Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

FS24 CAS PML - Python

Niklaus Johner

niklausbernhard.johner@bfh.ch

FS24 CAS PML - Python

7. More about strings

String Initialisierung

- Anführungszeichen

```
quote = 'He said: "Hi there!"'
quote = "He said: 'Hi there!'"
```

- Escape-Sequenzen

```
escape = 'I\'m running'
escape = "Daring\nescape"
```

- Mehrzeiliger Block

```
quote = """He said:"I'm here" """
multiline = """Two
lines"""
```

String Formatierung

- *format* Methode erlaubt Variablen in Strings zu ergänzen:
  - {} sind die Substitutionsziele
  - Parameter von *format* werden dort substituiert

```
In [1]: animal = "cat"
   ...: number = 1
   ...: "I have {} {}".format(number, animal)
Out[1]: 'I have 1 cat'
```

- *f-strings*:

```
In [2]: f"I have {number} {animal}"
Out[2]: 'I have 1 cat'
```

find und *replace* Methoden

- Die *s.find(element)* Methode gibt die erste Position (Index) zurück, wo man *element* in *s* findet.
  - Das ist das selbe wie die *index* Methode

```
In [4]: string = "this and that"

In [5]: string.find("th")
Out[5]: 0
```

- Die *s.replace(old, new)* tauscht *old* mit *new* aus für jede Instanz von *old* die in *s* gefunden wird

```
In [6]: "Fall is coming".replace("Fall", "Winter")
Out[6]: 'Winter is coming'
```

Der *in* Operator

- Der *in* Operator testet ob eine Zeichenkette in einer anderen vor kommt

```
In [114]: "test" in "this is a test"
Out[114]: True

In [115]: "d" in "abc"
Out[115]: False
```

split und *join* Methoden

- Die *string.split(sep=None)* Methode trennt die Zeichenkette *string* und gibt eine *list* zurück
- *sep* definiert den Separator wo die Zeichenkette getrennt wird

```
In [37]: string = "This, is a short sentence"
    ...: string.split()
    ...:
Out[37]: ['This,', 'is', 'a', 'short', 'sentence']

In [38]: string.split(",")
Out[38]: ['This', ' is a short sentence']
```

split und *join* Methoden

- Die *sep.join(iteratable)* Methode verkettet ein iterierbares Objekt von strings mit *sep* dazwischen

```
In [42]: ", ".join(["this","is","comma","sparated"])
Out[42]: 'this, is, comma, sparated'

In [43]: ",".join("test")
Out[43]: 't,e,s,t'

In [44]: "".join(["t","e","s","t"])
Out[44]: 'test'
```

Zusätzliche Folien

startswith und *endswith* Methoden

- Die *s.startswith(string)* testet ob *s* mit *string* startet

```
In [73]: s="winter is coming"

In [74]: s.startswith("winter")
Out[74]: True

In [75]: s.startswith("summer")
Out[75]: False
```

- Die *s.endswith(string)* testet ob *s* mit *string* endet

```
In [76]: s.endswith("ing")
Out[76]: True

In [77]: s.endswith("leaving")
Out[77]: False
```

Stripping Methoden

- Die *s.strip()* Methode löscht Leerräume am Anfang und Ende von *s*

```
In [97]: "   too many spaces   ".strip()
Out[97]: 'too many spaces'
```

- Es können auch andere Zeichen angegeben werden die gelöscht werden sollen.

```
In [98]: " , ,spaces and commas,  ".strip(" ,")
Out[98]: 'spaces and commas'
```

- *s.rstrip()* und *s.lstrip()* löschen respektiv nur am Anfang oder am Ende von *s*

```
In [99]: "   too many spaces   ".rstrip()
Out[99]: '   too many spaces'
```

Justification Methoden

- Die *s.rjust(breite)* und *s.ljust(breite)* Methoden fügen rechts (respektiv links) Leerräume ein.

```
In [100]: "I need more space(s)".rjust(25)
Out[100]: '      I need more space(s)'

In [101]: "I need more space(s)".ljust(25)
Out[101]: 'I need more space(s)      '
```

- Die *s.center(breite)* Methode fügen rechts und links Leerräume ein.

```
In [110]: "I float in space(s)".center(25)
Out[110]: '   I float in space(s)   '
```

Mehr string Formatierung

- Man kann das Format noch kontrollieren mit:
  - *{:breite.genauigkeit}*

```
In [12]: pi = 3.141593

In [13]: "pi={:.2}".format(pi)
Out[13]: 'pi=3.1'

In [14]: f"pi={pi:.2}"
Out[14]: 'pi=3.1'

In [15]: "pi={:10.2}".format(pi)
Out[15]: 'pi=       3.1'
```