1 Funktionen und Module

1.1 Funktionsdefinition

- Funktionen werden zur Modularisierung von Programmen verwendet
    - wiederkehrende Programmabschnitte werden in einer Funktion ausgelagert, die dann nur noch aufzurufen ist
    - umfangreiche Programmabschnitte in übersichtliche Teile zu gliedern
    - das führt zur Verbesserung der Übersichtlichkeit des Quellcodes
    - Steigerung der Effizienz bei der Programmierung durch Wiederverwendbarkeit von Funktionen
    - Vermeidung von Fehlern durch Verwendung von geprüften Funktionen
- Funktionsdefinition erfolgt mit **def**, dem Funktionsname und einem Paar aus runden Klammern **()**
- die Definition der Funktion muss zu Beginn stehen, vor dem ersten Aufruf dieser Funktion
- die Anweisungen der Funktion werden eingerückt
- Namenskonvention für Funktionsnamen analog Variablennamen
- Verarbeitungsschritte bei der Verwendung von Funktionen:
    - mit dem Funktionsaufruf wird die Verarbeitung im rufenden Programmabschnitt angehalten
    - die Anweisungen der Funktion werden durchlaufen
    - mit dem Ende der Funktion wird zum aufrufenden Programmabschnitt zurückgekehrt und die Verarbeitung dort fortgesetzt

- Beispiel: mit wiederkehrenden Abschnitten

```
if geschlecht == 'w':
    if stunde < 10:
        print("Guten Morgen ")
    elif stunde < 18:
        print("Guten Tag ")
    else:
        print("Guten Abend ")
    print("Frau", name)
elif geschlecht == 'm':
    if stunde < 10:
        print("Guten Morgen ")
    elif stunde < 18:
        print("Guten Tag ")
    else:
        print("Guten Abend ")
    print("Herr", name)
else:
    if stunde < 10:
        print("Guten Morgen ")
    elif stunde < 18:
        print("Guten Tag ")
    else:
        print("Guten Abend ")
    print("Frau/Herr", name)
```

- Beispiel: mit Funktionsdefinition und Funktionsaufruf

```
def begruessung():
    if stunde < 10:
        print("Guten Morgen ")
    elif stunde < 18:
        print("Guten Tag ")
    else:
        print("Guten Abend ")
:
if geschlecht == 'w':
    begruessung()
    print("Frau", name)
elif geschlecht == 'm':
    begruessung()
    print("Herr", name)
else:
    begruessung()
    print("Frau/Herr", name)
print("Ende")
```

- diese Funktionsdefinition verwendet die Variable *stunde* des aufrufenden Moduls
    - Verwendung von globalen Variablen ist schlechter Programmierstil
    - kann zu Fehlern und unerwünschten Seiteneffekten führen
    - eine Zuweisung zur Variablen *stunde* in der Funktion führt zu einer Fehlermeldung
    - besser die Variable *stunde* als Parameter übergeben

1.2 Übergabe von Werten an die Funktion

- einer Funktion können beim Aufruf Werte übergeben werden

- Beispiel: Funktionsdefinition und Funktionsaufruf mit Parametern

```
# Definition der Funktion begruessung
def begruessung(stunde):
    if stunde < 10:
        print("Guten Morgen ")
    elif stunde < 18:
        print("Guten Tag ")
    else:
        print("Guten Abend ")

if geschlecht == 'w':
    begruessung(stunde)
    print("Frau", name)
elif geschlecht == 'm':
    begruessung(stunde)
    print("Herr", name)
else:
    begruessung(stunde)
    print("Frau/Herr", name)
print("Ende")
```

1.3 Funktionsdefinition mit Rückgabewert

- eine Funktion kann einen Rückgabewert ermitteln und an das aufrufende Modul zurückgeben
- beim Funktionsaufruf können als Parameter sowohl Variablen wie auch Konstanten angegeben werden

- Beispiel:

```
# Definition der Funktion ermittle_min
def ermittle_min(a, b):
    if (a < b):
        return a
    else:
        return b

n1 = 12
n2 = 5
min_wert = ermittle_min(n1, n2)
print(min_wert)

min_wert = ermittle_min(n1, 34)
print(min_wert)
```