Programmiereinstieg mit

Python

Vorstellungs-runde

Erwartungs-abfrage

Agenda

Einstieg
Erste Schritte in Python
Kaffeepause
Steuerlogik & Wiederholungen
Mittagspause
Python für Automatisierung & Datenverarbeitung
Projekte
Präsentationen & Abschluss

01

Erste Schritte

Guido van Rossum

- Hat 1991 Python entwickelt
- Benannt nach Monty Python
- Programmieren soll Spaß machen und zugänglich sein
- Code Lesbarkeit

Python 3

- Web Development
- Data Science, Machine Learning
- Scripting

Variablen

- Eine Variable ist ein benannter Ort, an dem etwas gespeichert wird
- Variablen richtig benennen
  - Groß- und Kleinschreibung
  - “Snake case”-Schreibweise
  - Sprechende Namen
  - Reservierte keywords
- Variablen-Zuweisungen ändern

Der Typ wird einer Variable in Python bei der Ausführung des Programms zugewiesen. Er muss nicht explizit gemacht werden.

Das nennt man dynamische Typisierung.

Verschiedene Typen

```
a = "Hallo"    # Text
b = 123        # Ganzzahl
c = 2.5        # Kommazahl
```

Verschiedene Typen

```
a = "Hallo"    # String
b = 123        # Integer
c = 2.5        # Float
```

Eingabe und Ausgabe

Variablen können mit dem Befehl `print()` auf der Konsole ausgegeben werden.

Tastatur-Eingaben können mit dem Befehl `input()` eingelesen werden.

```
a = "Hallo"

print("A hat den Wert " + a + "!")

print(f"A hat den Wert {a}!")
```

Übung

Erstellt ein kleines Programm für ein Haustier / Tamagotchi mit folgenden Eigenschaften:

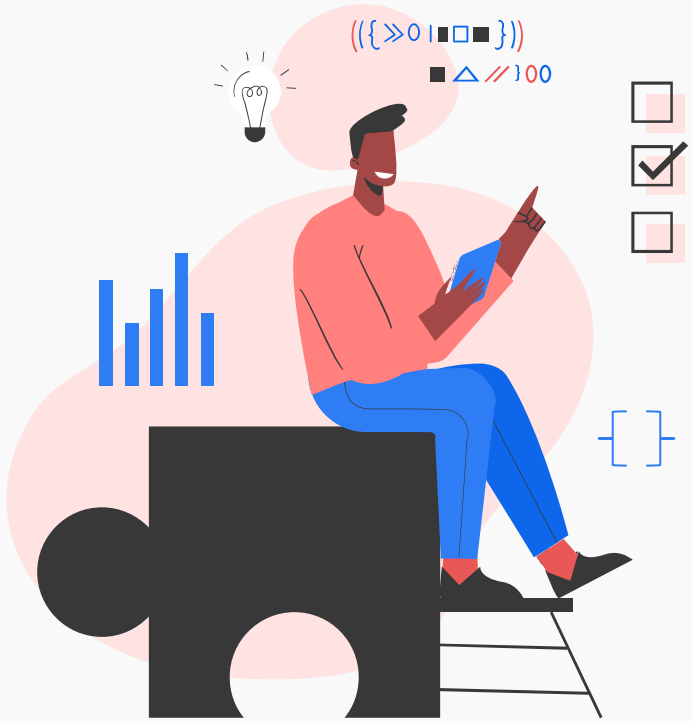
- Name
- Alter
- Gesundheitslevel
- Energielevel
- Erfahrungslevel

Name und Alter sollen über die Konsole eingelesen werden.
Alle Eigenschaften sollen formatiert ausgegeben werden.

Ein Programm kann über das Terminal mit dem Befehl `python dateiname.py` gestartet werden.

Pause

15 Minuten



Es gibt einen speziellen Typen, um Wahrheitswerte darzustellen. Er hat nur zwei Zustände: wahr oder falsch.

Dieser Typ heißt Boolean.

Vergleiche

<	kleiner als
<=	kleiner gleich
>	größer als
>=	größer gleich
==	gleich
!=	ungleich


energie < 25
ja
nein
müde
gesundheit < 25
ja
nein
krank
ansonsten
fit

Verzweigungen

wenn..., dann...	ansonsten wenn..., dann...	ansonsten...
`if`	`elif`	`else`

```
if energie < 25:
    ...
elif gesundheit < 25:
    ...
else:
    ...
```

wenn...

ansonsten, wenn...

ansonsten...

Übung

Erweitert euer bisheriges Programm um die folgenden Verzweigungen:

- Das Tamagotchi ist müde, wenn die Energie unter 50 ist
- Das Tamagotchi ist volljährig, wenn es mindestens 18 Jahre alt ist
- Das Tamagotchi ist entweder ein Neuling (Level 1- 20), Erfahren (Level 21–50) oder Profi (Level 51–100)

Die Ergebnisse der Verzweigungen sollen auf der Konsole ausgegeben werden.

Listen

- feste Reihenfolge
- kann verschiedene Typen enthalten

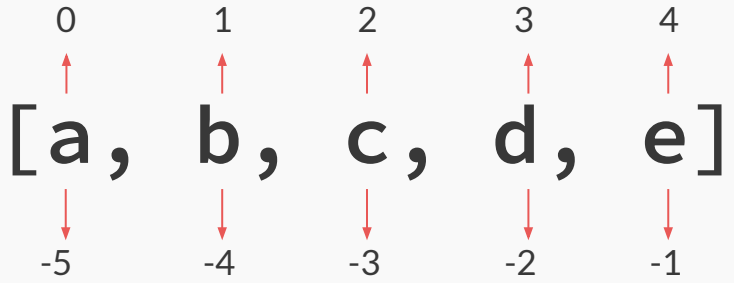
```
[a, 1, "b", True]
```

- kann doppelte Elemente enthalten

Listen erstellen

```
a = ["Hallo"]
b = list("Hallo")
```

Indizes



Indizes

```
liste = list("hallo")
print(liste[0])
print(liste[-1])
```

```
h
o
```

Schleifen

- n mal wiederholen
- für jedes Listenelement wiederholen
- wiederholen, bis ein Ereignis auftritt

for Schleife

Iterieren über eine Sequenz

```
farben = ["rot", "grün", "blau"]
for farbe in farben:
    print(farbe)
```

for Schleife

Iterieren über eine Sequenz

```
for i in range(5):
    print(i)
```

range

Erzeugen von Zahlen

```
range(3)
range(1, 4)
range(0, 10, 2)
```

range

Erzeugen von Zahlen

```
range(3)          → 0, 1, 2
range(1, 4)       → 1, 2, 3
range(0, 10, 2)   → 0, 2, 4, 6, 8
```

while Schleife

Wiederholen, solange eine Bedingung wahr ist

```
while bedingung:
    # Codeblock, der wiederholt wird
```

while Schleife

```
x = 0
while x < 5:
    print(x)
```

while Schleife

```
x = 0
while x < 5:
    print(x)
# x wird nie erhöht!
```

while Schleife

Abbruchbedingungen

```
x = 0
while x < 5:
    print(x)
    x += 1
```

for vs while

Situation	Schleifentyp
Bekannte Anzahl Wiederholungen	`for`
Iteration über eine Liste	`for`
Wiederholen, bis eine Bedingung eintritt	`while`
Warten auf Benutzereingabe	`while`

Funktionen

- Benannte Codeblöcke, die eine Aufgabe erfüllen
- Machen Programme strukturierter, kürzer, wiederverwendbar
- Funktionen können Parameter (Eingaben) und Rückgabewerte (Ausgaben) haben

Aufbau

```
def funktionsname(a1, a2, ...):
    # Codeblock
    return rückgabewert

ergebnis = funktionsname(x, y)
print(ergebnis)
```

Aufruf

```
def sag_hallo():
    print("Hallo!")

sag_hallo()
```

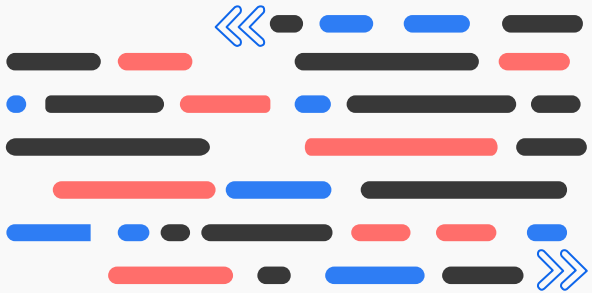
Übung

Erstellt ein Programm, bei dem eine Zahl zwischen 1 und 100 erraten werden soll.

Das Programm wählt eine zufällige Zahl. Über die Tastatur werden so lange Tipps eingegeben, bis die richtige Zahl erraten ist. Nach jedem Tipp soll ausgegeben werden, ob er zu niedrig, zu hoch oder richtig war. Am Ende soll zusätzlich ausgegeben werden, was die bisherigen Tipps waren und wie viele Versuche benötigt wurden.

Pause

60 Minuten



Analysebeispiele

Dateiumbenennung

100 Dateien nach Muster umbenennen

Daten bereinigen

Kundenlisten aufräumen

Automatisierte E-Mails

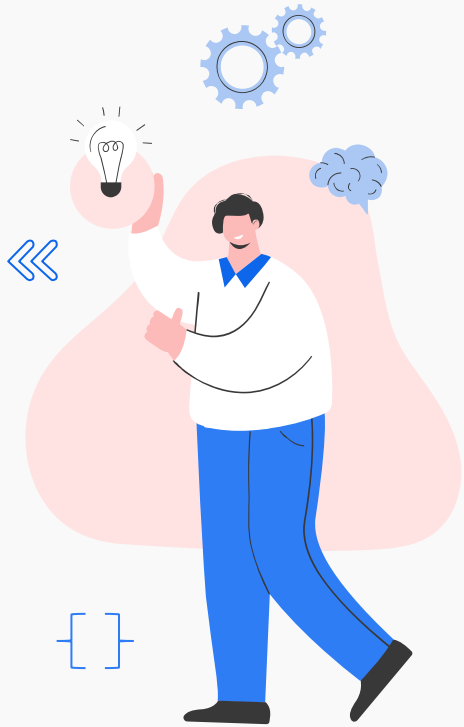
mit Anhängen oder Reports

Rechnungen sortieren

PDF-Dateien nach Datum & Inhalt

Berichte erstellen

Zahlen aus CSV-Dateien extrahieren und zusammenfassen



Dateien lesen und schreiben

```
with open("daten.txt", "r", encoding="utf-8") as f:
    for zeile in f:
        print(zeile.strip())
```

```
with open("ausgabe.txt", "w", encoding="utf-8") as f:
    f.write("Hallo Welt!\n")
```

Dateien bearbeiten

```
import os
```

`os.listdir(pfad)` – Liste von Dateien

`os.rename(alt, neu)` – Datei umbenennen

`os.remove(pfad)` – Datei löschen

Beispiel: Dateien bearbeiten

```
berichte
    → finaler bericht.txt
    → bericht anja 14.07.txt
    → familie.png
    → workshop anmeldeliste.txt
    → setup.exe
    → test_daten.txt
```

Beispiel: Dateien bearbeiten

```
import os

for datei in os.listdir("berichte/"):
    if datei.endswith(".txt"):
        neu = datei.replace(" ", "_")
        os.rename("berichte/" + datei, "berichte/" + neu)
```

Bereinigung

- Typische Probleme:
  - Falsches Dezimaltrennzeichen
  - Whitespace
  - Leere Felder oder fehlende Daten

→ **Ziel: Einheitliches Format für weitere Verarbeitung**

CSV Dateien

Name	Umsatz
Müller	9.99
Mayer	19.99
Schmidt	29.99

CSV Dateien einlesen

```
import csv

with open("daten.csv", newline='', encoding="utf-8") as f:
    reader = csv.DictReader(f)
    for zeile in reader:
        print(zeile["Name"], zeile["Umsatz"])
```

Automatisierte Verarbeitung

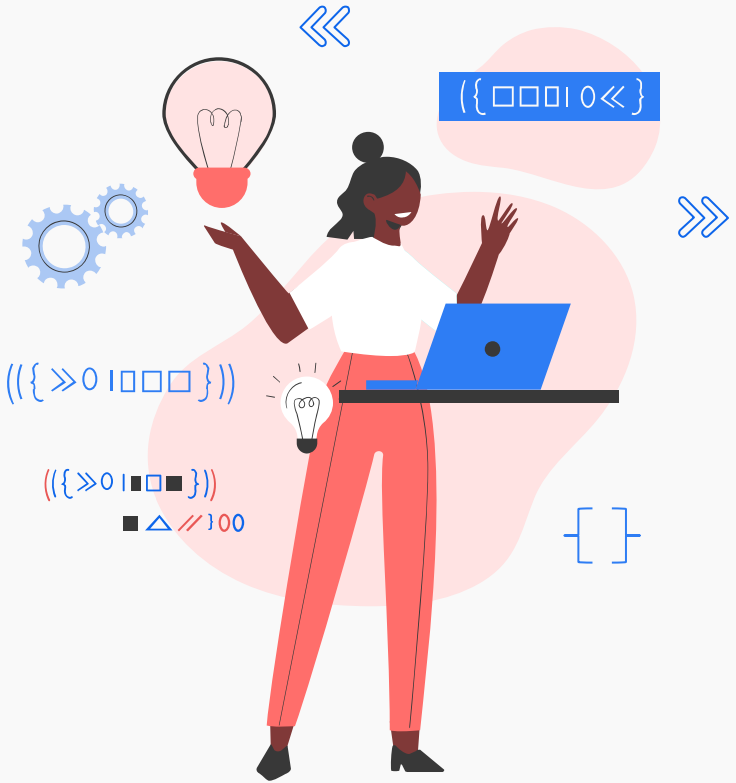
- Mehrere CSVs automatisch analysieren
- Schwellenwerte prüfen
- Ergebnisse filtern, markieren oder speichern

Beispiel: Schwellwert-Alarm

```
csvs
    → wetter_mai.csv
    → wetter_juni.csv
    → wetter_juli.csv
```

Tag	Temperatur
Montag	28.5
Dienstag	32.3
Mittwoch	23.9

Projekte



https://shorturl.at/Re7GR

Präsentation

Tools

Pandas

komfortabler CSV-Import & Datenanalyse

Matplotlib / Seaborn

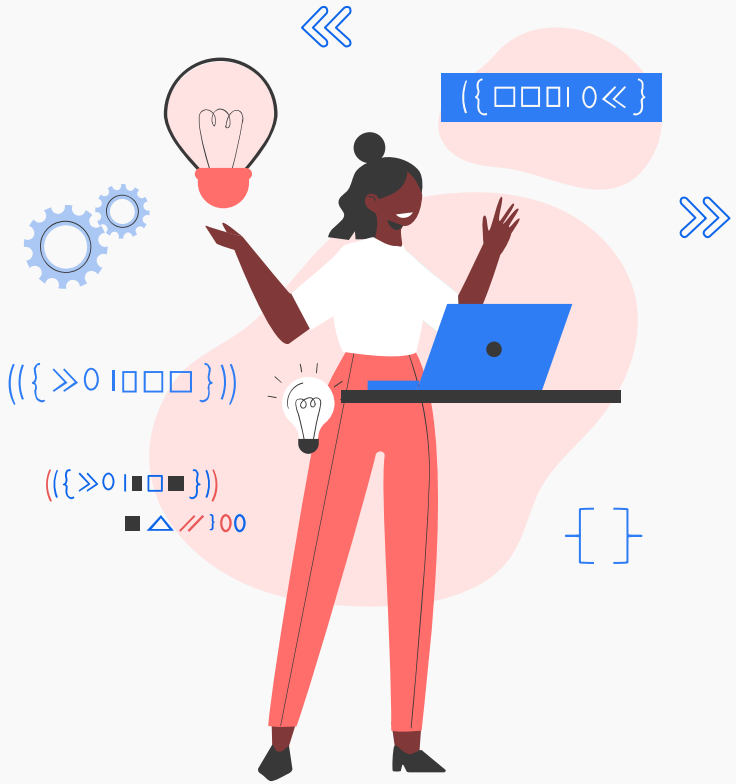
Daten visualisieren

smtplib

automatisierter E-Mail-Versand

SQLite / SQLAlchemy

Datenbankzugriffe



Feedback

https://www.menti.com/aldw5h8rfxu2

Danke!

anja.bertels@th-koeln.de
dominik.deimel@th-koeln.de