Kurs ▮ Cours

8. Datentypen
▮ Types de données

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 8 lesen!
Run mit WIN+*Mathematica* Version 5.2

▮ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 8.
Testé avec *Mathematica* version 5.2+WIN

Mathematica kennt keine Deklarationen von Typen für die Variablen wie z.B. in klassischen Programmiersprachen, z.B. Modula II. Trotzdem.... *Mathematica* kann selber merken, um welchen Typ es sich handelt...
▮ *Mathematica* ne connaît pas de déclarations de types pour les variables comme p.ex. dans des langues de programmation classiques, p.ex. Modula II. Pourtant.... *Mathematica* s'aperçoit de quel type il s'agit....

8.1. Atomare Typen
▮ Types atomiques

8.1.1. Rufe die Information selbst ab
▮ Appelle l'information toi-même

Jeder Ausdruck ist aus folgenden "atomaren Typen" zusammengesetzt: Integer, Real, Rational, Complex, Symbol, String.
▮ Chaque expression est composée des "types atomiques" suivants: Integer, Real, Rational, Complex, String.

```
In[1]:= ??Integer

        Integer is the head used for integers. Mehr…

        Attributes[Integer] = {Protected}
```

In[2]:= **?Real**

Real is the head used for real (floating-point) numbers. Mehr...

In[3]:= **?Rational**

Rational is the head used for rational numbers. Mehr...

In[4]:= **?Complex**

Complex is the head used for complex numbers. Mehr...

In[5]:= **?Symbol**

Symbol["name"] refers to a symbol with the specified name. Mehr...

In[6]:= **?String**

String is the head of a character string "text". Mehr...

8.1.2. Probiere aus:
Essaie:

In[7]:= **?Head**

Head[expr] gives the head of expr. Mehr...

In[8]:= **Head[3452]**

Out[8]= Integer

In[9]:= **Head[67.94]**

Out[9]= Real

In[10]:= **Head[3 + 4 I]**

Out[10]= Complex

In[11]:= **Head[E]**

Out[11]= Symbol

In[12]:= **Head[Pi]**

Out[12]= Symbol

In[13]:= **Head["Tschou zämä!"]**

Out[13]= String

8.2. Andere Typen und Prädikatenfunktionen
▮ Autres types et fonctions de prédicats

8.2.1. Beispiele
▮ Exemples

Neben den atomaren Typen gibt es übergeordnete Typen. Z.B. Integer, Real,
Rational u.s.w. sind Zahlen. Ein Vektor ist eine Liste etc.. Um zu prüfen, um welchen Typ es sich gerade handelt, existieren sogenannte Prädikatenfunktionen. Beispiele:

▮ A part des types atomiques il y a des types supérieurs. P.ex. Integer, Real, Rational etc. sont des nombres. Un vecteur est une liste etc.. Pour examiner de quel type il s'agit, il existe les dites fonctions de prédicats.

```
In[14]:= NumberQ[5/2 + 8 I/5]
Out[14]= True

In[15]:= NumberQ[5/2 + 8 I/5 - Pi]
Out[15]= False

In[16]:= VectorQ[{a, b, c, d, e, f, g}]
Out[16]= True
```

8.2.2. Uebersicht
▮ Vue d'ensemble

Die Prädikatenfunktionen enden mit "Q". Ihr Wertebereich ist {True, False}. Hier sind sie alle:

▮ Les fonctions de prédicats se terminent par "Q". Leur domaine de valeurs est "True, False". Voici tous:

In[17]:= **?*Q**

System`

```
ArgumentCountQ       LinkConnectedQ  PartitionsQ
ArrayQ               LinkReadyQ      PolynomialQ
AtomQ                ListQ           PrimeQ
DigitQ               LowerCaseQ      SameQ
EllipticNomeQ        MachineNumberQ  StringFreeQ
EvenQ                MatchLocalNameQ StringMatchQ
ExactNumberQ         MatchQ          StringQ
FreeQ                MatrixQ         SyntaxQ
HypergeometricPFQ    MemberQ         TensorQ
InexactNumberQ       NameQ           TrueQ
IntegerQ             NumberQ         UnsameQ
IntervalMemberQ      NumericQ        UpperCaseQ
InverseEllipticNomeQ OddQ            ValueQ
LegendreQ            OptionQ         VectorQ
LetterQ              OrderedQ
```

8.3. Interne Darstellung in *Mathematica*
▮ Représentation intérieure dans *Mathematica*

8.3.1. Allgemeines
▮ Généralités

Operationszeichen oder spezielle Symbole wie "+", "/", "#", "/@" u.s.w. werden intern dargestellt durch Funktionen. Mit den Funktionen "FullForm[...]", "FullForm[Hold[...]]" oder "TreeForm" kann die interne Darstellung in *Mathematica* sichtbar gemacht werden.

▮ Signes d'opérations ou symboles spéciaux tels que "+", "/", "#", "/@" etc. sont représentés intérieurement par des fonctions. Par les fonctions "FullForm[...]", "FullForm[Hold[...]]" ou "TreeForm" on peut rendre visible la représentation intérieure dans *Mathematica*.

In[18]:= **?FullForm**

FullForm[expr] prints as the full form of expr, with no special syntax. Mehr…

In[19]:= **?TreeForm**

TreeForm[expr] prints with different levels in expr shown at different depths. Mehr…

Beispiele: ▮ Exemples:

In[20]:= **FullForm[4/7]**

Out[20]//FullForm=

```
Rational[4, 7]
```

```
In[21]:= FullForm[4 - 7 I]

Out[21]//FullForm=
        Complex[4, -7]

In[22]:= FullForm[f[x]]

Out[22]//FullForm=
        f[x]

In[23]:= FullForm[x y^z / w]

Out[23]//FullForm=
        Times[Power[w, -1], x, Power[y, z]]

In[24]:= TreeForm[x y^z / w]

Out[24]//TreeForm=
        Times[ |            , x, |          ]
              Power[w, -1]     Power[y, z]

In[25]:= FullForm[2 + 3 15]

Out[25]//FullForm=
        47
```

8.3.2. Erzwungene oder verhinderte Ausführung
▮ Exécution obtenue de force ou empêchée

Im letzten Beispiel ist zuerst die Rechnung ausgeführt worden. Erst darauf wurde dann die Funktion "FullForm" angewandt. Die gültige Abarbeitungshierarchie kann aber geändert werden. Dazu dienen folgende Befehle:
▮ Dans le dernier exemple on a d'abord exécuté le calcul. Ensuite seulement on a appliqué la fonction "FullForm". On peut changer la hiérarchie de l'exécution valable par les ordres suivants:

```
In[26]:= ?Hold

        Hold[expr] maintains expr in an unevaluated form. Mehr…

In[27]:= ?HoldForm

        HoldForm[expr] prints as the expression expr, with expr maintained in an unevaluated form.
         Mehr…

In[28]:= ?ReleaseHold

        ReleaseHold[expr] removes Hold, HoldForm, HoldPattern and HoldComplete in expr. Mehr…

In[29]:= ?Evaluate

        Evaluate[expr] causes expr to be evaluated even if it appears as the argument of
           a function whose attributes specify that it should be held unevaluated. Mehr…
```

Beispiele: ▮ Exemples

```
In[30]:= FullForm[Hold[2 + 7 - 4]]

Out[30]//FullForm=
        Hold[Plus[2, 7, -4]]
```

In[31]:=
```
Hold[2 + 7 - 4]
```

Out[31]=
```
Hold[2 + 7 - 4]
```

In[32]:=
```
HoldForm[2 + 7 - 4]
```

Out[32]=
```
2 + 7 - 4
```

In[33]:=
```
ReleaseHold[Hold[2 + 7 - 4]]
```

Out[33]=
```
5
```

In[34]:=
```
meineListe = {"Or", "And", "Abs"};
Attributes[meineListe]
```

Out[35]=
```
{}
```

In[36]:=
```
Attributes[Evaluate[meineListe]]
```

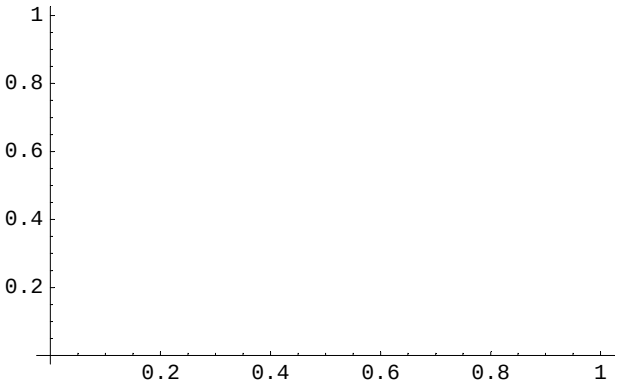
Out[36]=
```
{{Flat, HoldAll, OneIdentity, Protected},
 {Flat, HoldAll, OneIdentity, Protected}, {Listable, NumericFunction, Protected}}
```

In[37]:=
```
t[x_] = Table[x^n, {n,4}]
```

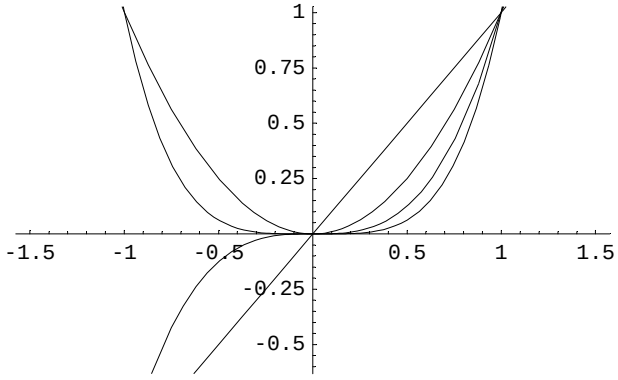
Out[37]=
```
{x, x^2, x^3, x^4}
```

In[38]:=
```
Plot[t[x], {x, -1.5, 1.5}];
```

```
Plot::plnr : t[x] is not a machine-size real number at x = -1.5. Mehr…

Plot::plnr : t[x] is not a machine-size real number at x = -1.3783. Mehr…

Plot::plnr : t[x] is not a machine-size real number at x = -1.24557. Mehr…

General::stop : Further output of Plot::plnr will be suppressed during this calculation. Mehr…
```



In[39]:= **Plot[Evaluate[t[x]], {x, -1.5, 1.5}];**



Was ist passiert?

■ Que s'est-il passé?

8.3.3. Der "Kopf" eines Ausdrucks und "FullForm"
■ La "tête" d'une expression et "FullForm"

Vergleiche: ■ Comparer:

In[40]:= **Head[x + y]**

Out[40]= Plus

In[41]:= **FullForm[x + y]**

Out[41]//FullForm=
```
Plus[x, y]
```

In[42]:= **Head[FullForm[x + y]]**

Out[42]= FullForm

In[43]:= **FullForm[{x, y}]**

Out[43]//FullForm=
```
List[x, y]
```

In[44]:= **Head[{x, y}]**

Out[44]= List

In[45]:= **FullForm[Cos[x^3 - 4 x^2]]**

Out[45]//FullForm=
```
Cos[Plus[Times[4, Power[x, 2]], Times[-1, Power[x, 3]]]]
```

In[46]:= **Head[Cos[x^3 - 4 x^2]]**

Out[46]= Cos

8.3.4. Veränderung des "Kopfs"
▮ Changement de la "tête"

Vergleiche: ▮ Comparer:

```
In[47]:= FullForm[{x, y, z, w}]

Out[47]//FullForm=
        List[x, y, z, w]

In[48]:= FullForm[x + y + z + w]

Out[48]//FullForm=
        Plus[w, x, y, z]

In[49]:= {x, y, z, w} /. List -> Plus

Out[49]= w + x + y + z
```

Was ist passiert? Der "Kopf" ist ausgewechselt worden! Doch *Mathematica* kann noch mehr: Mit "Apply" lässt sich ein weiterer "Kopf" einem Ausdruck voranstellen:
▮ Que s'est-il passé? On a remplacé la "tête"! *Mathematica* peut cependant davantage. Pour "Apply" on peut placer devant une expression une autre "tête" encore.

```
In[50]:= FullForm[{x, y, z, w}]

Out[50]//FullForm=
        List[x, y, z, w]

In[51]:= summe = Apply[Plus, {x, y, z, w}]

Out[51]= w + x + y + z

In[52]:= FullForm[summe]

Out[52]//FullForm=
        Plus[w, x, y, z]

In[53]:= Head[summe]

Out[53]= Plus
```

Oder so: ▮ Ou ainsi:

```
In[54]:= Apply[Plus, Range[1000]]

Out[54]= 500500
```

Das ist die Summe der natürlichen Zahlen von 1 bis 1000.
▮ Voici la somme des nombres naturels de 1 à 1000.

8.4. Herauslesen von gewissen Teilen von Ausdrücken
▮ Choisir certaines parties d'expressions

Allgemeines
▮ Généralités

Studiere: ▮ Etudier:

```
In[55]:= ?Position
```

```
Position[expr, pattern] gives a list of the positions at which objects
   matching pattern appear in expr. Position[expr, pattern, levspec] finds
   only objects that appear on levels specified by levspec. Position[expr,
   pattern, levspec, n] gives the positions of the first n objects found. Mehr...
```

```
In[56]:= ?[[
```

```
expr[[i]] or Part[expr, i] gives the ith part of expr. expr[[-i]]
   counts from the end. expr[[0]] gives the head of expr. expr[[i, j, ... ]]
   or Part[expr, i, j, ... ] is equivalent to expr[[i]] [[j]] ... . expr[[
   {i1, i2, ... } ]] gives a list of the parts i1, i2, ...  of expr. Mehr...
```

```
In[57]:= ?Part
```

```
expr[[i]] or Part[expr, i] gives the ith part of expr. expr[[-i]]
   counts from the end. expr[[0]] gives the head of expr. expr[[i, j, ... ]]
   or Part[expr, i, j, ... ] is equivalent to expr[[i]] [[j]] ... . expr[[
   {i1, i2, ... } ]] gives a list of the parts i1, i2, ...  of expr. Mehr...
```

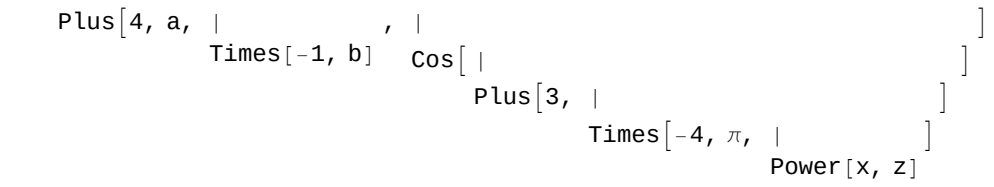
```
In[58]:= neueListe = 2 Range[9]

Out[58]= {2, 4, 6, 8, 10, 12, 14, 16, 18}

In[59]:= neueListe[[4]]

Out[59]= 8

In[60]:= Head[neueListe[[4]]]

Out[60]= Integer

In[61]:= HoldForm[neueListe[[4]]]

Out[61]= neueListe〚4〛
```

Bearbeite: ▮ Travailler:

```
In[62]:= ausdruck = 4 + a - b + Cos[4 Pi x^z - 3]

Out[62]= 4 + a - b + Cos[3 - 4 π x^z]
```

In[63]:= **TreeForm[ausdruck]**

Out[63]//TreeForm=



In[64]:= **ausdruck[[3]]**

Out[64]= $-b$

In[65]:= **ausdruck[[4]]**

Out[65]= $\text{Cos}[3 - 4\,\pi\,x^z]$

In[66]:= **ausdruck[[4, 1]]**

Out[66]= $3 - 4\,\pi\,x^z$

In[67]:= **ausdruck[[4, 1, 2]]**

Out[67]= $-4\,\pi\,x^z$

Was ist passiert? --- Studiere weiter die Funktion "Position":
■ Que s'est-il passé? --- Continuer d'étudier la fonction "Position":

In[68]:= **ausdruck1 = Expand[(r + s + w)^3]**

Out[68]= $r^3 + 3\,r^2\,s + 3\,r\,s^2 + s^3 + 3\,r^2\,w + 6\,r\,s\,w + 3\,s^2\,w + 3\,r\,w^2 + 3\,s\,w^2 + w^3$

In[69]:= **Position[ausdruck1,r]**

Out[69]= {{1, 1}, {2, 2, 1}, {3, 2}, {5, 2, 1}, {6, 2}, {8, 2}}

In[70]:= **Position[ausdruck1,s]**

Out[70]= {{2, 3}, {3, 3, 1}, {4, 1}, {6, 3}, {7, 2, 1}, {9, 2}}

Deute diesen Output! --- Damit kann man auch komponieren:
Interpréter cet output! --- On peut aussi en faire des compositions:

In[71]:=
```
u = {ausdruck1[[2]], ausdruck1[[2,2]],
     ausdruck1[[2,2,1]], ausdruck1[[5,2]]}
```

Out[71]= $\{3\,r^2\,s,\ r^2,\ r,\ r^2\}$

In[72]:= **v = Apply[Plus, u]**

Out[72]= $r + 2\,r^2 + 3\,r^2\,s$

In[73]:= **v /. r->3**

Out[73]= $21 + 27\,s$

8.5. Symbole wie "unendlich" etc.
▮ Symboles tels que "infini" etc.

Allgemeines
▮ Généralités

Studiere: ▮ Etudier

In[74]:= **?Infinity**

Infinity is a symbol that represents a positive infinite quantity. Mehr...

In[75]:= **Head[Infinity]**

Out[75]= DirectedInfinity

In[76]:= **1 a**

Out[76]= a

In[77]:= **0 a**

Out[77]= 0

In[78]:= **0 5**

Out[78]= 0

In[79]:= **0 Infinity**

∞::indet : Indeterminate expression 0 ∞ encountered. Mehr…

Out[79]= Indeterminate

In[80]:= **?Limit**

Limit[expr, x->x0] finds the limiting value of expr when x approaches x0. Mehr...

In[81]:= **Limit[1/x, x->0]**

Out[81]= ∞

In[82]:= **Limit[Log[x], x->0]**

Out[82]= -∞

In[83]:= **Head[-Infinity]**

Out[83]= DirectedInfinity

In[84]:= **FullForm[-Infinity]**

Out[84]//FullForm=

DirectedInfinity[-1]

```
In[85]:= Limit[Log[1/x], x->0]
Out[85]= ∞

In[86]:= -5 Infinity
Out[86]= -∞

In[87]:= 7 / 0
        Power::infy : Infinite expression 1/0 encountered. Mehr…
Out[87]= ComplexInfinity

In[88]:= ?ComplexInfinity
        ComplexInfinity represents a quantity
           with infinite magnitude, but undetermined complex phase. Mehr…

In[89]:= 0 / 0
        Power::infy : Infinite expression 1/0 encountered. Mehr…

        ∞::indet : Indeterminate expression 0 ComplexInfinity encountered. Mehr…
Out[89]= Indeterminate

In[90]:= ?Indeterminate
        Indeterminate is a symbol that represents a
           numerical quantity whose magnitude cannot be determined. Mehr…
```

8.6. Memory: Wie *Mathematica* speichert
▮ Memory: La façon de mémoriser de *Mathematica*

8.6.1. Allgemeines
▮ Généralités

Studiere: ▮ Etudier:

```
In[91]:= ?MemoryInUse
        MemoryInUse[ ] gives the number of bytes currently
           being used to store all data in the current Mathematica session. Mehr…
```

Beispiele: ▮ Exemples:

```
In[92]:= MemoryInUse[]
Out[92]= 3398800
```

"Putzmaschine" einsetzen:
▮ Effacer:

```
In[93]:= Remove["Global`@*"]

In[94]:= MemoryInUse[]

Out[94]= 3398832

In[95]:= t = 1;

In[96]:= MemoryInUse[]

Out[96]= 3399616
```

Was hat t sowie die ausgeführten Operationen "gekostet"?
■ Qu'est-ce que t ainsi que les opérations exécutées ont-ils "coûté"?

```
In[97]:= vorher = MemoryInUse[]

Out[97]= 3400160

In[98]:= Table[Random[],{10000}];

In[99]:= nachher = MemoryInUse[]

Out[99]= 3482472

In[100]:=
       N[nachher - vorher] / 1000

Out[100]=
        82.312
```

Wieviel Speicher hat *Mathematica* "verbraten"?
■ Combien de mémoire *Mathematica* a-t-il "gaspillé"?

8.6.2. Gemeinsame Speichernutzung (sharing)
■ Utilisation commune de la mémoire (sharing)

Studiere: ■ Etudier:

```
In[101]:=
       (   Clear[a, b];
           vorher1 = MemoryInUse[];
           a = Table[Random[],{10000}];
           nachher1 = MemoryInUse[];
           b = a;
           nachher2 = MemoryInUse[];
           {nachher1 - vorher1, nachher2- nachher1}
       )

Out[101]=
        {80128, 16}
```

Beachte: b hat keinen Speicher mehr belegt. b belegt erst dann Speicher, wenn a manipuliert wird. Offenbar gibt es von a und b aus Zeiger auf diesselbe Speicheradresse. Das kann mit der Funktion "Share" auch künstlich bewirkt werden. Im folgenden Beispiel werden 1000 Pseudozufallszahlen im Intervall zwischen 10'000 und 10'100 gerechnet. Da dieses Intervall nur 100 Zahlen enthält, gibt es sicher Duplikate. Mit "Share" kann dabei eine Mehrfachspeicherung vermieden werden. Probiere:

■ Retenir: b n'a pas plus occupé de mémoire. b occupe la mémoire seulement si l'on manipule a. Il est évident qu'il a de a et de b des indicateurs pour la même adresse dans la mémoire. On peut obtenir cela aussi artificiellement par la fonction "Share". Dans l'exemple suivant on calcule 1'000 nombres pseudo-aléatoires dans l'intervalle entre 10'000 et 10'100. Comme cet intervalle ne compte que 100 nombres, il y aura certainement des doubles. Par "Share" on peut éviter une mise en mémoire répétitive. Essaie:

In[102]:=
```
?Share
```

```
Share[expr] changes the way expr is stored internally, to try and minimize the amount of
   memory used. Share[ ] tries to minimize the memory used to store all expressions. Mehr…
```

In[103]:=
```
(   Clear[c];
    vorherSpeicher = MemoryInUse[];
    c = Table[Random[Integer, {10000, 10100}],
                    {10000}];
    nachherSpeicher = MemoryInUse[];
    c = Share[c];
    nachherShare = MemoryInUse[];
    {nachherSpeicher - vorherSpeicher,
     nachherShare- vorherSpeicher}
)
```

Out[103]=
```
{40464, 1416}
```

Probiere: ■ Essaie:

In[104]:=
```
?ByteCount
```

```
ByteCount[expr] gives the number of bytes used internally by Mathematica to store expr.
  Mehr…
```

In[105]:=
```
t = 2
```

Out[105]=
```
2
```

In[106]:=
```
ByteCount[t]
```

Out[106]=
```
16
```

In[107]:=
```
ByteCount[neueVariable]
```

Out[107]=
```
0
```

In[108]:=
```
neueVariable = 1;
```

In[109]:=
```
ByteCount[neueVariable]
```

Out[109]=
```
16
```

In[110]:=
```
ByteCount[a]
```
Out[110]=
```
80056
```

In[111]:=
```
ByteCount[{a, b}]
```
Out[111]=
```
160144
```

Beispiele für Speicherbelegung:

■ Exemple pour l'occupation de mémoire:

In[112]:=
```
z = 15;  {Head[z], ByteCount[z]}
```
Out[112]=
```
{Integer, 16}
```

In[113]:=
```
z = 1;  {Head[z], ByteCount[z]}
```
Out[113]=
```
{Integer, 16}
```

In[114]:=
```
z = 100;  {Head[z], ByteCount[z]}
```
Out[114]=
```
{Integer, 16}
```

In[115]:=
```
z = 1000000000000000;  {Head[z], ByteCount[z]}
```
Out[115]=
```
{Integer, 32}
```

In[116]:=
```
z = 3.1;  {Head[z], ByteCount[z]}
```
Out[116]=
```
{Real, 16}
```

In[117]:=
```
z = 3.14159666666666;  {Head[z], ByteCount[z]}
```
Out[117]=
```
{Real, 16}
```

In[118]:=
```
z = 3/8;  {Head[z], ByteCount[z]}
```
Out[118]=
```
{Rational, 64}
```

In[119]:=
```
z = 3 + 5 I;  {Head[z], ByteCount[z]}
```
Out[119]=
```
{Complex, 64}
```

In[120]:=
```
z = Range[10];  {Head[z], ByteCount[z]}
```
Out[120]=
```
{List, 96}
```

In[121]:=
```
z = Range[10000];  {Head[z], ByteCount[z]}
```
Out[121]=
```
{List, 40056}
```

In[122]:=
```
z = "tschou";  {Head[z], ByteCount[z]}
```
Out[122]=
```
{String, 24}
```

In[123]:=
```
z = "t";  {Head[z], ByteCount[z]}
```
Out[123]=
```
{String, 24}
```

In[124]:=
```
z = "tschou zaeaeaeaeaeaemmaeaeaeaeaeaeaeae";
     {Head[z], ByteCount[z]}
```
Out[125]=
```
{String, 56}
```

"Putzmaschine" einsetzen
▮ Employer la "machine de nettoyage"

In[126]:=
```
(* Old Form: Remove["Global`@*"] *)
```

In[127]:=
```
Remove["Global`*"]
```