Institut für Theoretische
Physik
Universität zu Köln

Prof. Dr. Simon Trebst

Computerphysik
Vorlesung — Programmiertechniken 2

Sommersemester 2019

Website: http://www.thp.uni-koeln.de/trebst/Lectures/2019-CompPhys.shtml (http://www.thp.uni-koeln.de/trebst/Lectures/2019-CompPhys.shtml)

0. Erinnerung letzte Woche

In [1]:
```
a=4711
```

Out[1]:
```
4711
```

In [2]:
```
sqrt(a)
```

Out[2]:
```
68.63672486358888
```

In [3]:
```
68.63^2
```

Out[3]:
```
4710.076899999999
```

In [4]:
```
1//3+1//6
```

Out[4]:
```
1//2
```

In [5]:
```
s="Hello world!"
```

Out[5]:
```
"Hello world!"
```

In [6]:
```
println(s, "\t", a, "\t", 1//3+1//6)
```

```
Hello world!	4711	1//2
```

1. Schleifen – Beispiele

Potenzen

In [8]:
```
pot=1
for i in 1:10
    pot = pot*2
    println(i, "\t", pot)
end
```

```
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
```

Gauss'sche Summe

In [9]:
```
sum = 0
for i in 1:100
    sum = sum + i
end
println("sum = ", sum)
```

```
sum = 5050
```

Fakultät

In [19]:
```
fac = Int128(1); i=1; n=23;
```

In [20]:
```
while(i<=n)
    fac = fac * i
    i = i + 1
end
println(fac)
```

```
25852016738884976640000
```

Fibonacci Zahlen

In [14]:
```
ϕ=(sqrt(5)+1)/2
```

Out[14]: 1.618033988749895

In [29]:
```
fib = 2
```

Out[29]: 2

In [30]:
```
f=1
while (f<20)
    fib = fib * ϕ
    fib = round(fib)
    println(fib)
    f=f+1
end
```

```
3.0
5.0
8.0
13.0
21.0
34.0
55.0
89.0
144.0
233.0
377.0
610.0
987.0
1597.0
2584.0
4181.0
6765.0
10946.0
17711.0
```

2. Verzweigungen

In [42]:
```
v=5
```

Out[42]: 5

In [38]:
```
if v>5
    println("Die Variable ist grösser als 5.")
end
```

```
Die Variable ist grösser als 5.
```

In [43]:
```
if v>5
    println("Die Variable ist grösser als 5.")
else
    println("Die Variable ist kleiner/gleich als 5.")
end
```

```
Die Variable ist kleiner/gleich als 5.
```

In [47]:
```
v=5
if v>10
    println("Die Variable ist grösser als 10.")
elseif v<5
    println("Die Variable ist kleiner als 5.")
else
    println("Die Variable ist irgendwo dazwischen.")
end
```

```
Die Variable ist irgendwo dazwischen.
```

In [58]:
```
a=1; b=3;
```

In [61]:
```
z = a<b ? a : b;
println("z = ", z)
```

```
z = 1
```

3. Variablen (cont'd)

Arrays

In [65]:
```
a = [10, 20, 30, 40]
```

Out[65]:
```
4-element Array{Int64,1}:
 10
 20
 30
 40
```

In [66]:
```
a[1]
```

Out[66]:
```
10
```

In [67]:
```
a[3]
```

Out[67]:
```
30
```

In [68]:
```
a[0]
```

```
BoundsError: attempt to access 4-element Array{Int64,1} at index [0]

Stacktrace:
 [1] getindex(::Array{Int64,1}, ::Int64) at ./array.jl:731
 [2] top-level scope at In[68]:1
```

In [69]:
```
a[5]
```

```
BoundsError: attempt to access 4-element Array{Int64,1} at index [5]

Stacktrace:
 [1] getindex(::Array{Int64,1}, ::Int64) at ./array.jl:731
 [2] top-level scope at In[69]:1
```

In [70]:
```
b = ["hello", "world", 4711]
```

Out[70]:
```
3-element Array{Any,1}:
     "hello"
     "world"
 4711
```

In [71]:
```
b[2]
```

Out[71]:
```
"world"
```

In [72]:
```
b[3]+12
```

Out[72]:
```
4723
```

In [73]:
```
c = collect(1:10)
```

Out[73]:
```
10-element Array{Int64,1}:
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
```

In [74]:
```
d = collect(10:3:30)
```

Out[74]:
```
7-element Array{Int64,1}:
 10
 13
 16
 19
 22
 25
 28
```

In [75]:
```
e = zeros(5)
```

Out[75]:
```
5-element Array{Float64,1}:
 0.0
 0.0
 0.0
 0.0
 0.0
```

In [76]:
```
r = rand(5)
```

Out[76]:
```
5-element Array{Float64,1}:
 0.33363217586041594
 0.6489009647418249
 0.7420766562147945
 0.5660112605709471
 0.35818710736230774
```

In [82]:
```
g = rand(1:15,5)
```

Out[82]:
```
5-element Array{Int64,1}:
 14
 10
 13
  9
  1
```

In [83]:
```
h = collect(1:4)
```

Out[83]:
```
4-element Array{Int64,1}:
 1
 2
 3
 4
```

In [84]:
```
push!(h,5)
```

Out[84]:
```
5-element Array{Int64,1}:
 1
 2
 3
 4
 5
```

In [85]:
```
push!(h,67)
```

Out[85]:
```
6-element Array{Int64,1}:
  1
  2
  3
  4
  5
 67
```

In [87]:
```
h
```

Out[87]:
```
6-element Array{Int64,1}:
  1
  2
  3
  4
  5
 67
```

In [88]:
```
h[2]
```

Out[88]:
```
2
```

In [89]:
```
h[2]=333
```

Out[89]:
```
333
```

In [90]:
```
h
```

Out[90]:
```
6-element Array{Int64,1}:
   1
 333
   3
   4
   5
  67
```

In [94]:
```
size(h)
```

Out[94]:
```
(6,)
```

In [92]:
```
ndims(h)
```

Out[92]:
```
1
```

Mehr-dimensionale Arrays

In [95]:
```
M = [1 2 3; 4 5 6; 7 8 9]
```

Out[95]:
```
3×3 Array{Int64,2}:
 1  2  3
 4  5  6
 7  8  9
```

In [96]:
```
N = [1 2 3; 4 5 6]
```

Out[96]:
```
2×3 Array{Int64,2}:
 1  2  3
 4  5  6
```

In [100]:
```
size(M)
```

Out[100]:
```
(3, 3)
```

In [104]:
```
O=rand(3,3)
```

Out[104]:
```
3×3 Array{Float64,2}:
 0.899745  0.232449  0.251345
 0.248355  0.645398  0.15412
 0.276952  0.258471  0.63812
```

In [105]:
```
P = zeros(3,3)
```

Out[105]:
```
3×3 Array{Float64,2}:
 0.0  0.0  0.0
 0.0  0.0  0.0
 0.0  0.0  0.0
```

In [106]:
```
M + O
```

Out[106]:
```
3×3 Array{Float64,2}:
 1.89974  2.23245  3.25134
 4.24835  5.6454   6.15412
 7.27695  8.25847  9.63812
```

In [108]:
```
N = N .+ 0.2
```

Out[108]:
```
2×3 Array{Float64,2}:
 1.2  2.2  3.2
 4.2  5.2  6.2
```

In [109]:
```
N .* 2
```

Out[109]:
```
2×3 Array{Float64,2}:
 2.4   4.4   6.4
 8.4  10.4  12.4
```

4. Funktionen & Plots

Funktionen sind Programmabschnitte, die eine wohldefinierte Aufgabe übernehmen. Mit ihrer Hilfe kann ein Programm übersichtlicher gestaltet werden und bestimmte Teile für spätere Aufgaben wiederverwendet werden. Man kann je nach Bedarf Variablen übergeben, bearbeiten und auch wiederzurückgeben.

Die einfachste Art, eine mathematische Funktion zu definieren ist über eine direkte Anweisung, wie etwa

In [134]:
```
fun(x) = x^2 + x-2.0
```

Out[134]:
```
fun (generic function with 1 method)
```

In [135]:
```
fun(4)
```

Out[135]:
```
18.0
```

In [136]:
```
fun(-2)
```

Out[136]:
```
0.0
```

In [137]:
```
x_values = range(-5, stop=5, length=201)
```

Out[137]:
```
-5.0:0.05:5.0
```

In [139]:
```
f_values = fun.(x_values);
```

Plots

Zunächst sollten wir das Paket "PyPlot" installieren.

] add PyPlot

In [140]:
```
] add PyPlot
```

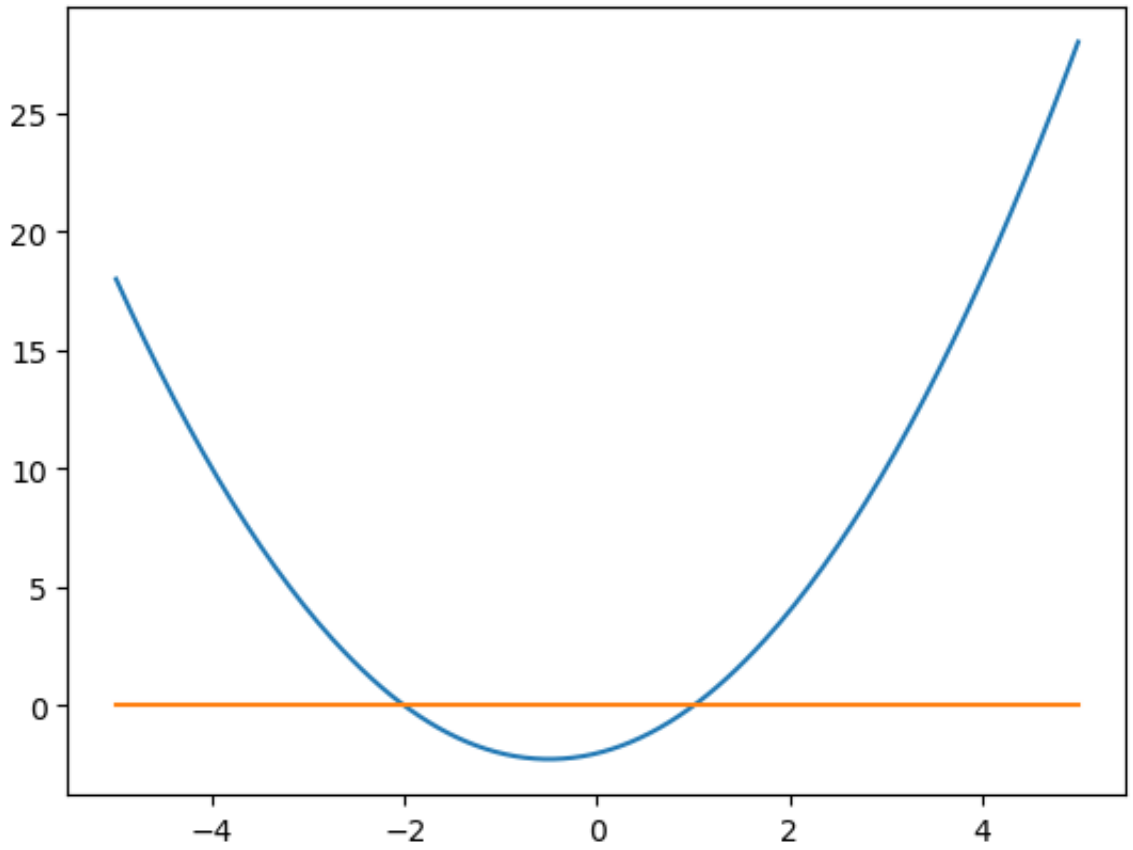
```
  Updating registry at `~/.julia/registries/General`
  Updating git-repo `https://github.com/JuliaRegistries/General.git`
 Resolving package versions...
  Updating `~/.julia/environments/v1.0/Project.toml`
[no changes]
  Updating `~/.julia/environments/v1.0/Manifest.toml`
[no changes]
```

In [141]:
```
using PyPlot
```

```
┌ Info: Precompiling PyPlot [d330b81b-6aea-500a-939a-2ce795aea3ee]
└ @ Base loading.jl:1186
┌ Info: Installing matplotlib via the Conda matplotlib package...
└ @ PyCall /Users/cp2019/.julia/packages/PyCall/a5Jd3/src/PyCall.jl:70
5
┌ Info: Running `conda install -y matplotlib` in root environment
└ @ Conda /Users/cp2019/.julia/packages/Conda/CpuvI/src/Conda.jl:112

Collecting package metadata: ...working... done
Solving environment: ...working... done
```

```
## Package Plan ##

  environment location: /Users/cp2019/.julia/conda/3

  added / updated specs:
    - matplotlib


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    cycler-0.10.0              |           py37_0          14 KB
    freetype-2.9.1             |       hb4e5f40_0         864 KB
    kiwisolver-1.0.1           |   py37h0a44026_0          56 KB
    matplotlib-3.0.3           |   py37h54f8f79_0         6.6 MB
    pyparsing-2.3.1            |           py37_0         105 KB
    pytz-2018.9                |           py37_0         268 KB
    ------------------------------------------------------------
                                           Total:         7.9 MB

The following NEW packages will be INSTALLED:

  cycler             pkgs/main/osx-64::cycler-0.10.0-py37_0
  freetype           pkgs/main/osx-64::freetype-2.9.1-hb4e5f40_0
  kiwisolver         pkgs/main/osx-64::kiwisolver-1.0.1-py37h0a44026_0
  matplotlib         pkgs/main/osx-64::matplotlib-3.0.3-py37h54f8f79_0
  pyparsing          pkgs/main/osx-64::pyparsing-2.3.1-py37_0
  pytz               pkgs/main/osx-64::pytz-2018.9-py37_0


Downloading and Extracting Packages
pyparsing-2.3.1      | 105 KB    | ########## | 100%
pytz-2018.9          | 268 KB    | ########## | 100%
freetype-2.9.1       | 864 KB    | ########## | 100%
kiwisolver-1.0.1     | 56 KB     | ########## | 100%
cycler-0.10.0        | 14 KB     | ########## | 100%
matplotlib-3.0.3     | 6.6 MB    | ########## | 100%
Preparing transaction: ...working... done
Verifying transaction: ...working... done
Executing transaction: ...working... done

WARNING: The conda.compat module is deprecated and will be removed in
a future release.
```

In [143]:
```
plot(x_values, f_values)
plot(x_values, zeros(201))
```



Out[143]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x1468d0d68>
```

In [144]:
```
fun2(x) = exp(x)*log(x) - x^2
```

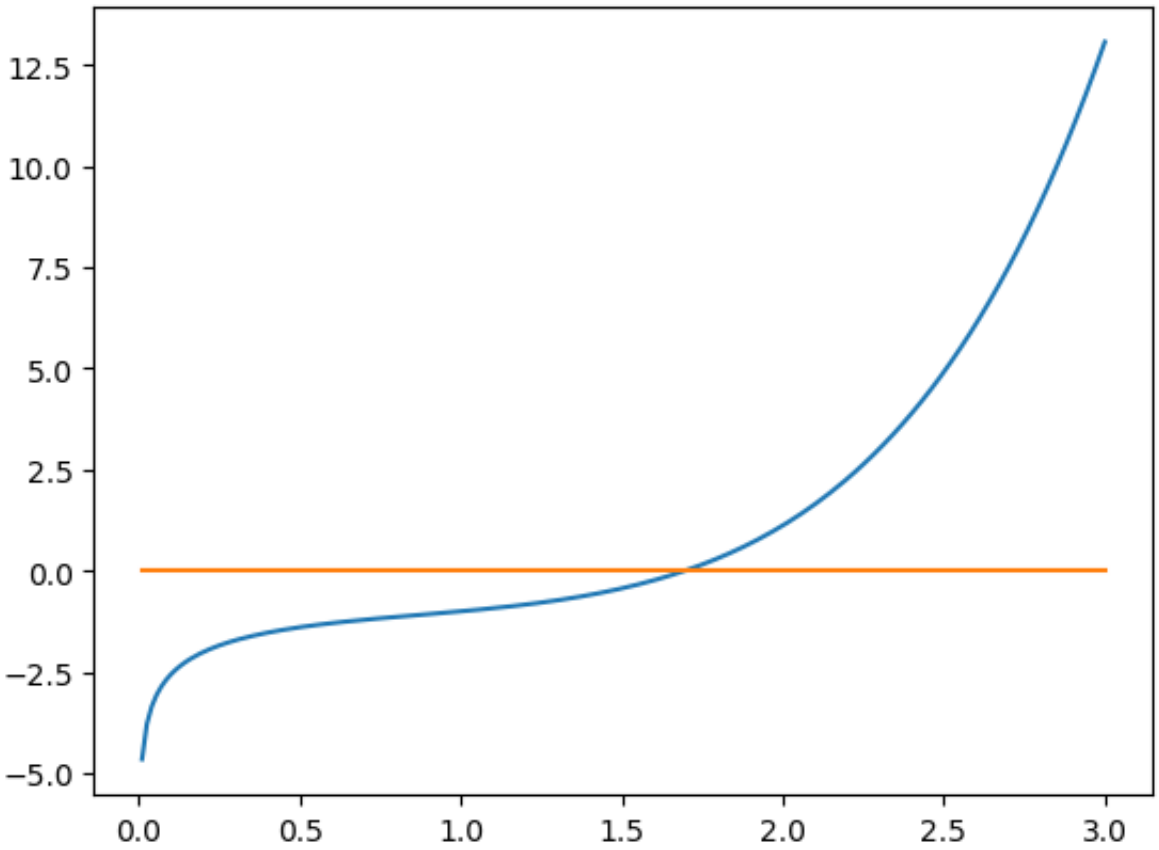
Out[144]:
```
fun2 (generic function with 1 method)
```

In [145]:
```
x_values = range(0.01, stop=3, length=201)
```

Out[145]:
```
0.01:0.01495:3.0
```

In [146]:
```
f_values = fun2.(x_values);
```

In [147]:
```
plot(x_values, f_values)
plot(x_values, zeros(201))
```



Out[147]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x12ca88cf8>
```

In [ ]: