, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1}}
```

Note that the same goal can be achieved in a slightly simpler way...

```
In[82]:= IntegerPartitions[n]

Out[82]= {{6}, {5, 1}, {4, 2}, {4, 1, 1}, {3, 3}, {3, 2, 1}, {3, 1, 1, 1},
          {2, 2, 2}, {2, 2, 1, 1}, {2, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1}}
```

Lecture 2

Along with the input and output cells, it is handfull to use the title cells of different levels whose style can be modified by means of the menu. There are also text cells that can be used for comments. Comments can be placed also inside the input cell with the help of pairs “(*” and “*)”.

The cells with the common title can be closed temporary in order to reduce the space on the screen. For example, if the reader does not need the content of the first lecture any longer, he or she can close it. In order to close or to open a cell one can doubleclick the corresponding blue bracket on the right border of the window. Alternatively, one can use the menue Cell->Grouping->Open All Subgroups.

Programming

Standard programming tools like branching (If, Which, Switch) and cycles (Do, While, For) are available also in *Mathematica*. The precise syntax can be checked through the Help (in order to call Help, put the mouse on the name of the command and press the F1 button on the keyboard); here we only point out several specific details. Being commands in *Mathematica*, they consist of the Head and the list of arguments in the square brackets.

```
In[1]:= Z[a_] := If[a == 0, "zero", "non-zero"];
        Z[0]
        Z[1]
        Z[x]

Out[2]= zero

Out[3]= non-zero

Out[4]= If[x == 0, zero, non-zero]
```

Note that if the result of the test a==0 is neither True nor False, then the result of the If command remains unevaluated. To fix this, one can add the 4th argument to the If command or to use “===” instead of “==”

```
In[5]:= x == 0
        x === 0

Out[5]= x == 0

Out[6]= False
```

Let us look now at the followig four commands, namely, Do, Sum, Product, and Table having a similar syntax

```
In[7]:=  Clear[a];
         tot = 0;
         Do[tot = tot + a_i, {i, 5}]
         tot
         Sum[a_i, {i, 5}]
         Product[a_i, {i, 5}]
         Table[a_i, {i, 5}]
```

Out[10]= $a_1 + a_2 + a_3 + a_4 + a_5$

Out[11]= $a_1 + a_2 + a_3 + a_4 + a_5$

Out[12]= $a_1\ a_2\ a_3\ a_4\ a_5$

Out[13]= $\{a_1, a_2, a_3, a_4, a_5\}$

The action of these commands is also similar: they evaluate the expression for the subsequent values of the iterator in turn. The only difference is that they substitute different heads to the resulting list of values, namely: Plus (for Sum), Times (for Product), List (for Table) or no head for Do.
Let us point out two specific features. First, the Sum and the Product commands admit nice 2D versions

In[14]:= $\sum_{i=1}^{5} a_i$

Out[14]= $a_1 + a_2 + a_3 + a_4 + a_5$

Second, the range of values of the iterator can be represented explicitely by a list. This could be rather convenient

```
In[15]:= list = {1, q, Sin};
         tot = 0;
         Do[tot = tot + a_i, {i, list}]
         tot
```

$\sum_{i}^{list} a_i$

Out[18]= $a_1 + a_q + a_{Sin}$

Out[19]= $a_1 + a_q + a_{Sin}$

The variables used for the iterator are `local' meaning that they do not change the values of these variables assigned outside of the cycle. Such local variables can be organized explicitely in one of the following ways

```
In[20]:= tot = 0;
         Block[{tot = 5}, tot]
         Module[{tot = 6}, tot]
         With[{tot = 7}, tot]
         tot

Out[21]= 5

Out[22]= 6

Out[23]= 7

Out[24]= 0
```

As one can see, the result is the same in all three cases. The difference of these commands is in their implementation. I recommend to use Block, and a concerned reader can figure out the details in the Help.

As one can see, the result is the same in all three cases. The difference of these commands is in their implementation. I recommend to use Block, and a concerned reader can figure out the details in the Help.

Here is an example of using While for finding the Taylor expansion of an implicit function. Note that the While command has two arguments (the continuation test and the body of the cycle) but each argument may contain a sequence of several commands separated by a semicolon. This is a manifestation of a general (even though somewhat unexpected) principle in *Mathematic:*

the comma is more important than the semicolon

Namely, the comma separates the arguments of functions, and the semicolon collects a sequence of commands inside an argument.

```
In[25]:= Clear[f, y]
         f[x_, y_] := Sin[ x y] + x^3 - y^2;
         y == f[x, y]
         y = Block[{y = 0, k = 0, ynew, z, eq, sol},
           While[k = k + 1;
            k ≤ 10,
            ynew = y + a x^k;
            z = ynew - f[x, ynew];
            eq = SeriesCoefficient[z, {x, 0, k}] == 0;
            sol = Solve[eq];
            y = ynew /. sol[[1]]];
           y + O[x]^11]
         (* check *)
         f[x, y] - y

Out[27]= y == x^3 - y^2 + Sin[x y]

Out[28]= x^3 + x^4 + x^5 - 2 x^7 - 5 x^8 - 7 x^9 - 4 x^10 + O[x]^11

Out[29]= O[x]^11
```

Such a big number of auxiliary variables is used here to demonstrate the usage of “;”. One can readily manage without them in this particular case:

```
In[30]:= Block[{y = 0},
          Do[
           y = y + a x^k;
           y - f[x, y] //
              SeriesCoefficient[#, {x, 0, k}] & //
             Solve[# == 0] & //
            (y = y /. #[[1]]) &,
           {k, 10}];
          y + O[x]^11]

Out[30]= x^3 + x^4 + x^5 - 2 x^7 - 5 x^8 - 7 x^9 - 4 x^10 + O[x]^11
```

Mathematica provides also a number of variations of cycles that have no analogues in other programming languages. These are the cycles Nest, NestWhile, and FixedPoint. An interesting soprety of these cycles is that they use a function as one of their arguments.

```
In[31]:= Nest[g, x, 5]

Out[31]= g[g[g[g[g[x]]]]]
```

The command Nest applies the function to the initial value a given number of times, NestWhile applies it until a condition remains True, and FixedPoint does while application of the function changes the result. Here are examples of how these cycles can be used to compute the Taylor expansion of an implicit function.

```
In[32]:= Clear[y];
         y == f[x, y]
         Nest[f[x, #] &, O[x], 10]
         NestWhile[f[x, #] &, O[x], #[[5]] < 11 &]
         FixedPoint[(f[x, #] + O[x]^11) &, 0]

Out[33]= y == x^3 - y^2 + Sin[x y]

Out[34]= x^3 + x^4 + x^5 - 2 x^7 - 5 x^8 - 7 x^9 - 4 x^10 + O[x]^11

Out[35]= x^3 + x^4 + x^5 - 2 x^7 - 5 x^8 - 7 x^9 - 4 x^10 + O[x]^11

Out[36]= x^3 + x^4 + x^5 - 2 x^7 - 5 x^8 - 7 x^9 - 4 x^10 + O[x]^11
```

Manipulation with lists

In spite of the fact that *Mathematica* has all standard attributes of programming languages like cycles of different kinds, the programming style in *Mathematica* has its own specific character. In particular, it shows up in a wide range of tools of manipulation with lists. The lists can be joined, cut, rearranged and so on. Here are, for example, several ways to select elements of a given list starting from the 3rd one.

```
In[37]:= list = Table[a_k, {k, 7}]
         Drop[list, 2]
         Take[list, -5]
         list[[3 ;;]]

Out[37]= {a_1, a_2, a_3, a_4, a_5, a_6, a_7}

Out[38]= {a_3, a_4, a_5, a_6, a_7}

Out[39]= {a_3, a_4, a_5, a_6, a_7}

Out[40]= {a_3, a_4, a_5, a_6, a_7}
```

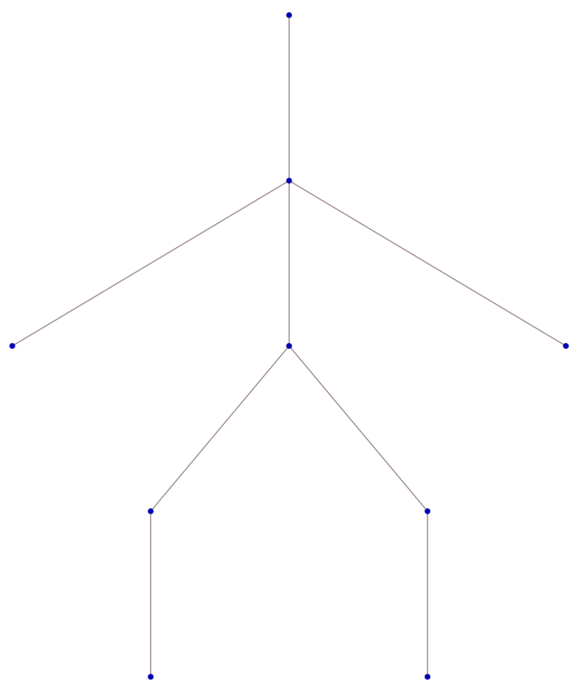
All other tools can easily be found in the Help in the section `See Also’ of any already known command.
Let us recall the convenient replacement operation “/.”, and also mention existence of two important commands Map “/@” and Apply “@@”. The first one applies the function to each element of the list in turn, the second applies the function to the whole list (i.e. it uses the elements of the list as arguments of the function).

```
In[41]:= g /@ {a, b, c}
         g @@ {a, b, c}

Out[41]= {g[a], g[b], g[c]}

Out[42]= g[a, b, c]
```

Example. Enumeration of trees

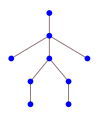
The “tree-like” structute of data in *Mathematica* is well adapted to the problem of enumeration of (plane) trees. Here is an example of a plane tree. Its vertices are denoted by the symbol “v”.

```
In[43]:= t = v[v[v[], v[v[v[]], v[v[]]], v[]]]
         TreeForm[t, VertexLabeling → False]

Out[43]= v[v[v[], v[v[v[]], v[v[]]], v[]]]

Out[44]//TreeForm=
```



In order not to transform the output of a tree each time to a specific form one can use the Format command. It does not affect the result of computations but just changes automatically the formatting of the output expressions.

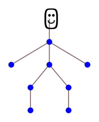
```
In[45]:= Format[t_v] := TreeForm[t,
           VertexRenderingFunction → ({Blue, Point[#]} &),
           ImageSize → Scaled[0.08]]
         t

Out[46]=
```



In fact, the trees used in our enumeration are rooted. It is worth, therefoe, to mark somehow the root vertex in the output.

```
In[47]:= Format[t_v] := TreeForm[u @@ t,
           VertexRenderingFunction → (
             If[#2 === u, Text[" ☺", #], {Blue, Point[#]}] &),
           ImageSize → Scaled[0.08]]
         t
```

Out[48]=

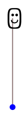


We turn now to enumeration of trees. Let tlist[*k*] produce the list of plane trees with *k* edges.

```
In[49]:= tlist[0] = v[]
         tlist[1] = v[v[]]
```

Out[49]=

Out[50]=



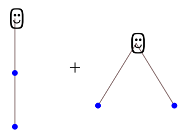
Introduce the “gluing” operation of two subtrees along the top right edge. Every tree with *n* edges can be obtained by glung of two subtrees whose total number of edges is equal to *n* – 1.

```
In[51]:= a_v · b_v := Append[a, b];
```

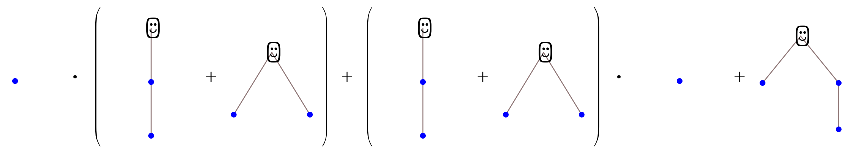
$$\texttt{tlist[k\_]} := \sum_{j=0}^{k-1} \texttt{tlist[j]} \cdot \texttt{tlist[k - 1 - j]};$$

```
         tlist[2]
         tlist[3]
```

Out[53]=



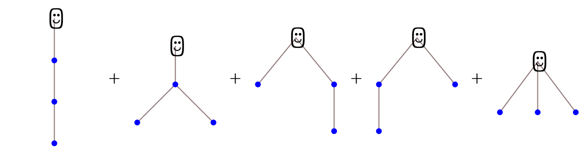
Out[54]=



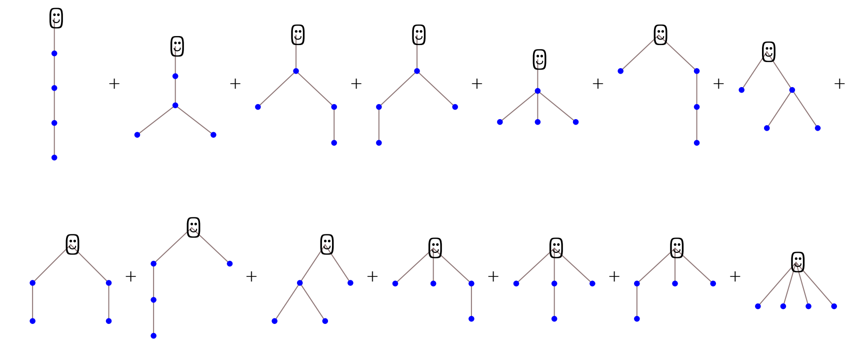
To make this operation working it remains to add the distributivity property to the introduced gluing operation.

```
In[55]:= a___ · b_Plus · c___ := a · # · c & /@b;
         a___ · (k_ b_) · c___ /; FreeQ[k, v] := k a · b · c;
         tlist[3]
         tlist[4]
```

Out[57]=



Out[58]=



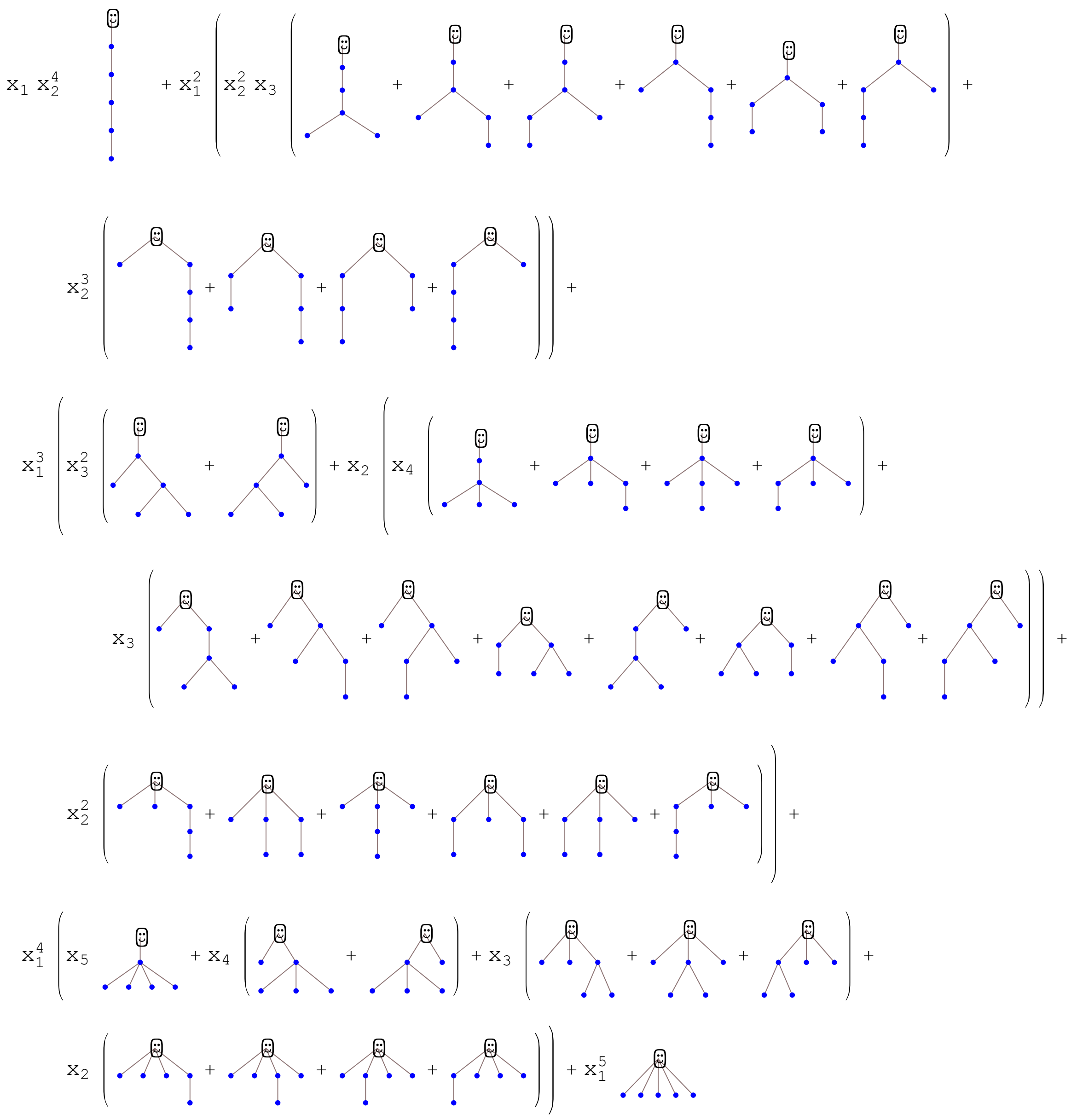
Let the *weight* of a vertex of valency k be the formal variable x_k, and the weight of a tree be the product of the weights of its internal vertices (i.e. excluding the root). Let us enumerate the trees together with their weights. To do that, we insert the corresponding coefficient into the definition of the gluing operation. Here is the final code (consisting of just 5 lines!)

```
In[59]:= Clear[tlist];
         a_v · b_v := x_{Length[b]+1} Append[a, b];
         a___ · b_Plus · c___ := a · # · c & /@b;
         a___ · (k_ b_) · c___ /; FreeQ[k, v] := k a · b · c;
         tlist[0] = v[];
```

$$\texttt{tlist[k\_]} := \sum_{j=0}^{k-1} \texttt{tlist[j]} \cdot \texttt{tlist[k - 1 - j]};$$

```
In[65]:= Collect[tlist[5], x_]
```

Out[65]=



$x_1 x_2^4$
$+ x_1^2 \left(x_2^2 x_3 \right.$
x_2^3
$x_1^3 \left(x_3^2 \right.$
$+ x_2 \left(x_4 \right.$
x_3
x_2^2
$x_1^4 \left(x_5 \right.$
$+ x_4$
$+ x_3$
x_2
$+ x_1^5$

We are ready now to start experiments. The generating function *T* given below for the numbers of (unrooted) plane trees satisfies a number of interesting identities that we are going to check experimentally. For a given tree, its root and the initial edge exiting the root can be chosen in $2n$ different ways (by the number of halfedges). This explains the coefficient $\frac{1}{2n}$ used in the definition of the function *T* below.

In[66]:=
```
ord = 6;
```
$$T = \sum_{n=1}^{ord} \frac{1}{2\,n} \left(\text{tlist[n] /. t\_v} \mapsto x_{\text{Length[t]}}\right) s^n + O[s]^{ord+1}$$

Out[67]=
$$\frac{x_1^2\, s}{2} + \frac{1}{2} x_1^2\, x_2\, s^2 + \left(\frac{1}{2} x_1^2\, x_2^2 + \frac{1}{3} x_1^3\, x_3\right) s^3 + \left(\frac{1}{2} x_1^2\, x_2^3 + x_1^3\, x_2\, x_3 + \frac{1}{4} x_1^4\, x_4\right) s^4 +$$
$$\frac{1}{10}\left(5\, x_1^2\, x_2^4 + 20\, x_1^3\, x_2^2\, x_3 + 5\, x_1^4\, x_3^2 + 10\, x_1^4\, x_2\, x_4 + 2\, x_1^5\, x_5\right) s^5 +$$
$$\frac{1}{6}\left(3\, x_1^2\, x_2^5 + 20\, x_1^3\, x_2^3\, x_3 + 15\, x_1^4\, x_2\, x_3^2 + 15\, x_1^4\, x_2^2\, x_4 + 6\, x_1^5\, x_3\, x_4 + 6\, x_1^5\, x_2\, x_5 + x_1^6\, x_6\right) s^6 + O[s]^7$$

The derivative over x_k enumerates rooted trees having a root of valency k:

In[68]:=
$$Y = \partial_{x_1} T$$

Out[68]=
$$x_1\, s + x_1\, x_2\, s^2 + \left(x_1\, x_2^2 + x_1^2\, x_3\right) s^3 + \left(x_1\, x_2^3 + 3\, x_1^2\, x_2\, x_3 + x_1^3\, x_4\right) s^4 +$$
$$\frac{1}{10}\left(10\, x_1\, x_2^4 + 60\, x_1^2\, x_2^2\, x_3 + 20\, x_1^3\, x_3^2 + 40\, x_1^3\, x_2\, x_4 + 10\, x_1^4\, x_5\right) s^5 +$$
$$\frac{1}{6}\left(6\, x_1\, x_2^5 + 60\, x_1^2\, x_2^3\, x_3 + 60\, x_1^3\, x_2\, x_3^2 + 60\, x_1^3\, x_2^2\, x_4 + 30\, x_1^4\, x_3\, x_4 + 30\, x_1^4\, x_2\, x_5 + 6\, x_1^5\, x_6\right) s^6 + O[s]^7$$

Here is the identity illustrating the gluing operation for trees

In[69]:=
$$\sum_{k=1}^{ord} k\, x_k\, \partial_{x_k} T - s^{-1}\, Y^2$$

Out[69]=
$$O[s]^7$$

The following identities are called the `equations of the dispersionless KdV hierarchy'. They mean that erasing a root of valency k we get k trees all of which have a root of valency 1.

In[70]:=
```
Table[
 k ∂_{x_k} T - Y^k,
 {k, 5}]
```

Out[70]=
$$\left\{O[s]^7,\ O[s]^7,\ O[s]^7,\ O[s]^7,\ O[s]^7\right\}$$

Given a tree with a leaf as a root (i.e. a root of valency 1), erasing this root together with the edge exiting from it, we get a tree with a root of arbitrary valency. This provides an implicit equation on the function Y (the so called *string equation*)

In[71]:=
$$Y - s \sum_{k=0}^{ord} x_{k+1}\, Y^k$$
$$s^{-1}\, Y == x_1 + x_2\, Y + x_3\, Y^2 + x_4\, Y^3 + x_5\, Y^4 + x_6\, Y^5 + O[s]^6 \text{ // ExpandAll}$$

Out[71]=
$$O[s]^7$$

Out[72]=
```
True
```

Finally, there is an explicit closed factorial formula for the coefficients of the series T.

In[73]:=
$$x_{\mu\_List} := \prod_{i}^{\mu} x_i \;/.\; x_{k\_}^{i\_} :\!\to \frac{x_k^i}{i\,!};$$
$$T == \sum_{n=1}^{ord} (n-1)!\, s^n \sum_{\mu}^{IntegerPartitions[2\,n,\{n+1\}]} x_\mu + O[s]^{ord+1} \;//\; ExpandAll$$

```
Out[74]= True
```

Debugging and performance acceleration

If the program `hangs up' or if the computing time is too big, there is a way to interrupt computations by pressing "ALT," or "ALT.". In the last resort, one can use the menu item Evaluation→Quit Kernel. In order to reduce the usage of such extreme means it is useful to control the course of computations. The simplest recommendation is to include debugging prints to the code. A more advanced tool is the usage of dynamic expressions

```
In[75]:= Dynamic[n]
Out[75]= 10
```

The output value of such expression is updated authomatically in real time together with the changes of the expression itself

```
In[76]:= n = 0
         Do[++n; Pause[0.2], {20}]
Out[76]= 0
```

Let us mention a number of workarounds whose aim is to increase the performance of *Mathematica* (I mention only those that appeared in my practical research; the number of simiar tricks is certainly much bigger).

It was already mentioned that there is no restriction in *Mathematica* in the size of integers. On one hand, it is convenient: overflow never occurs. On the other hand, this causes unnecessary waste of resources. Many commands manipulating with polynomials provide an option of computation modulo some prime integer. In many cases, this is quite sufficient for applications but increases substantially the speed of computations.

In[78]:= $PolynomialRemainder[x^{1000}, x^3 - x + 1, x]$

```
Out[78]= 23 518 973 106 647 474 243 173 570 831 046 246 926 693 330 712 148 526 032 572 591 687 622 ⋱
            309 498 489 688 482 516 580 760 616 726 262 316 220 520 458 739 919 681 -
          41 272 920 637 912 249 073 864 590 582 945 550 865 740 267 991 400 135 197 254 350 357 292 ⋱
            734 149 703 593 325 545 149 741 399 703 561 147 103 329 416 371 912 158 x +
          31 156 006 010 332 160 401 160 473 851 414 794 003 295 789 640 412 352 096 378 804 615 733 ⋱
            389 710 564 215 305 223 035 363 727 947 153 368 902 855 636 107 767 041 x^2
```

In[79]:= $PolynomialRemainder[x^{1000}, x^3 - x + 1, x, Modulus \to 113]$

Out[79]= $59 + 35\,x + 87\,x^2$

Combinatorics deals often with matrices of big size with a relatively small number of non-zero entries. It is convenient to store matrices of this type using the command SparseArray. Most matrix operations like summation, multiplication, computing determinant, taking eigenvectors and eigenvalues are typically set up to work as they do with usual matrices.

```
In[80]:= M = SparseArray[{{i_, i_} → 1, {2, 5} → 5}, {6, 6}]
Out[80]= SparseArray[<7>, {6, 6}]
```

```
In[81]:= M // MatrixForm
         M.M // MatrixForm
```

Out[81]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 5 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Out[82]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 10 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

If the results of computations are stores in the replacement list of huge size, then, in order to increase the efficiency, it is worth to apply to this list the command Dispatch. This command packs the replacement list making a more efficient handling of it by the *Mathematica* kernel.

In[83]:= $\texttt{repl = Table}\left[x_i \to i^2, \{i, 10\,000\}\right];$

In[84]:= `Short[repl]`

Out[84]//Short= $\{x_1 \to 1, x_2 \to 4, \ll 9997 \gg, x_{10\,000} \to 100\,000\,000\}$

In[85]:= $\texttt{Timing}\left[\sum_{i=1}^{10\,000} x_i \texttt{ /. repl}\right]$

Out[85]= {8.140625, 333 383 335 000}

In[86]:= `disp = Dispatch[repl];`

In[87]:= $\texttt{Timing}\left[\sum_{i=1}^{10\,000} x_i \texttt{ /. disp}\right]$

Out[87]= {0.015625, 333 383 335 000}

By the way, we used here the Timing command, yet another useful tool to control the efficiency of computations.

For a function defined recursively, it makes sence to store the computed values in order not to cause the program to call the recursion every time when the function is referred to. This may increase the speed of computations to several orders. For example, here is the definition of the Chebyshev polynomials.

In[88]:= `Clear[T];`

$\texttt{T /: } T_1 = x;$

$\texttt{T /: } T_{n\_} := x\, T_{n-1} - \dfrac{1 - x^2}{n - 1}\, \partial_x T_{n-1} \texttt{ // Expand;}$

$\texttt{Timing}[T_{20}]$

Out[91]= $\{21.078125,\ 1 - 200\,x^2 + 6600\,x^4 - 84\,480\,x^6 + 549\,120\,x^8 - 2\,050\,048\,x^{10} + 4\,659\,200\,x^{12} - 6\,553\,600\,x^{14} + 5\,570\,560\,x^{16} - 2\,621\,440\,x^{18} + 524\,288\,x^{20}\}$

Compare with the timing used when the intermediate values are stored:

```
In[92]:= Clear[T];
         T /: T1 = x;
```

$$\texttt{T /: } T_{n\_} \texttt{ := T /: } T_n = x\, T_{n-1} - \frac{1 - x^2}{n - 1} \partial_x T_{n-1} \texttt{ // Expand;}$$

```
         Timing[T20]
```

Out[95]= $\{0., 1 - 200 x^2 + 6600 x^4 - 84\,480 x^6 + 549\,120 x^8 - 2\,050\,048 x^{10} + 4\,659\,200 x^{12} - 6\,553\,600 x^{14} + 5\,570\,560 x^{16} - 2\,621\,440 x^{18} + 524\,288 x^{20}\}$

The symbol “/:” is used here in order to relate the definition of T_n with the symbol T so that the command Clear[T] could be used to clear the definition in the course of debugging.

If there is a need to store the results of computations to a file and to use them in another session of *Mathematica*, this can be done in the following way:

```
In[96]:= SetDirectory[NotebookDirectory[]]
         DumpSave["tmp.mx", {T}]

Out[96]= C:\Users\maxim_2\Documents\Maxim\WNMATH\Разное

Out[97]= {T}
```

Without indication of the directory, *Mathematica* tries to use the system one (which is, most probably, write-protected).
The DumpSave can be replaced by just Save. The difference is that Save stores the data in the readable text form but DumpSave does this in the binary machine-dependent form which is less compatible with different systems but is certainly more efficient. We can now stop the kernel or even exit the *Mathematica* program.

```
In[1]:= SetDirectory[NotebookDirectory[]];
        << tmp.mx
```

All definitions related to the symble T (both the commands defining the Chebyshev polynomials and the computed values) are restored completely.

```
In[3]:= ? T
```

```
Global`T
```

T /: Subscript[T, 1] = x

T /: Subscript[T, 2] = $-1+2\,x^{2}$

T /: Subscript[T, 3] = $-3\,x+4\,x^{3}$

T /: Subscript[T, 4] = $1-8\,x^{2}+8\,x^{4}$

T /: Subscript[T, 5] = $5\,x-20\,x^{3}+16\,x^{5}$

T /: Subscript[T, 6] = $-1+18\,x^{2}-48\,x^{4}+32\,x^{6}$

T /: Subscript[T, 7] = $-7\,x+56\,x^{3}-112\,x^{5}+64\,x^{7}$

T /: Subscript[T, 8] = $1-32\,x^{2}+160\,x^{4}-256\,x^{6}+128\,x^{8}$

T /: Subscript[T, 9] = $9\,x-120\,x^{3}+432\,x^{5}-576\,x^{7}+256\,x^{9}$

T /: Subscript[T, 10] = $-1+50\,x^{2}-400\,x^{4}+1120\,x^{6}-1280\,x^{8}+512\,x^{10}$

T /: Subscript[T, 11] = $-11\,x+220\,x^{3}-1232\,x^{5}+2816\,x^{7}-2816\,x^{9}+1024\,x^{11}$

T /: Subscript[T, 12] = $1-72\,x^{2}+840\,x^{4}-3584\,x^{6}+6912\,x^{8}-6144\,x^{10}+2048\,x^{12}$

T /: Subscript[T, 13] = $13\,x-364\,x^{3}+2912\,x^{5}-9984\,x^{7}+16\,640\,x^{9}-13\,312\,x^{11}+4096\,x^{13}$

T /: Subscript[T, 14] = $-1+98\,x^{2}-1568\,x^{4}+9408\,x^{6}-26\,880\,x^{8}+39\,424\,x^{10}-28\,672\,x^{12}+8192\,x^{14}$

T /: Subscript[T, 15] =
$-15\,x+560\,x^{3}-6048\,x^{5}+28\,800\,x^{7}-70\,400\,x^{9}+92\,160\,x^{11}-61\,440\,x^{13}+16\,384\,x^{15}$

T /: Subscript[T, 16] =
$1-128\,x^{2}+2688\,x^{4}-21\,504\,x^{6}+84\,480\,x^{8}-180\,224\,x^{10}+212\,992\,x^{12}-131\,072\,x^{14}+32\,768\,x^{16}$

T /: Subscript[T, 17] =
$17\,x-816\,x^{3}+11\,424\,x^{5}-71\,808\,x^{7}+239\,360\,x^{9}-452\,608\,x^{11}+487\,424\,x^{13}-278\,528\,x^{15}+65\,536\,x^{17}$

T /: Subscript[T, 18] = $-1+162\,x^{2}-4320\,x^{4}+44\,352\,x^{6}-$
$228\,096\,x^{8}+658\,944\,x^{10}-1\,118\,208\,x^{12}+1\,105\,920\,x^{14}-589\,824\,x^{16}+131\,072\,x^{18}$

T /: Subscript[T, 19] = $-19\,x+1140\,x^{3}-20\,064\,x^{5}+160\,512\,x^{7}-$
$695\,552\,x^{9}+1\,770\,496\,x^{11}-2\,723\,840\,x^{13}+2\,490\,368\,x^{15}-1\,245\,184\,x^{17}+262\,144\,x^{19}$

T /: Subscript[T, 20] = $1-200\,x^{2}+6600\,x^{4}-84\,480\,x^{6}+549\,120\,x^{8}-$
$2\,050\,048\,x^{10}+4\,659\,200\,x^{12}-6\,553\,600\,x^{14}+5\,570\,560\,x^{16}-2\,621\,440\,x^{18}+524\,288\,x^{20}$

Subscript[T, n_] ^:=
T /: Subscript[T, n] = Expand$\left[x\ \text{Subscript[T, n - 1]} - \frac{\left(1-x^{2}\right)\,\partial_{x}\text{Subscript[T,n-1]}}{n-1}\right]$

Lecture 3

Input and output

One can ouput the results of computations omitting the “;” symbol at the end of the line. Another, more explicit way to do that is to use the Print command. It prints out to the output cell all its arguments one after another without spaces. In order to separate the fields, they can be alternated with text strings (enclosed into double quotes). The text strings, in turn, can include expressions using the StringForm command.

In[4]:=
```
Do[Print["n = ", n, StringForm[",  2^`1` = `2`", n, 2^n]], {n, 5}];
```

$n = 1,\ \ 2^1 = 2$

$n = 2,\ \ 2^2 = 4$

$n = 3,\ \ 2^3 = 8$

$n = 4,\ \ 2^4 = 16$

$n = 5,\ \ 2^5 = 32$

If the visual presentation of data in *Mathematica* does not satisfy your needs, it can be changed using the Format command. It does not change the expression itself but just the way it appears in the output. Assume that we have a power series (or a polynomial) whose coefficients are labelled by partitions (Young diagrams).

In[5]:=
$$\mathbf{F} = \sum_{n=1}^{5}\ \sum_{\lambda}^{\mathtt{IntegerPartitions[n]}} \mathbf{Y}[\lambda] \prod_{i}^{\lambda} \mathbf{x_i}$$

Out[5]=
$$x_1\,Y[\{1\}] + x_2\,Y[\{2\}] + x_3\,Y[\{3\}] + x_4\,Y[\{4\}] + x_5\,Y[\{5\}] + x_1^2\,Y[\{1,1\}] + x_1\,x_2\,Y[\{2,1\}] + x_2^2\,Y[\{2,2\}] + x_1\,x_3\,Y[\{3,1\}] + x_2\,x_3\,Y[\{3,2\}] + x_1\,x_4\,Y[\{4,1\}] + x_1^3\,Y[\{1,1,1\}] + x_1^2\,x_2\,Y[\{2,1,1\}] + x_1\,x_2^2\,Y[\{2,2,1\}] + x_1^2\,x_3\,Y[\{3,1,1\}] + x_1^4\,Y[\{1,1,1,1\}] + x_1^3\,x_2\,Y[\{2,1,1,1\}] + x_1^5\,Y[\{1,1,1,1,1\}]$$

Here is the way to enhance the visualization of this expression

In[6]:=
$$\mathtt{Format[Y[\lambda\_List]] := Y_{Row[\lambda]}}$$
```
F
```

Out[7]=
$$x_1\,Y_1 + x_2\,Y_2 + x_3\,Y_3 + x_4\,Y_4 + x_5\,Y_5 + x_1^2\,Y_{11} + x_1\,x_2\,Y_{21} + x_2^2\,Y_{22} + x_1\,x_3\,Y_{31} + x_2\,x_3\,Y_{32} + x_1\,x_4\,Y_{41} + x_1^3\,Y_{111} + x_1^2\,x_2\,Y_{211} + x_1\,x_2^2\,Y_{221} + x_1^2\,x_3\,Y_{311} + x_1^4\,Y_{1111} + x_1^3\,x_2\,Y_{2111} + x_1^5\,Y_{11111}$$

Mathematica contains a lot of nice commands with reduced infix or prefix notations without built-in meaning. These are brackets of different kinds, binary operations like ·,∘,∧,⊗,⊕, and many others. They can be used and defined according to the needs (similarly to the way we defined the gluing operation for trees in the previous lecture). It is worth to recall, however, that these commands are just reduced notations for the corresponding functions with their own heads and the lists of arguments etc.

In[8]:=
```
a ∧ b // FullForm
```

Out[8]//FullForm=
```
Wedge[a, b]
```

Finally, if this is still insufficient, *Mathematica* provides the opportunity to introduce your own notation. To use it, one needs to load an additional Notation package

In[9]:=
```
Needs["Notation`"]
```

Here is an example of an invented notation for the limit:

In[10]:=
$$\mathtt{Notation}\Big[\ \boxed{\lim_{x\_\to a\_} f\_}\ \Longleftrightarrow\ \boxed{\mathtt{Limit[f\_,\ x\_ \to a\_]}}\ \Big]$$

In[11]:=
$$\lim_{x\to 0}\ \mathtt{(Sin[Tan[x]] - Tan[Sin[x]]) / (ArcSin[ArcTan[x]] - ArcTan[ArcSin[x]])}$$

Out[11]=
```
1
```

And here is an example of a new notation for the operation of taking square

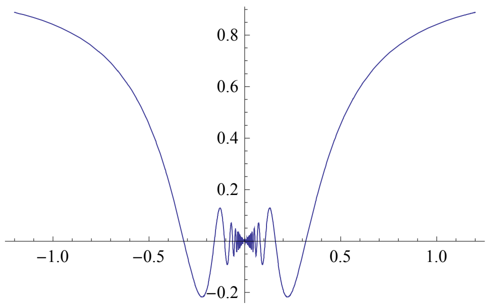
```
In[12]:= Notation[ ■x_  ⟺  x_^2 ]
         ■5
Out[13]= 25

In[14]:=
         Clear[a, b];
         (a + b)^3 // Expand
Out[15]= a^3 + 3 (■a) b + 3 a (■b) + b^3

In[16]:= RemoveNotation[ ■x_  ⟺  x_^2 ]
```

Graphics

Features of woking with graphics are boundless in *Mathematica*. Being unable to describe all details, we will restrict ourselves to just consideration of several elementary examples. The reader is encouraged to develop and to enhance these examples.

The simplest graphic command is Plot. It is used to draw the graph of a function in one variable

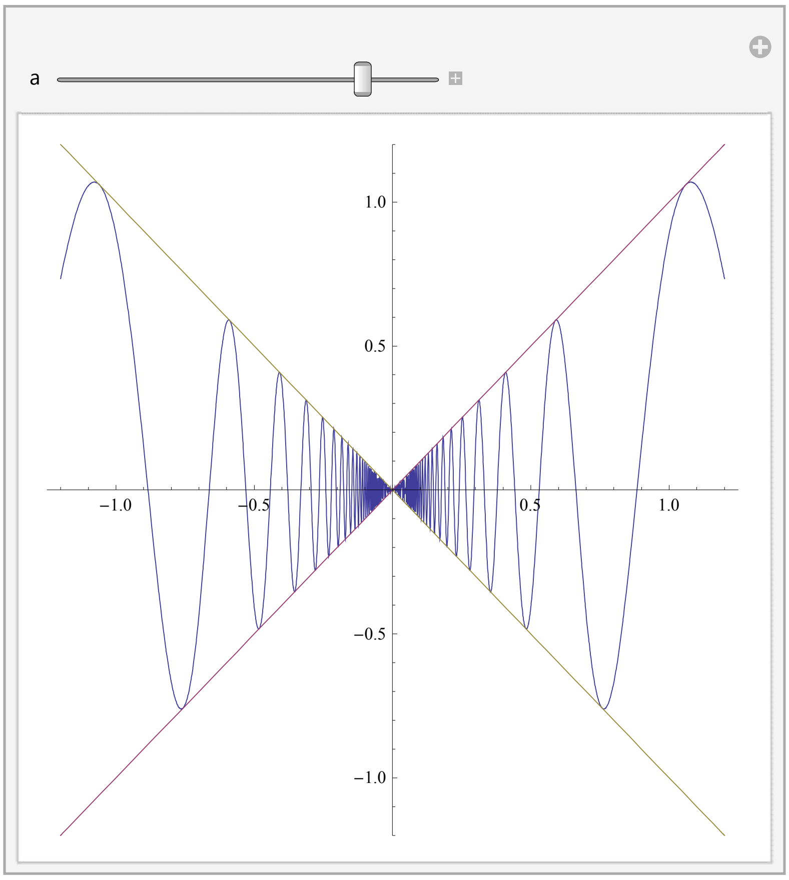
```
In[17]:= Plot[x Sin[1 / x], {x, -1.2, 1.2}]
Out[17]=
```



Let us make a number of enhancements. 1) Add two more functions x and -x for better visualization; 2) Add the option PlotRange to specify the Range of the values of the function and AspectRatio to specify the scaling of the axis; 3) Add a parameter and a simple manipulator for variating its values

```
In[18]:= Manipulate[
          Plot[{x Sin[a / x], x, -x}, {x, -1.2, 1.2},
           PlotRange → {-1.2, 1.2}, AspectRatio → 1],
          {a, 0, 10}]
```

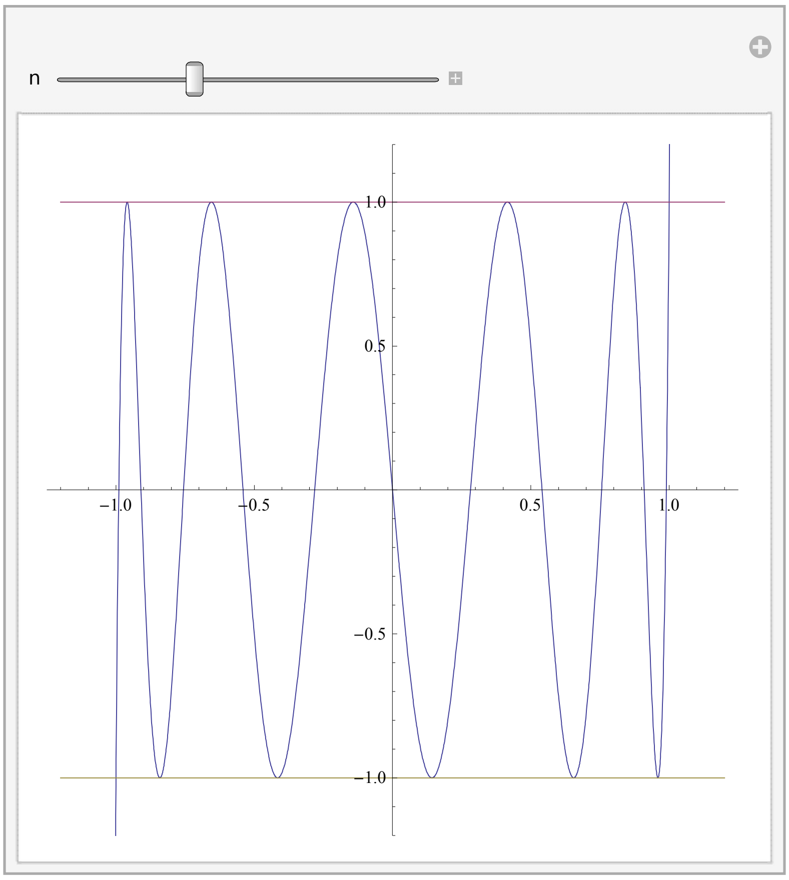
Out[18]=



Here are the familiar Chebyshev polynomials. They have the property that all critical points are real, half of them have the critical value 1, and the remaining ones have the critical value −1. The Plot command works better if the function is computed in advance rather than while plotting the graph.

In[19]:=
```
Manipulate[y = Tn;
 Plot[{y, 1, -1}, {x, -1.2, 1.2},
  PlotRange → {-1.2, 1.2}, AspectRatio → 1],
 {n, 1, 30, 1}]
```

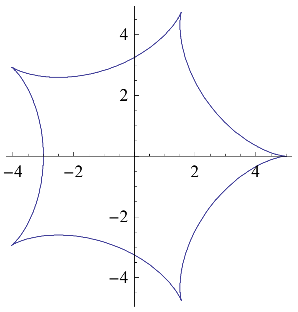
Out[19]=



Here I use the ParametricPlot command to plot the cycloid.

In[20]:=
```
r[ϕ_] := {Cos[ϕ], Sin[ϕ]};
n = 4;
ParametricPlot[n r[ϕ] + r[-n ϕ], {ϕ, 0, 2 π}]
```

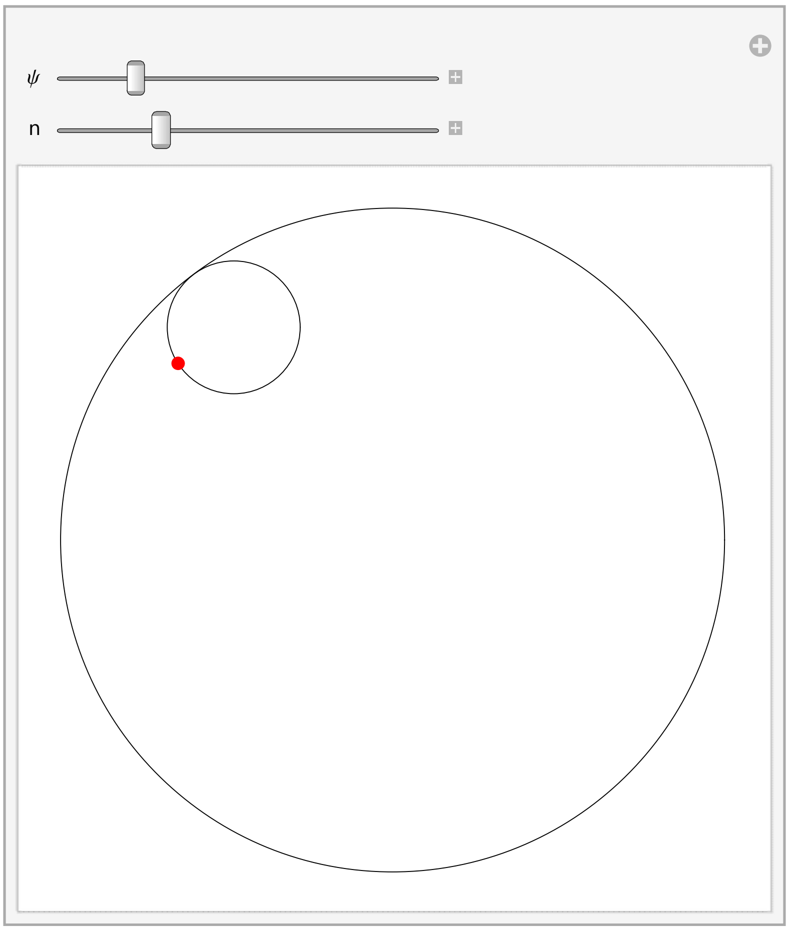
Out[22]=



In order to plot something by hand, one uses the Graphics command. Its argument consists of a sequence of `primitives’ and `directives’ grouped by curly braces. Primitives are the elementary objects to be drawn: a point, a line, a circle, a rectangle etc. A directive describes the way to draw the objects: colour, dashing, thickness of the lines etc. Here is a simple graphics consisting of two circles and a point.

```
In[23]:= Manipulate[
          Graphics[{Circle[{0, 0}, n + 1], Circle[n r[ψ], 1],
            {PointSize[Large], Red, Point[n r[ψ] + r[-n ψ]]}}],
          {ψ, -2 π, 2 π}, {n, 2, 10, 1}]
```

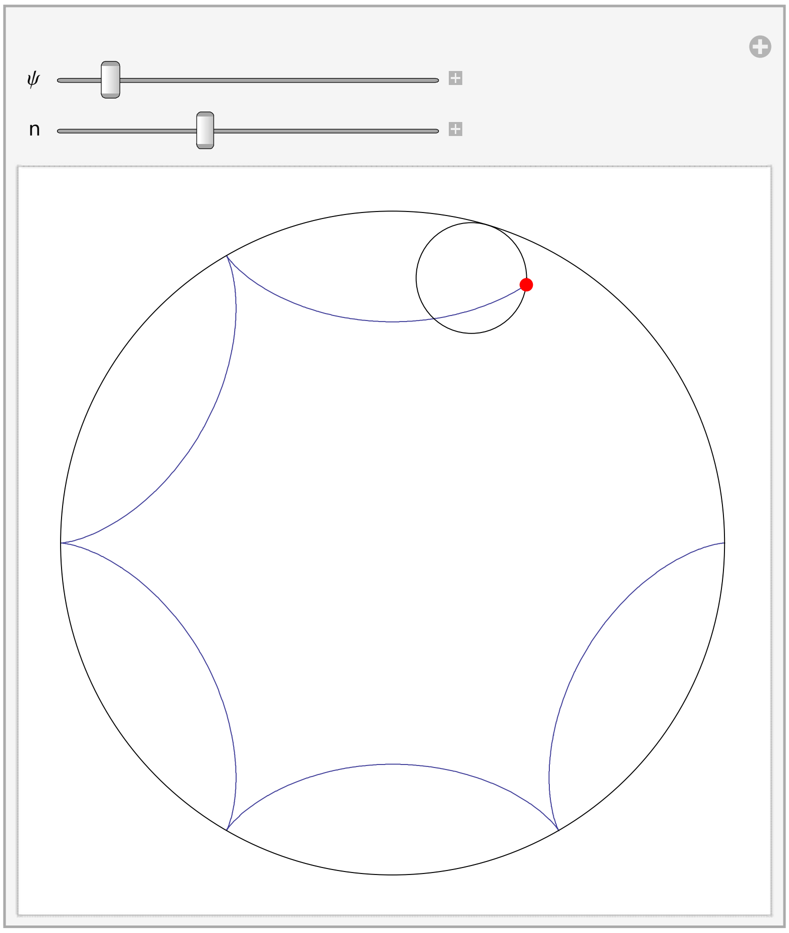
Out[23]=



The cycloid drawn above is the trace of a given point on the small circle rolling along the big one. How to combine two graphics in one picture? It can be done using the Show command. Besides, the Show command can be supplied with graphics options valid for all graphics command to which it is applied.

```
In[24]:= Manipulate[
          Show[
           ParametricPlot[n r[ϕ] + r[-n ϕ], {ϕ, 0, ψ}],
           Graphics[{Circle[{0, 0}, n + 1], Circle[n r[ψ], 1],
             {PointSize[Large], Red, Point[n r[ψ] + r[-n ψ]]}
            }],
           Axes → False, PlotRange → (n + 1) {{-1, 1}, {-1, 1}}],
          {ψ, -2 π, 2 π}, {n, 2, 10, 1}]
```

Out[24]=



The next example illustrates some interactivity opportunities. Let us consider a matrix composed of units and zeroes.

```
In[25]:= n = 10;
         A = RandomInteger[1, {n, n}];
         MatrixForm[A]
```

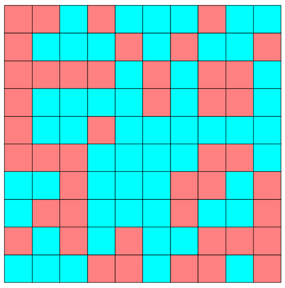
Out[27]//MatrixForm=

$$\begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1
\end{pmatrix}$$

Let us represent this matrix as a checkboard whose fields are colored in two colors depending on the value of the component. Remark that graphics uses traditional orientation of the axes (the first one from left to right, the second one from bottom to top). In order to put this in accrodance with the traditional numeration of rows and columns of matrices we have to rotate the coordinates by 90°.

In[28]:=
```
Graphics[Table[{EdgeForm[Thin], If[A[[n+1-j,i]] == 0, Cyan, Pink], Rectangle[{i, j}]},
    {i, n}, {j, n}], ImageSize → 150] //
 Row[{#, "   A = ", MatrixForm[A]}] &
```

Out[28]=

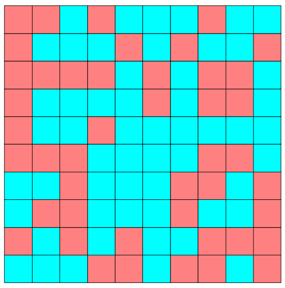


$$A = \begin{pmatrix} 1&1&0&1&0&0&0&1&0&0\\ 1&0&0&0&1&0&1&0&0&1\\ 1&1&1&1&0&1&0&1&1&0\\ 1&0&0&0&0&1&0&1&1&0\\ 1&0&0&1&0&0&0&0&0&0\\ 1&1&1&0&0&0&0&1&1&0\\ 0&0&1&0&0&0&1&1&0&1\\ 0&1&1&0&0&0&1&0&0&1\\ 1&0&1&0&1&0&0&1&1&1\\ 0&0&0&1&1&0&1&1&0&1 \end{pmatrix}$$

And now we would like to realize the possibility to change the colour of a cell (and the corresponding values of the matrix) by just clicking it with the mouse. For this aim we use the command ClickPane. Its first argument is the graphics itself (to which Dynamic is applied), and the second argument is the function which is applied each time the mouse is clicked. The argument of the function is the list of the coordinates of the mouse. In our case we round the coordinates to the integers and change the corresponding entry of the matrix.

In[29]:=
```
ClickPane[Dynamic[
   {Graphics[
      Table[{EdgeForm[Thin], If[A[[n+1-j,i]] == 0, Cyan, Pink], Rectangle[{i, j}]},
       {i, n}, {j, n}], ImageSize → 150], "   A = ", MatrixForm[A]} // Row],
  ({i, j} = Round[# - 1/2]; If[1 ≤ i ≤ n && 1 ≤ j ≤ n, A[[n+1-j,i]] = 1 - A[[n+1-j,i]]]) &]
```

Out[29]=



$$A = \begin{pmatrix} 1&1&0&1&0&0&0&1&0&0\\ 1&0&0&0&1&0&1&0&0&1\\ 1&1&1&1&0&1&0&1&1&0\\ 1&0&0&0&0&1&0&1&1&0\\ 1&0&0&1&0&0&0&0&0&0\\ 1&1&1&0&0&0&0&1&1&0\\ 0&0&1&0&0&0&1&1&0&1\\ 0&1&1&0&0&0&1&0&0&1\\ 1&0&1&0&1&0&0&1&1&1\\ 0&0&0&1&1&0&1&1&0&1 \end{pmatrix}$$

Along with the 2D graphics *Mathematica* provides tools generating 3D graphics. Commands like Plot3D or ParametricPlot3D draw surfaces, and using Graphics3d one can compose 3D graphic objects by hand. 3D graphics can be rotated using the mouse.

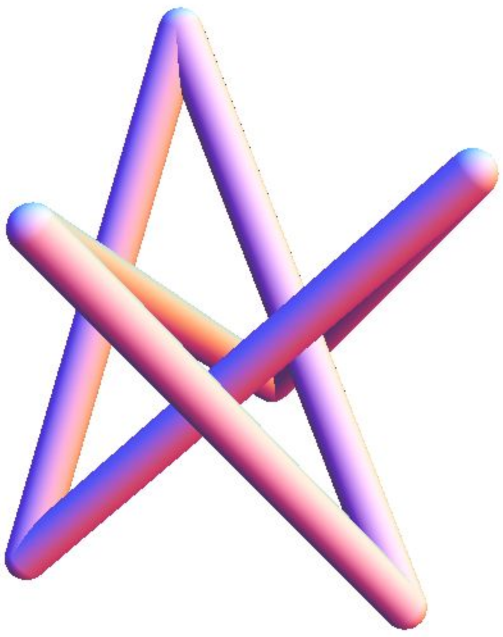
In[30]:=
```
Clear[pp];
pp[a_] := Join @@ Table[{Cos[2 π (i + a z) / 3], Sin[2 π (i + a z) / 3], z},
      {i, 0, 2}, {z, {-1, 1}}] // Append[#, #[[1]]] &;
pp[0]
```

Out[32]=

$$\left\{\{1, 0, -1\}, \{1, 0, 1\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}, -1\right\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}, 1\right\}, \left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}, -1\right\}, \left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}, 1\right\}, \{1, 0, -1\}\right\}$$

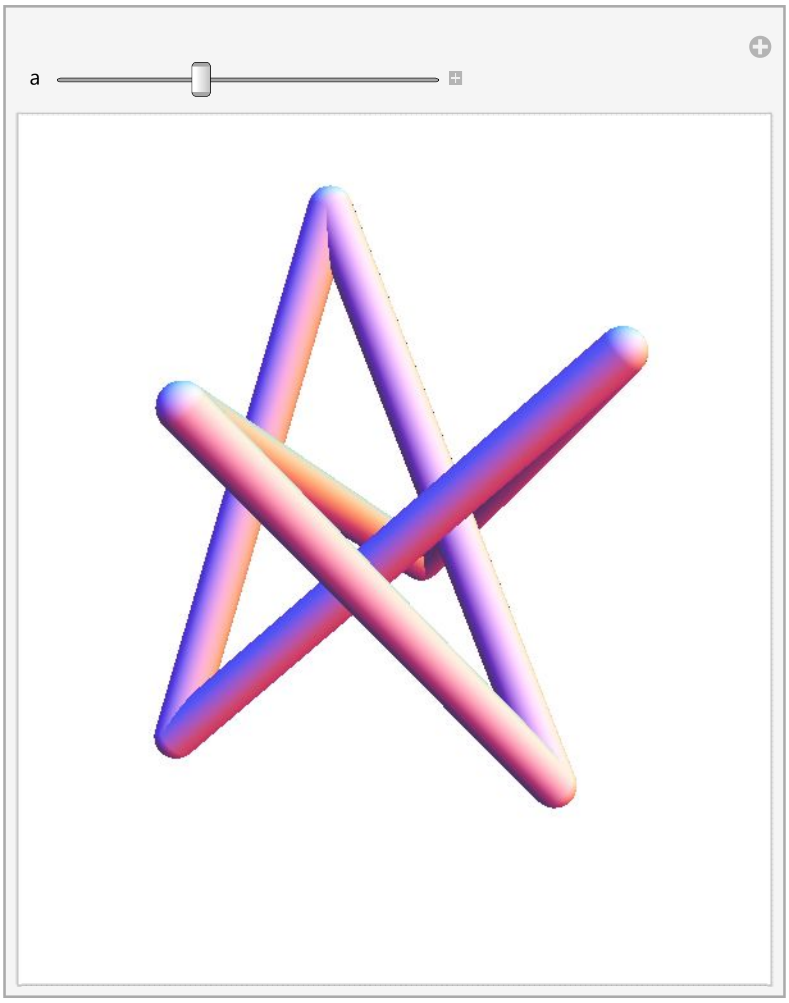
```
In[33]:= Graphics3D[{Tube[pp[1.1], 0.1]}, Boxed → False]
```

Out[33]=



```
In[34]:= Manipulate[
          Graphics3D[{Tube[pp[a], .1]}, Boxed → False], {{a, 1.1}, 0, 3}]
```

Out[34]=



Conclusion

We have shown a very small portion of tools provided by *Mathematica*. More information can be obtained from the Help as well as by studying various examples collected in the Demonstrations project on the Wolfram Website (a reference is avalable in the Help item of the menu).

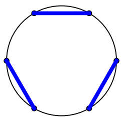
Some more examples

Enumeration of Chord diagrams

A chord diagram is a collection of 2 n points on the circle split into n pairs (represented by chords). Two diagrams are isomorphic (or just the same) if they can be obtained from one another by an orientation-preserving diffeomorphism of the circle.

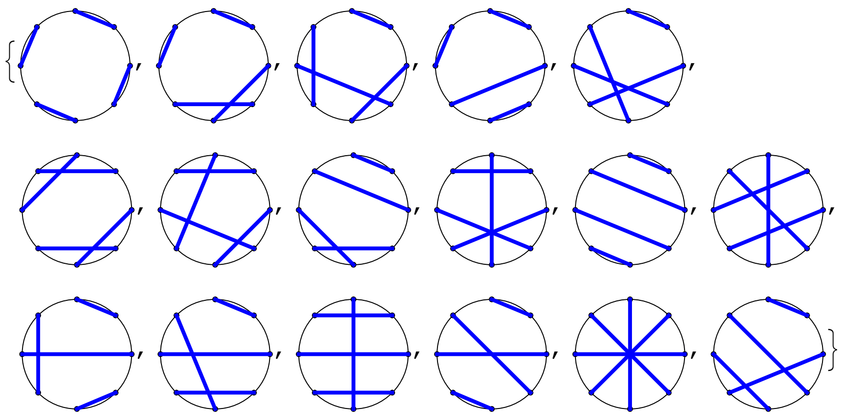
```
In[1]:= pt[α_] := {Cos[2 π α], Sin[2 π α]};
        Format[diag[λ___]] := Block[{n = Length[{λ}]},
            Graphics[{Circle[{0, 0}, 1],
              Thick, Table[Point[pt[k / (2 n)]], {k, 2 n}],
              Blue, Table[Line[pt /@ s / (2 n)], {s, {λ}}]}, ImageSize → 60]];
        diag[{1, 2}, {3, 4}, {5, 6}]
```

Out[3]=



```
In[4]:= smplf[λ_] := Table[Sort[Sort /@ (λ /. j_Integer :→ Mod[j + k, 2 Length[λ]] + 1)],
              {k, 1, 2 Length[λ]}] // Sort // First;
        diag[{1, 3}, {2, 5}, {4, 6}];
        dlist[0] = {diag[]};
        dlist[n_] := dlist[n] =
            Table[Append[λ /. {j_Integer /; j ≥ k :→ j + 1}, {k, 2 n}] // smplf,
               {k, 1, 2 n - 1}, {λ, dlist[n - 1]}] // Flatten // DeleteDuplicates;
        dlist[4]
        Length[%]
```

Out[8]=



```
Out[9]= 17
```

Inversion of a series

$x = y + a_1 y^2 + a_2 y^3 + \ldots$

```
In[10]:= Clear[a, y];
         Block[{c, y},
          y = x;
          Do[
           y = y + c x^k;
           y = y /. (SeriesCoefficient[y + Sum[a_i y^(i+1), {i, 1, k-1}] - x, {x, 0, k}] == 0 //
                 Solve[#, c] & // First),
           {k, 2, 6}];
          y]
```

Out[11]= $x - x^2 a_1 + x^3 (2 a_1^2 - a_2) + x^4 (-5 a_1^3 + 5 a_1 a_2 - a_3) + x^5 (14 a_1^4 - 21 a_1^2 a_2 + 3 a_2^2 + 6 a_1 a_3 - a_4) + x^6 (-42 a_1^5 + 84 a_1^3 a_2 - 28 a_1 a_2^2 - 28 a_1^2 a_3 + 7 a_2 a_3 + 7 a_1 a_4 - a_5)$

$y = x - a_1 y^2 - a_2 y^3 + \ldots$

```
In[12]:= Y = Nest[(x - Sum[a_i #^(i+1), {i, 1, 5}]) &, O[x], 6] // ExpandAll

         Y - FixedPoint[ExpandAll[x - Sum[a_i #^(i+1), {i, 1, 6}] + O[x]^7] &, 0]

         Y - InverseSeries[y + Sum[a_i y^(i+1), {i, 1, 5}] + O[y]^7, x]
```

Out[12]= $x - a_1 x^2 + (2 a_1^2 - a_2) x^3 + (-5 a_1^3 + 5 a_1 a_2 - a_3) x^4 + (14 a_1^4 - 21 a_1^2 a_2 + 3 a_2^2 + 6 a_1 a_3 - a_4) x^5 + (-42 a_1^5 + 84 a_1^3 a_2 - 28 a_1 a_2^2 - 28 a_1^2 a_3 + 7 a_2 a_3 + 7 a_1 a_4 - a_5) x^6 + O[x]^7$

Out[13]= $O[x]^7$

Out[14]= $O[x]^7$