Python

Einführung in Python

TECH STARTER

Agenda

1. Was ist Python?
2. Python und VS Code installieren
3. Primitive Datentypen
4. IP-Adressen und CIDR
5. Domains

Einführung in Python

- ➔ Python ist eine hochrangige, interpretierte Programmiersprache
- ➔ Sie wurde von Guido van Rossum in den späten 1980er Jahren entwickelt
- ➔ Python zeichnet sich durch eine einfache und leicht lesbare Syntax aus
- ➔ Die Beliebtheit von Python ist auf seine Vielseitigkeit und breite Anwendungsbereiche zurückzuführen

Vorteile von Python

- ➔ Einfache Lesbarkeit und übersichtliche Syntax
- ➔ Umfangreiche Standardbibliothek mit vorgefertigten Modulen
- ➔ Plattformunabhängigkeit (Windows, macOS, Linux)
- ➔ Aktive Entwicklergemeinschaft und regelmäßige Updates
- ➔ Schnelle Prototypenentwicklung, verkürzte Entwicklungszeiten

Herausforderungen von Python

- ➔ Python kann im Vergleich zu anderen Sprachen etwas langsamer sein, da es interpretiert wird
- ➔ Es ist nicht die beste Wahl für rechenintensive Aufgaben
- ➔ Die Verwendung von Python in eingebetteten Systemen oder speicherintensiven Anwendungen kann problematisch sein
- ➔ Python 2 und Python 3 sind nicht vollständig kompatibel, was zu Kompatibilitätsproblemen führen kann

Anwendungsbereiche von Python

➔ Python wird in verschiedenen Bereichen eingesetzt: Webentwicklung, Datenanalyse, maschinelles Lernen und Automatisierung

- Entwicklung von Websites, Webanwendungen und Servern
- Bibliotheken für statistische Analysen, Visualisierung und Datenmanipulation
- Bibliotheken für künstlichen Intelligenz und im maschinellen Lernen
- Durch Automatisierungsmöglichkeiten erleichtert Python repetitive Aufgaben

Vergleich Bash vs. Python

Bash-Skript

```bash
#!/bin/bash

# Zwei Zahlen definieren
a=5
b=10

# Die Zahlen addieren
sum=$((a + b))

# Das Ergebnis ausgeben
echo "Die Summe von $a und $b ist: $sum"
```

Python-Skript

```python
# Zwei Zahlen definieren
a = 5
b = 10

# Die Zahlen addieren
sum = a + b

# Das Ergebnis ausgeben
print(f"Die Summe von {a} und {b} ist: {sum}")
```

Primitive Datentypen

- ➔ Primitive Datentypen ermöglichen die Speicherung und Manipulation grundlegender Daten in Python
- ➔ Primitive Datentypen werden verwendet, um grundlegende Informationen zu repräsentieren und zu verarbeiten.

Integer (int)

- ➔ Darstellung ganzer Zahlen ohne Dezimalstellen
- ➔ Beispiele: -5, 0, 42
- ➔ In Python
- ➔

```
# Beispiel für Integer (int):
age: int = 25
print("Das Alter beträgt:", age)
```

Float (float)

- ➔ Darstellung von Zahlen mit Dezimalstellen
- ➔ Beispiele: 3.14, -0.5, 1.0
- ➔ In Python
- ➔

```
# Beispiel für Float (float):
pi: float = 3.14
print("Der Wert von Pi ist:", pi)
```

String (str)

- ➔ Darstellung von Zeichenketten oder Text
- ➔ Beispiele: "Hallo", 'Welt', "123"
- ➔ In Python
- ➔

```
# Beispiel für String (str):
name: str = "John Doe"
print("Der Name ist:", name)
```

Boolean (bool)

- ➔ Darstellung von Wahrheitswerten (True oder False).
- ➔ Beispiele: True, False
- ➔ In Python
- ➔

```
Beispiel für Boolean (bool):
is_raining: bool = True
print("Regnet es?", is_raining)
```

None

- ➔ Ein spezieller Datentyp, der das Fehlen eines Wertes oder eine leere Variable darstellt
- ➔ Beispiel: None
- ➔ In Python
- ➔

```
# Beispiel für None:
value: None = None
print("Der Wert ist:", value)
```

Dynamische Typisierung (Duck Typing)

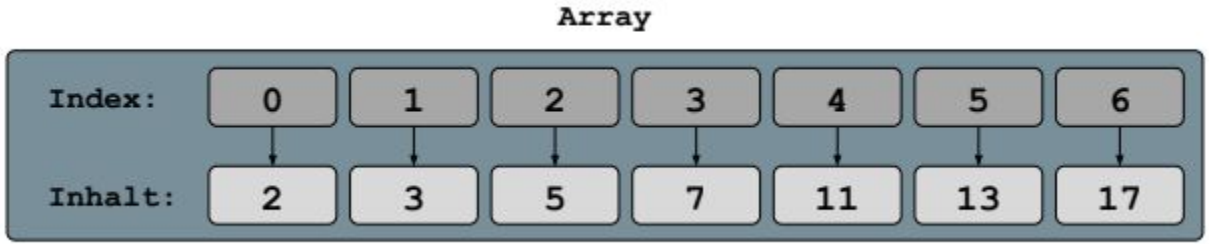
- ➔ Python ist eine dynamisch typisierte Sprache.
- ➔ Du musst den Datentyp einer Variablen nicht explizit angeben.
- ➔ Python erkennt den Datentyp automatisch anhand des zugewiesenen Werts.

Warum können wir Datentypen trotzdem besser angeben?

- ➔ Klare Absicht: Erhöht Lesbarkeit und Verständnis
- ➔ Fehlererkennung: Frühe Erkennung von Typfehlern
- ➔ Dokumentation: Selbstdokumentierender Code
- ➔ Performance-Optimierung: Effizientere Nutzung bestimmter Datentypen

Arrays in Python

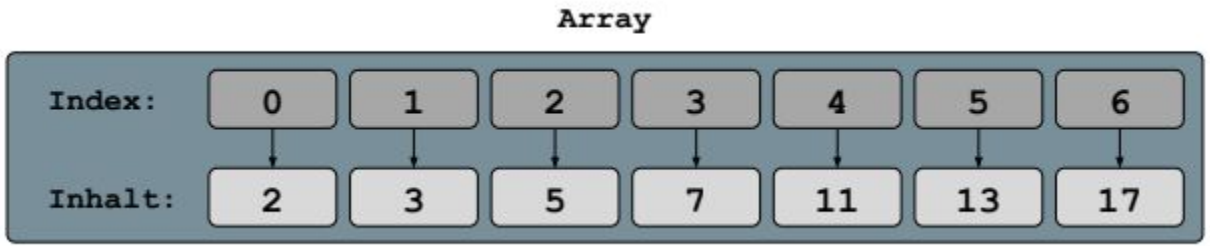
- ➔ Arrays sind spezielle Datenstrukturen, welche eine festgelegte Größe haben
- ➔ Elemente in einem Array sind in einer festen Reihenfolge angeordnet und können durch Indizes abgerufen werden



Arrays in Python

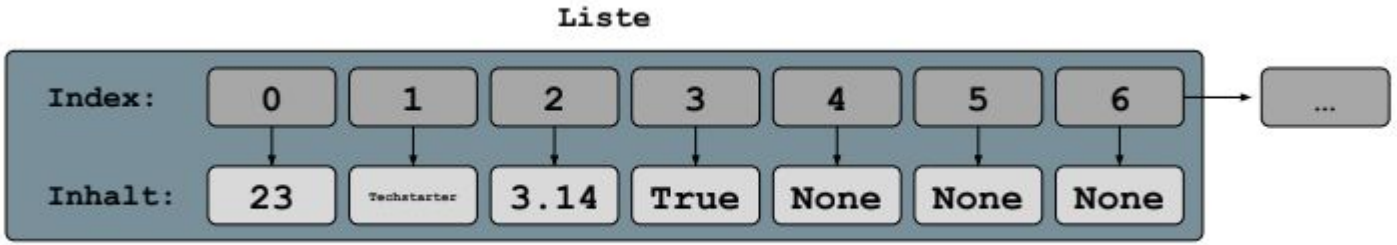
➔

```
# Beispiel für Array:
integer_array = [2, 3, 5, 7, 11, 13, 17]
print(integer_array[0])
print(integer_array[2])
```



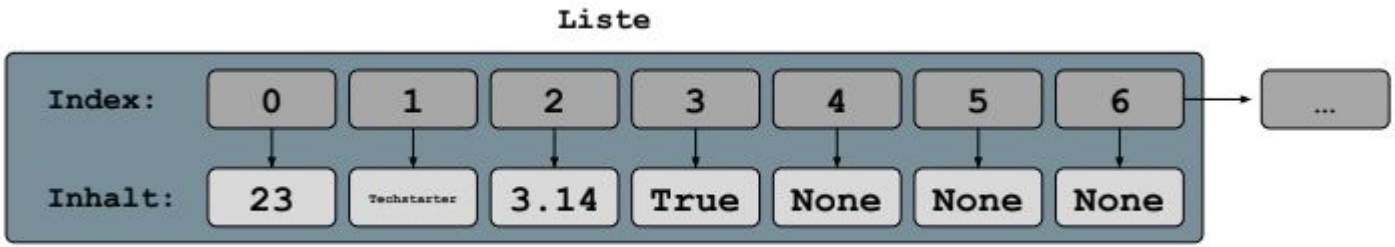
Listen in Python

- ➔ Flexible Datenstruktur für unterschiedliche Datentypen
- ➔ Listen können verändert werden und ermöglichen das Hinzufügen, Entfernen und Ändern von Elementen
- ➔ Elemente in einer Liste sind in einer festen Reihenfolge angeordnet und können durch Indizes abgerufen werden



Listen in Python

```
# Beispiel für Liste:
my_list = [23, "Techstarter", 3.14, True, None, None, None]
print(my_list[0])
print(my_list[1])
```



Hinzufügen eines Elements zur Liste

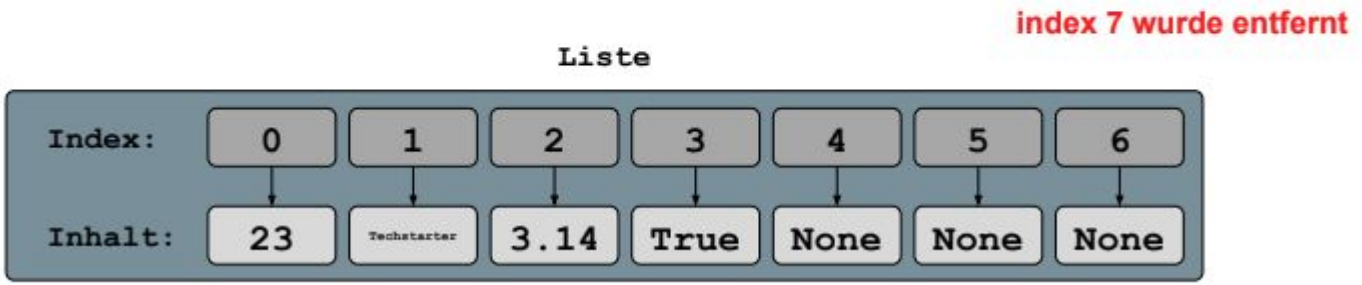
- In Python

```
# Hinzufügen eines Elements zur Liste
my_list.append("Moin!")
```

Entfernen eines Elements aus der Liste

- In Python

```
# Entfernen des Elements aus der Liste
removed_element = my_list.pop(7)
```



Beispiel

➔ Schreibe ein Python-Programm mit den folgenden Schritten

- Definiere eine Variable mit dem Namen name und weise ihr Namen als String zu
- Definiere eine Variable age und weise ihr dein Alter als Ganzzahl (integer) zu
- Definiere eine Variable height und weise ihr deine Körpergröße in Metern als Fließkommazahl (float) zu
- Definiere eine Variable is_student und weise ihr den Wert True zu

Beispiel

➔ Schreibe ein Python-Programm mit den folgenden Schritten

- Definiere eine Variable mit dem Namen name und weise ihr Namen als String zu
- Definiere eine Variable age und weise ihr dein Alter als Ganzzahl (integer) zu
- Definiere eine Variable height und weise ihr deine Körpergröße in Metern als Fließkommazahl (float) zu
- Definiere eine Variable is_student und weise ihr den Wert True zu

Beispiel

➔ Schreibe ein Python-Programm mit den folgenden Schritten

- Definiere eine Variable mit dem Namen name und weise ihr Namen als String zu
- Definiere eine Variable age und weise ihr dein Alter als Ganzzahl (integer) zu
- Definiere eine Variable height und weise ihr deine Körpergröße in Metern als Fließkommazahl (float) zu
- Definiere eine Variable is_student und weise ihr den Wert True zu

```python
## Beispiel Python Skript
# Definiere eine Variable mit dem Namen name und weise ihr Namen als String zu
name: str = "Helen Haveloh"
# Definiere eine Variable age und weise ihr dein Alter als Ganzzahl (integer) zu
age: int = 26
# Definiere eine Variable height und weise ihr deine Körpergröße in Metern als Fließkommazahl (float) zu
height: float = 1.80
# Definiere eine Variable is_student und weise ihr den Wert True zu
is_student: bool = True

```

Installation VSCode und Python

- ➔ Installiere VSCode mit choco install vscode
- ➔ Installiere Python mit choco install python
  - ◆ Überprüfe in der Eingabeaufforderung mit python –version, ob es funktioniert hat
- ➔ Starte VSCode und suche in Extensions nach Python und installiere diese
- ➔ Öffne die Kommandopalette mit STRG+SHIFT+P und gib Python: Select Interpreter ein und wähle den installierten Python-Interpreter aus
- ➔ Gehe zu File > New File (oder STRG+N). Speichere die Datei mit der Dateiendung .py (z.B. hello.py)
- ➔ Füge den folgenden Code ein
  - ◆ print(“Hello World!”)
- ➔ Speichere die Datei mit STRG+S, Öffne das Terminal in VSCode, stelle sicher, dass du im richtigen Verzeichnis mit der Python-Datei bist. Gib dann im Terminal python hello.py ein und drücke Enter.
- ➔ Es sollte ein Hello World! erscheinen

Beispielaufgabe

```python
# Definiere zwei Ganzzahlen a und b mit den Werten 10 und 3.
a = 10
b = 3
# Berechne und gib die folgenden Werte aus:
# Die Summe von a und b
print(a + b)
# Die Differenz von a und b
print(a - b)
# Das Produkt von a und b
print(a * b)
# Den ganzzahligen Quotienten von a und b
print(a // b)
# Den Rest der Division von a und b
print(a % b)
# Den Wert von a hoch b
print(a ** b)
# Definiere eine Fließkommazahl c mit dem Wert 4.5 und einen String d mit dem Wert "Python".
c = 4.5
d = "Python"
# Kombiniere c und d zu einem String und gib das Ergebnis aus.
print(str(c) + d)
# Wandle c in eine Ganzzahl um und gib das Ergebnis aus.
print(int(c))
# Vergleiche a und b und gib aus, ob a größer, kleiner oder gleich b ist.
if a > b:
    print("a ist größer als b")
elif a < b:
    print("a ist kleiner als b")
else:
    print("a ist gleich b")
# Gib aus, ob a durch 2 teilbar ist.
if a % 2 == 0:
    print("a ist durch 2 teilbar")
else:
    print("a ist nicht durch 2 teilbar")
```

TECH STARTER

Schreibe ein Python-Programm, das folgende Schritte ausführt:

1. Definiere zwei Ganzzahlen a und b mit den Werten 10 und 3.
2. Berechne und gib die folgenden Werte aus:
    a. Die Summe von a und b
    b. Die Differenz von a und b
    c. Das Produkt von a und b
    d. Den ganzzahligen Quotienten von a und b
    e. Den Rest der Division von a und b
    f. Den Wert von a hoch b
3. Definiere eine Fließkommazahl c mit dem Wert 4.5 und einen String d mit dem Wert "Python".
4. Kombiniere c und d zu einem String und gib das Ergebnis aus.
5. Wandle c in eine Ganzzahl um und gib das Ergebnis aus.
6. Vergleiche a und b und gib aus, ob a größer, kleiner oder gleich b ist.

input-Befehl

Der input-Befehl in Python wird verwendet, um Benutzereingaben zu erfassen. Wenn dieser Befehl aufgerufen wird, hält das Programm an und wartet darauf, dass der Benutzer eine Eingabe macht und die Eingabetaste drückt. Die eingegebene Zeichenfolge wird dann als Ergebnis des input-Befehls zurückgegeben und kann in einer Variable gespeichert werden.

```
variable_name = input("Eingabeaufforderung: ")
```

input-Befehl

```
# Benutzer nach seinem Namen fragen
name = input("Wie heißt du? ")

# Begrüßungsnachricht ausgeben
print("Hallo, " + name + "! Schön dich kennenzulernen.")
```

Schreibe ein kurzes Programm, das den Benutzer nach seinem Namen fragt und dann eine

Begrüßungsnachricht ausgibt:

input-Befehl

```
# Benutzer nach seinem Namen fragen
name = input("Wie heißt du? ")

# Benutzer nach seinem Lieblingsessen fragen
lieblingsessen = input("Was ist dein Lieblingsessen? ")

# Persönliche Nachricht ausgeben
print("Hallo " + name + ", es ist toll zu wissen, dass du " + lieblingsessen
+ " liebst!")
```

Überlege dir ein eigenes kleines Programm, bei dem der input-Befehl mindestens zweimal

verwendet wird. Ein Beispiel könnte ein kleines Quiz oder ein Programm sein, das

persönliche Daten sammelt und eine Zusammenfassung ausgibt.

Argumente übergeben an Funktionen

```python
def begruessung(name):
    print(f"Hallo {name}! Willkommen zum Python-Programmieren")

begruessung("Helen")
```

Wie rufe ich das ganze dann auf?

→ Mit begruessung(“Helen”)