Freie Universität Berlin

**Prof. Dr. Claudia Müller-Birn, Barry Linnert**

# Objektorientierte Programmierung, SoSe 17

## Übung 02

**TutorIn: Thierry Meurers**
**Tutorium 10**

Stefaan Hessmann, Jaap Pedersen, Mark Niehues

9. Mai 2017

---

## 1 Datentypen in Python

$< input >:< value >, < type >$

1. $complex(0) : 0 + 0j, complex$
2. $complex(3) : 3 + 0j, complex$
3. $(1 + 2j) * (3 + 0j) : 3 + 6j, complex$
4. $(2 + 3j)/5j : 0.6 - 0.4j, complex$
5. $() : (), tuple$
6. $(10) : (10), tuple$
7. $[] : [], list$
8. $(0, 3) + (1, 0) : (0, 3, 1, 0), tuple$
9. $2 * [0, 1] * 2 : [0, 1, 0, 1, 0, 1], list$
10. $[1, 2, 3] + [5, 4] : [1, 2, 3, 5, 4], list$
11. 2 in (1,3,3) : $False, bool$
12. $2/3 : 0.666.., float$
13. $3^16 : 19, int$
14. $5|6 : 7, int$
15. $9\%7 : 2, int$
16. $-3 : -3, int$
17. $2 << 4 : 32, int$
18. $2 >> 2 : 0, int$
19. $-2 << 4 : -32, int$
20. $-2 >> 2 : -1, int$

21. $1//4 + 3//4 : 0, int$
22. $3 * *3 : 27, int$
23. $0.3 + 0.1 - 0.3 : 0.1, float$
24. $0.1 - 0.3 : -0.2, float$

# 2 Anwendung von Datentypen in Python

Listing 1: Output des Programms

```
[2, 3, 8]

[[2, 3, 8], [2, 3, 8], 100]

[[2, 3, 8], 100, [[2, 3, 8], [2, 3, 8], 100]]

[2, 3, 8]
```

# 3 Un- und Veränderliche Datentypen

Listing 2: kommentiertes, angepasstes Program

```
a=[1 ,3 ,(4 ,2) ] # lege liste a an
b=(a,a ,[1 ,5 ,4 ,3]) # lege tuple b an, b[0]=b[1]=id(a)
print(b [1][2]) # ausgabe: (4,2)
b [1][2]="wer"  # an der zweiten stelle von b ist adresse von a,
                #  in der liste a wird an dritte stelle (4,2) durch "wer" ersetzt
print(b)        # ausgabe : ([1, 3, 'wer'], [1, 3, 'wer'], [1, 5, 4, 3])
a [2][1]        # ausgabe : 'e'
#a [2][1]="wann" # ERROR! einzelnes element kann nicht geändert werden, da tuple! andere
    variante:
a[2] = "wann"   # ersetzt "wer" mit "wann"
print(b)        # ausgabe : ([1, 3, 'wann'], [1, 3, 'wann'], [1, 5, 4, 3])
b [2][1]        # ausgabe : 5
b [2][1]=["wo","wie"]   # 5 wird mit der liste ["wo", "wie"] ersetzt
print(b)        # ausgabe : ([1, 3, 'wann'], [1, 3, 'wann'], [1, ['wo', 'wie'], 4, 3])
#b[2]=[2,4]      # ERROR! b ist ein tuple, und element kann nicht geändert werden!
    Alternative
b[2][:] = [2,4]

print(b)        # ausgabe : ([1, 3, 'wann'], [1, 3, 'wann'], [2, 4])
a="warum"       # "neue" variable a wird mit string belegt
print(b)        # ausgabe : ([1, 3, 'wann'], [1, 3, 'wann'], [2, 4])
```

# 4 Dynamische Typsysteme

Bei der dynamischen Typisierung wird einer Variablen erst zur Laufzeit ein Datentyp zugeordnet und nicht explizit vorher im Programmcode angegeben. Das ermögicht ggf. mehr Flexibilität und das Programm bleibt in manchen fällen lauffähig obwohl ein nicht vorgeseher Datentyp verwendet wurde (und der Code ist meist besser lesbar). Andererseits fallen auch Zuweisungsfehler erst zur Laufzeit auf.

Quelle: https://de.wikipedia.org/wiki/Typisierung_(Informatik)