Python - Befehlsübersicht

1 Grundlagen

1.1 Kommentare

```
# Ein Kommentar, der bis ans Zeilenende geht
"""
Ein Kommentar, der über
mehrere Zeilen geht.
"""
```

1.2 Variablen

```
# Eine neue Variable Zahl mit dem Namen 'meineZahl' und Wert 42
meineZahl = 42

# Eine andere Variable mit einem Text-Wert, Texte müssen in ".." stehen.
meinText = "Hallo Welt"
```

1.3 Ausgabe

```
# Gibt einen Text aus
print("Hallo Welt")
```

```
# Gibt den Wert einer Variablen aus
name = "Mark"
print("Hallo")
print(name)

# 'print' akzeptiert mehrere Werte gleichzeitig
print("Hallo", name)
```

1.4 Operationen

1.4.1 Umgang mit Zahlen

```
# Mit Zahlen kann ganz normal rechnen
wert = 10
```

```
wert = 10 + 20    # = 30
wert = 10 - 20    # = -10
wert = 10 / 2    # = 5
wert = 10 * 2    # = 20
wert = 10 % 3    # = 1, Da 10 / 3 = 3 Rest 1
```

1.4.2 Umgang mit Text

```
# Texte kann man '+' aneinander anfügen
begruessung  = "Hallo "
text = begruessung + "Mark" # "Hallo Mark"
# Texte kann man auch multiplizieren/wiederholen
text = "Ha" * 3  # HaHaHa
```

1.5 if-Abfragen

1.5.1 Bedingung

```
# Eine Bedingung kann entweder Wahr (True) oder Falsch (False) sein
bedingung = 10 > 20 # False
print("Die Bedingung ist:", bedingung)
```

1.5.2 if

```
wert = 10 # Eine Variable mit einem beliebigen Wert (hier 10)

# eine if-Abfrage überprüft die Bedingung ob diese Wahr oder falsch ist
if wert > 20:
    # wird ausgeführt wenn die Bedingung wahr ist (ACHTUNG: Einrückung)
    print("Der Wert ist größer als 20")

# Hier gehts weiter (ACHUTUNG: Nicht eingerückt)
```

1.5.3 if-else

```
# mit einer if-else Abfrage kann man auch auf eine nicht erfüllte
# Bedingung mit dem 'else'-Zweig reagieren
if wert > 20:
    # wird ausgeführt wenn die Bedingung erfüllt ist
    print("Der Wert ist größer als 20")
else:
    # wird ausgeführt wenn die Bedingung nicht erfüllt ist
    print("Wert ist kleiner oder gleich 20.")

# Hier gehts weiter (nicht eingerückt)
```

1.6 Schleifen

1.6.1 while-Schleife

```
# Führt solange den eingerückten Code aus, wie die Bedingung erfüllt ist
zahl = 1
while zahl < 10:
    # dieser eingerückte Code wird wiederholt (ACHTUNG: Einrückung)
    print("Zahl:", zahl) # Wert ausgeben
    zahl = zahl +1 # Zähler erhöhen

# Hier gehts weiter (ACHTUNG: nicht eingerückt)
```

1.6.2 for-Schleife

```
# Führt solange den eingerückten Code aus, wie die Bedingung erfüllt ist

for zahl in range(1, 10): # In nimmt Werte 1-9 an
    # dieser eingerückte Code wird wiederholt (ACHTUNG: Einrückung)
    print("Zahl:", zahl) # Wert ausgeben

# Hier gehts weiter (ACHTUNG: nicht eingerückt)
```

1.7 Eingabe durch den Benutzer

1.7.1 input - Text einlesen

```
# mit input(..) kann man den Benutzer aufforden einen Text einzugeben
eingabeText = input("Bitte Text eingeben: ");
print(eingabeText); # gibt den eingelesenen Text aus
```

1.7.2 input - Eine Zahl einlesen

```
# mit input(..) kann man den Benutzer aufforden einen Text einzugeben
eingabeText = input("Bitte eine Zahl eingeben: ");

# Diesen Text wollen wir mittels int(..) in eine Zahl konvertieren:
eingabeZahl = int(eingabeText)
# ACHTUNG: wird keine Zahl eingeben, so stürzt das Programm ab
print(eingabeZahl); # gibt die eingelesene Zahl aus
```

1.8 Zufallszahlen

```
# das random Modul muss (einmal) importiert werden um die Funktion
# random.randint() verwenden zu können
```

```
import random

# generiert eine Zufallszahl zwischen 1 und 100
zufallsZahl = random.randint(1,100)
```

2 Fortgeschritte Grundlagen

2.1 Geschachtelte For-Schleifen

```
# geschachtelte for-Schleife, die die innere Schleife 9 mal ausführt
for zahlAussen in range(1, 10):      # nimmt Werte 1-9 an
    for zahlInnen in range(1, 10):    # nimmt Werte 1-9 an
        # beide Wert ausgeben
        print("Äußere Zahl:", zahlAussen, "Innere Zahl:", zahlInnen)
```

2.2 Funktionen

2.2.1 Eine einfache Funktion

```
# Funktion definieren
def sageHallo():
    print("Hallo Welt")

# Funktion aufrufen
sageHallo()
```

2.2.2 Funktion mit Übergabeparametern

```
# Funktion definieren
def sageHallo(name, alter):
    print("Hallo", name)
    print("Du bist", alter, "Jahre alt")

# Funktion aufrufen
sageHallo("Mark", 22)
```

2.2.3 Funktion mit Rückgabewerten

```
# Funktion definieren
def addiere(zahl1, zahl2):
```

```
3       summe = zahl1 + zahl2
4       return summe
5
6   # Funktion aufrufen
7   ergebnis = addiere(12, 22)
8   print(ergebnis)
```

2.3 Listen

2.3.1 Eine Liste erstellen

```
1   # eine Liste erstellen
2   liste = [1, 2, 3, 4, 5]
```

2.3.2 Eine Liste ausgeben

```
1   # die Elemente einer Liste könne so ausgeben werden
2   for element in liste:
3       print(element)
4
5   # Hilfsfunktion für das Ausgeben einer Liste
6   def liste_ausgeben(liste):
7       print("Liste mite", len(liste), "Einträgen")
8       # die Elemente ausgeben
9       for element in liste:
10      print(element)
11
12  # Diese ruft man dann so auf
13  liste_ausgeben(liste)
```

2.3.3 Einen Eintrag aus der Liste lesen

```
1   liste = ["hallo", "test", "welt"]
2   # Ein Eintrag kann so gelesen werden:
3   ersterEintrag = liste[0] # = "hallo". Achtung wir beginnen bei 0 !
4
5   # Negative Indices beginnen am Ende zu zählen
6   letzterEintrag = liste[-1] # = "welt". Wir beginnen bei -1 !
7
8   print(ersterEintrag, letzterEintrag) # => "Hallo Welt"
```

2.3.4 Die Liste erweitern

Einen Eintrag hinzufügen

```
# einen Eintrag anfügen
liste.append(6)
liste_ausgeben(liste)
```

Zwei Listen kombinieren

```
# Liste mit einer anderen Liste kombinieren
liste = liste + [7, 8, 9]
liste_ausgeben(liste)
```