

```
#Variable Vorname
# CSharp:
# string vorname;
# string nachname;
# int alter;
# vorname = "Nicolas";
# vorname = 10; Fehler
```

```
# Python:
# vorname = 'Nicolas'
# vorname = 30
# vorname = True
# vorname = [123,34,45,5]
vorname = 'Nicolas'
# ['Nicolas'] | | | | |
# 0x1kjbasdf23uo24 => vorname
del(vorname)
```

```
def summe(zahl1, zahl2):
    return zahl1+zahl2
```

```
def liste_summe(*zahlen):
    summe = 0
    for zahl in zahlen:
        summe += zahl
    return summe
```

```
zahlen1 = [1,2,32,344,45,45,23,45]
liste_summe(zahlen1)
zahlen2 = [233434,3,23,2445,365,65]
liste_summe(zahlen2)
# # Unterscheidung des Alters
```

```
# alter = 17 # festes alter, oder benutzer eingeben lassen:
# alter = int(input('Wie alt bist du? '))
```

```
# # fallunterscheidung
# if alter >= 18:
#     print('Du bist volljährig')
# else:
#     print('Du bist minderjährig')
```

```
# # Notenberechnung
```

```
# punkte = 85 # feste punktzahl, oder benutzer eingeben lassen:
# punkte = int(input('Wie viele Punkte hast du erreicht? '))
```

```
# # fallunterscheidung mit mehreren fällen.
# # genau einer der fälle wird auf jeden fall ausgeführt.
# if punkte >= 90:
```

```
# print('Sehr gut')
# elif punkte >= 75:
# print('Gut')
# elif punkte >= 60:
# print('Befriedigend')
# else:
# print('Ausreichend')
```

```
## Listendefinition
# countries = ['Deutschland', 'Frankreich', 'Polen']
```

```
## index von listen
# zahlen = [
#     1,    # das 0-te Element
#     0,    # das 1-te Element
#     -4,   # das 2-te Element
#     12,   # das 3-te Element
#     3413, # das 4-te Element
#     -28432, # das 5-te Element
#     7     # das 6-te Element
# ]        # ^ der Index
```

```
## zugriff auf listenelement mittels index-operator:
# zahlen[4]
```

```
## Range
```

```
## range(5) == [0,1,2,3,4]
## range(10) == [0,1,2,3,4,5,6,7,8,9]
## range(100) == [0,1,2,3,4,5,6,7,.....,97,98,99]
## range(11,20,3) == [11,14,17]
```

```
## f-Strings
# name = 'Bernd'
# wetter = 'regnerisch'
# begruessung = 'Hallo ' + name # == 'Hallo Bernd'
# begruessung = f'Hallo {name}, das Wetter ist heute {wetter}!' # Hallo Bernd, das Wetter ist heute regnerisch
# begruessung = 'Hallo ' + str(name) + ', das Wetter ist heute ' + str(wetter) + '!'
```

```
# s1 = 'foo'
# s2 = 'bar'
# v = s1 + s2
# v = f'{s1}{s2}'
```

```
## Schleifen
```

```
## durchlauf einer liste mittels einer for-Schleife
# for country in countries:
```

```

# # f-Strings f'...' anstelle von String-Verkettung 'abc' + 'def'
#  ausgabe = f'Das aktuelle Land ist {country}'
#  print(ausgabe)

# # while-Schleife
#  eingabe = ""

#  while eingabe != 'ja':
#      print('Hallo!')
#      eingabe = input('Beenden? ')

# # Alle Zahlen >= 0 addieren, aber mit einer while-Schleife (s.u. für for-schleife)

# zahlen = [1, 0, -4, 12, 3413, -28432, 7]

# index = 0
# laenge = len(zahlen)
# sum = 0

# while index < laenge:
#     zahl = zahlen[index]
#     if zahl >= 0:
#         sum += zahl
#     index += 1

# print(sum)

# # normalerweise würde man das mit einer for-Schleife machen:

# zahlen = [1, 0, -4, 12, 3413, -28432, 7]
# sum = 0
# for zahl in zahlen:
#     if zahl >= 0:
#         sum += zahl

# print(sum)

# # Scope von Variablen

# x = 'cool' # globale variable

# def function():
#     y = 3 # lokale variable (lokal bezogen auf die Funktion "function")

#     print(x) # hier wird die globale Variable x ausgegeben

# function()
# print(x)

```

```

## return-Anweisung
# return beendet eine Funktion an dieser Stelle. Der Programmfluss
# kehrt an die Stelle des Aufrufs der Funktion zurück.

# CSharp
# Console.WriteLine();
# cw
# ----
# Java
# System.out.println();
# sout

# def fun(a):
#     if a == 0:
#         return
#     print(f'{a} ist ungleich null') # wird nicht ausgeführt wenn a gleich 0 ist

# fun(0) # keine Ausgabe
# fun(3) # 3 ist ungleich null

## Funktion die prüft, ob Zahl gerade ist

# def gerade(zahl):
#     if zahl % 2 == 0:
#         return True
#     else:
#         return False

## 'kombinierte Zuweisungsoperatoren' (engl. augmented assign operators
## oder in-place operators)
# i = 0 # kein in-place operator, nur um i zu definieren

# i = i + 1
# i += 1

# i = i - 1
# i -= 1

# i = i * 1
# i *= 1

# i = i / 1
# i /= 1

# i = i % 1
# i %= 1

## Umrechnung Celsius Fahrenheit
## Vom Nutzer Temperatur in Celsius einlesen

```

```
# # Daraus die Temperatur in Fahrenheit berechnen und ausgeben
```

```
# print('Was möchtest du tun?')  
# print('[F]ahrenheit in Celsius')  
# print('[C]elsius in Fahrenheit')
```

```
# auswahl = input()
```

```
# if auswahl == 'C':  
#     celsius = float(input('Welche Temperatur in Celsius? '))  
#     fahrenheit = celsius * 9 / 5 + 32  
#     print(f'{fahrenheit} °F')  
# else:  
#     fahrenheit = float(input('Welche Temperatur in Fahrenheit? '))  
#     celsius = (fahrenheit - 32) * 5 / 9  
#     print(f'{celsius} °C')
```

```
# # Beispiel Fahrenheit mit tkinter GUI
```

```
# import tkinter as tk  
# from tkinter import messagebox
```

```
# def celsius_zu_fahrenheit():  
#     """Liest die Celsius-Eingabe, rechnet um und zeigt das Ergebnis an."""  
#     try:  
#         celsius_str = eingabe_celsius.get()  
#         celsius = float(celsius_str)  
#         fahrenheit = (celsius * 9.0 / 5) + 32  
#         ergebnis_text = f"↔■ {celsius}°C entspricht {round(fahrenheit, 2)}°F"  
#         label_ergebnis.config(text=ergebnis_text)  
#     except ValueError:  
#         messagebox.showerror("Fehler", "Ungültige Eingabe! Bitte gib eine Zahl ein.")  
#         label_ergebnis.config(text="Bitte Zahl eingeben.")
```

```
# root = tk.Tk()  
# root.title("Celsius ■ Fahrenheit")  
# root.geometry("300x150")
```

```
# label_prompt = tk.Label(root, text="Temperatur in Celsius (°C):")  
# label_prompt.pack(pady=5)
```

```
# eingabe_celsius = tk.Entry(root, width=15)  
# eingabe_celsius.pack(pady=5)
```

```
# button_umrechnen = tk.Button(root, text="Umrechnen", command=celsius_zu_fahrenheit)  
# button_umrechnen.pack(pady=5)
```

```
# label_ergebnis = tk.Label(root, text="Ergebnis: Warten auf Eingabe...", fg="blue")  
# label_ergebnis.pack(pady=10)
```

```

# root.mainloop()

# # Dictionaries

# # Ein Dictionary ist eine Sammlung von Key-Value-Pairs (Schlüssel-Wert-Paaren)
# # Key (Schlüssel) ist immer ein String, der Value (Wert) ist beliebig:

# schueler = {
#     'name': 'Paul Bach', # Value ist ein String
#     'alter': 14, # Value ist ein Integer
#     'wohnort': 'Berlin', # Value ist ein String
#     'durchschnittsnote': 2.5, # Value ist ein Float
#     'noten': { 'mathe': 3, 'deutsch': 2 } # Value ist ein Dictionary
# }

# # Zugriff auf values mittels dict[key]
# name = schueler['name']

# print(f'Der Name der Person lautet: { name }')
# print(f'Sie ist { schueler['alter'] } Jahre alt.')

# # man kann Dictionaries auch komplett ausgeben
# print(schueler)

# # beim lesenden Zugriff auf Keys, die es nicht gibt, kommt ein Fehler
# print(schueler['plz'])

# # man kann aber schreibend auf Keys zugreifen, die es nicht gibt,
# # also neue Einträge dem dictionary hinzufügen
# schueler['plz'] = '01234'

# # in einer Schleife alle Keys durchlaufen und jeden Key ausgeben:
# for key in schueler:
#     print(key)

# # in einer Schleife alle Keys durchlaufen und jeden Value dazu ausgeben:
# for key in schueler:
#     print(schueler[key])

# # in einer Schleife alle Values durchlaufen und jeden Value ausgeben:
# for value in schueler.values():
#     print(value)

# # in einer Schleife alle Items (Key-Value-Pairs) durchlaufen:
# for key, value in schueler.items():
#     print(f'Der Wert zu dem Key {key} ist {value}')

```

```
# # Webseiten/-apps mit Flask
```

```
# from flask import Flask
```

```
# # Flask-Instanz erzeugen
```

```
# app = Flask(__name__)
```

```
# # Route '/', also die "Startseite", also z.B. example.com/, dann wird
```

```
# # diese Funktion aufgerufen
```

```
# @app.route('/')
```

```
# def hello():
```

```
#     return 'Hello World' # Im Browser steht dann Hello World
```

```
# # Flask App starten (Main loop wird ausgeführt, App wartet auf HTTP requests)
```

```
# app.run(debug=True)
```