Computerphysik Programmiertutorial 7a

Prof. Dr. Matteo Rizzi und Dr. Markus Schmitt - Institut für Theoretische Physik, Universität zu Köln

Github: https://github.com/markusschmitt/compphys2022

Inhalt dieses Notebooks: Geltungsbereiche von Variablen

Geltungsbereiche von Variablen (Scope)

Der Geltungsbereich einer Variable bestimmt in welchen Teilen des Programmcodes die Variable bekannt ist. Geltungsbereiche entsprechen immer Code-Blöcken wie z.B. Funktionen oder Schleifen.

Generell werden globale und lokale Variablen unterschieden. In Jupyter notebooks sind Variablen, die außerhalb von Code-Blöcken deklariert werden, global:

In [1]:
```
x = 7
```

Out[1]:
```
7
```

Globale Variablen sind innerhalb von Code-Blöcken wie Funktionen bekannt:

In [2]:
```
function print_global()
    println(x)
end

print_global()
```

```
7
```

Lokale Variablen, deren Geltungsbereich auf einen Code-Block beschränkt ist, sind außerhalb des Code-Blocks nicht bekannt:

In [3]:
```
function define_local(y)
    my_local = y
end

define_local(3)
println(my_local)
```

```
UndefVarError: my_local not defined

Stacktrace:
 [1] top-level scope
   @ In[3]:6
 [2] eval
   @ ./boot.jl:373 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::String, filename::String)
   @ Base ./loading.jl:1196
```

In [4]:
```
for i in 1:1
    my_local = 3
end

println(my_local)
```

```
UndefVarError: my_local not defined

Stacktrace:
 [1] top-level scope
   @ In[4]:5
 [2] eval
   @ ./boot.jl:373 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::String, filename::String)
   @ Base ./loading.jl:1196
```

Hard vs. soft local

Die lokalen Geltungsbereiche sind weiter unterteilt in *hard local* und *soft local*. Der Unterschied besteht darin wie mit der Zuweisung neuer Werte für globale Variablen umgegangen wird.

Der Geltungsbereich innerhalb von Funktionen ist *hard local*. Das bedeutet, dass bei einer Zuweisung zu einer Variablen `x` immer eine lokale Variable mit diesem Namen angelegt wird - auch wenn bereits eine globale Variable `x` definiert ist.

In [5]:
```
function f(y)
    x=y
    println("x in f: ", x)
end

f(3)
println("x am Ende: ", x)
```

```
x in f: 3
x am Ende: 7
```

Der Geltungsbereich innerhalb von Schleifen ist *soft local*. Das bedeutet, dass bei einer Zuweisung zu einer globalen Variablen `x` der Wert der globalen Variablen geändert wird.

In [6]:
```
for i in 1:1
    x=3
end

println("x am Ende: ", x)
```

```
x am Ende: 3
```

In [7]:
```
sum = 0
for i in 1:20
    sum = sum + i
end

println(sum)
```

```
210
```

Code-Blöcke ohne lokalen Geltungsbereich

Nicht jeder Code-Block definiert einen lokalen Geltungsbereich, z.B. ein `if`-block:

In [8]:
```
b = false

if b
    x = 23
else
    y = 17
end
```

Out[8]:
```
17
```

Mit `b=false` haben wir so eine neue globale Variable `y` deklariert, die auch außerhalb des `if`-Blocks bekann ist:

In [9]:
```
println(y)
```

```
17
```

Zusammenfassung:

Für unsere Zwecke:

- Funktionen - *hard local*
- Schleifen (`for`, `while`) - *soft local*
- `if`-Blöcke - kein eigener Geltungsbereich

Eine Übersicht über Geltungsbereiche von Variablen bei weiteren Strukturen findet sich hier.

`local` und `global` Schlüsselwörter

Um das Standardverhalten von *hard local* und *soft local* zu umgehen, können die Schlüsselwörter `local` und `global` verwendet werden.

So können wir z.B. in einem Schleifenblock eine lokale Variable mit dem Namen einer existierenden globalen Variable deklarieren:

In [10]:
```
x = 7

for i in 1:1
    local x=3
    println("x in Schleife: ", x)
end

println("x am Ende: ", x)
```

```
x in Schleife: 3
x am Ende: 7
```

Oder wir können eine globale Variable innerhalb einer Funktion deklarieren:

In [11]:
```
function define_global(y)
    global my_global = y
end

define_global(3)
my_global
```

Out[11]:
```
3
```