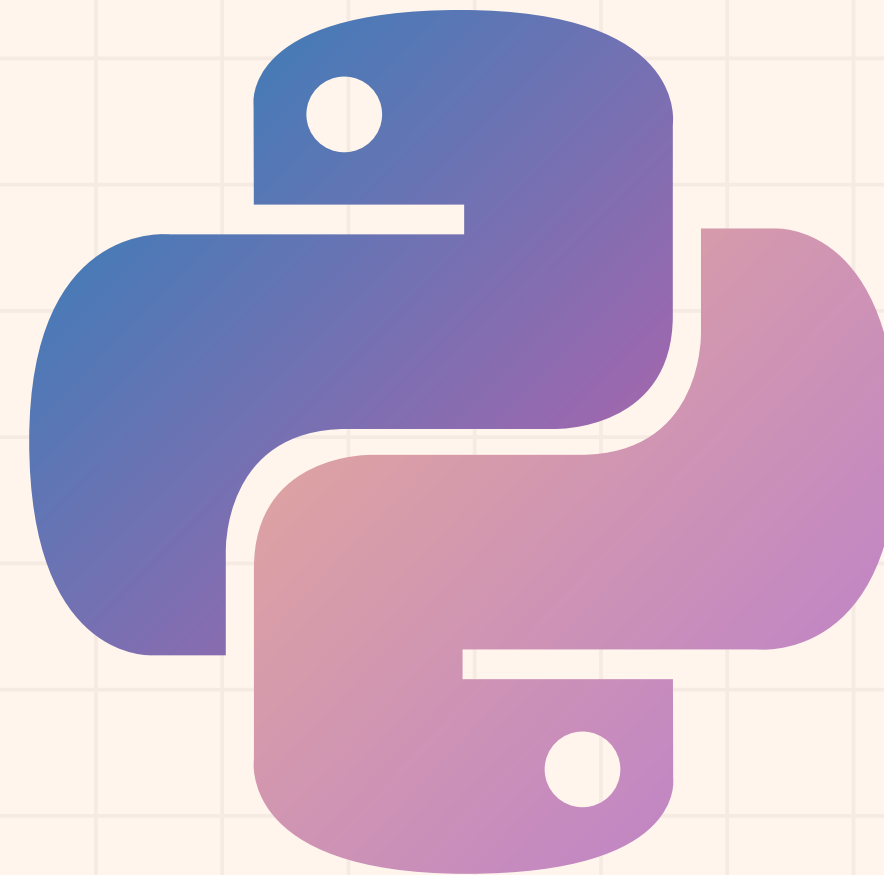


PYTHON

101

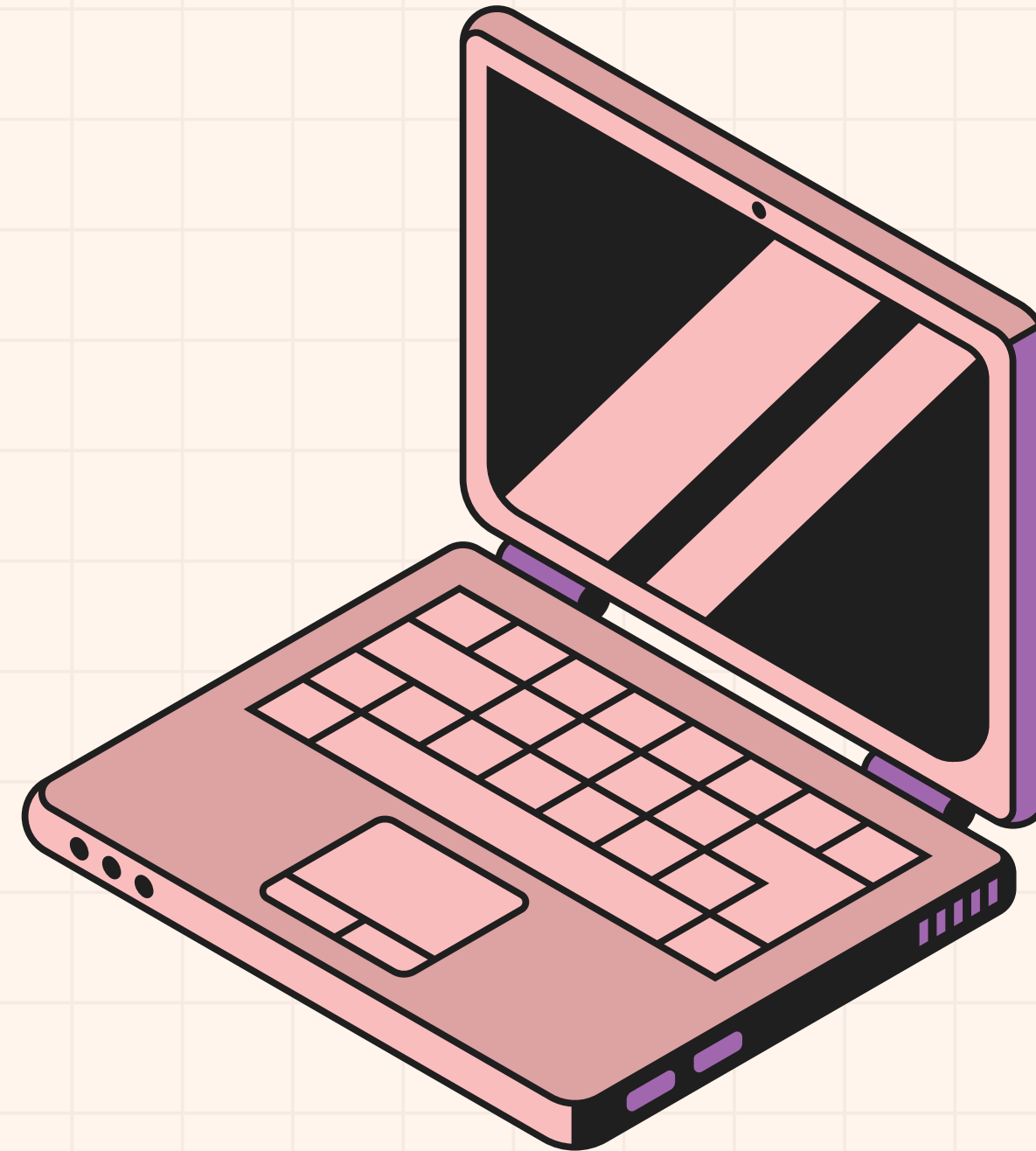
Deine ersten Schritte in die Python Programmierung



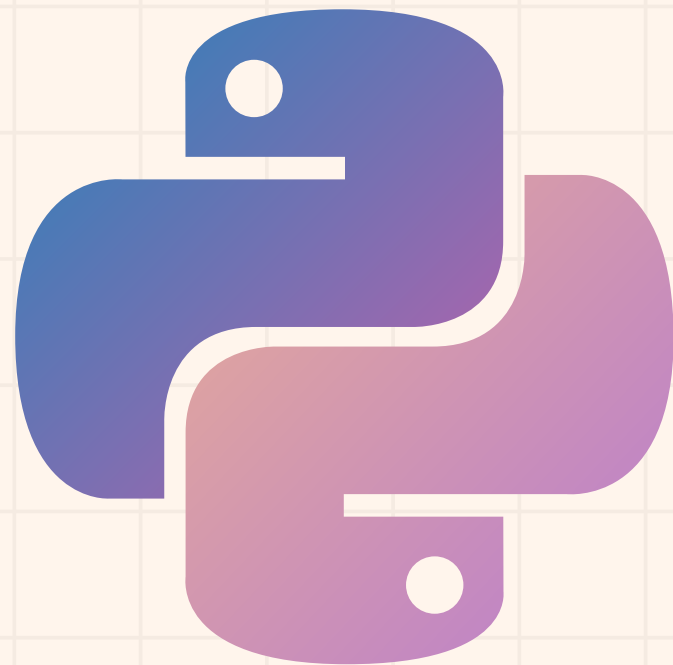
EINFÜHRUNG

Was ist Python?

- Einfache, einsteigerfreundliche Programmiersprache
- Weit verbreitet in Web, Datenanalyse, KI und Automatisierung
- Kostenlos und Open Source

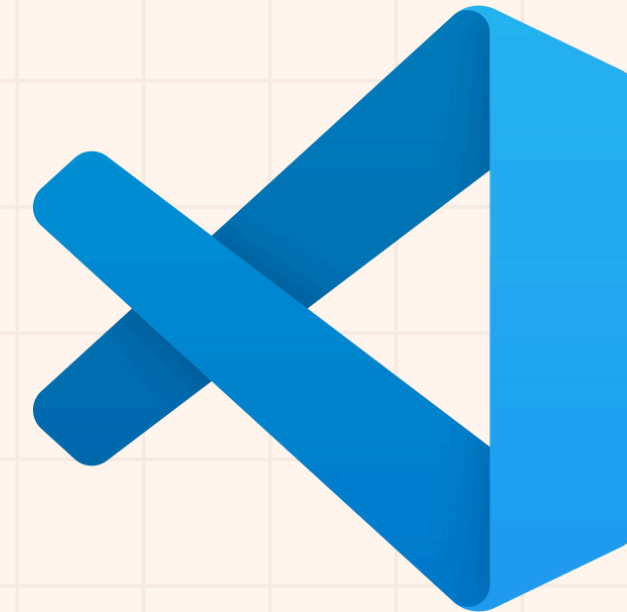


WORKFLOW



CODECADET

Auf <https://codecadet.lovable.app/> findet ihr alle aufgaben und Befehle beschrieben. Ihr löst die Aufgaben und Fragt uns nach Bedarf nach Lösung, Hilfe oder Korrektur. Selbstkorrektur ist möglich mit dem vorgegeben Output



VS CODE

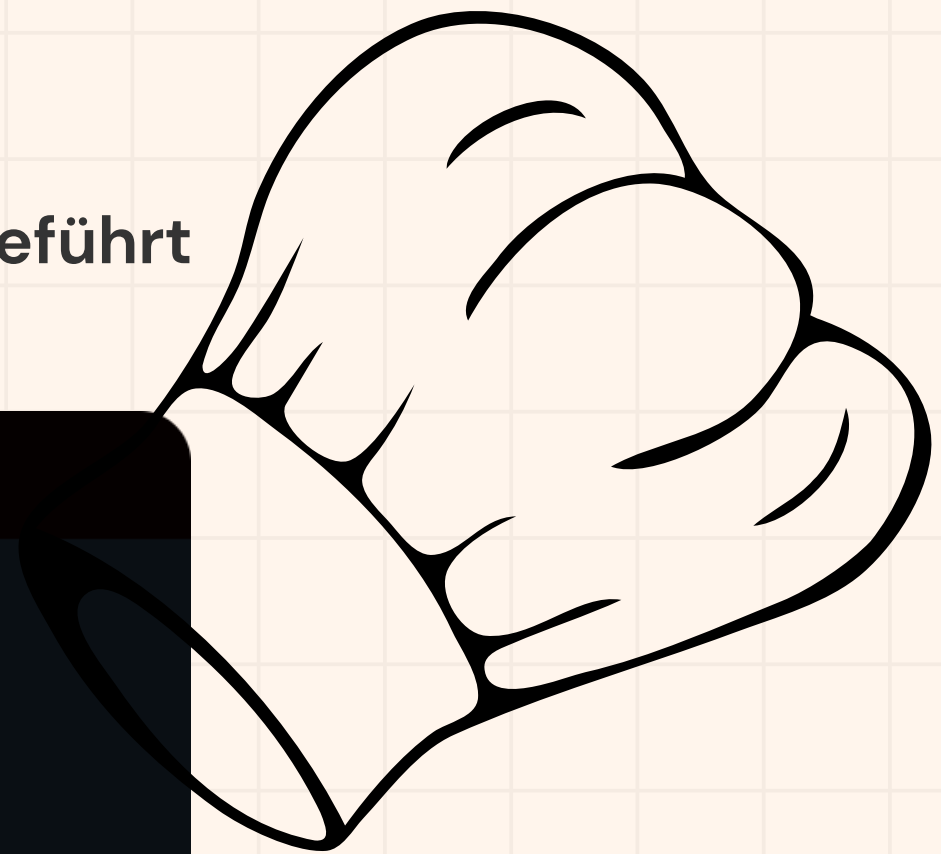
Ihr bearbeitet die Aufgaben auf VS code und lädt uns ein wenn ihr Hilfe braucht über live share

INSTALLIEREN

- 1. Visual Studio Code installieren
 - Gehe auf <https://code.visualstudio.com/download>
 - Lade die Mac-Version herunter (Intel oder Apple Silicon beachten)
 - Entpacke die .zip-Datei → Visual Studio Code.app in Programme ziehen
 - VS Code starten (bei Warnung „Öffnen“ klicken)
- 2. Python installieren
 - Gehe auf www.python.org/downloads
 - Lade den passenden macOS-Installer (pkg-Datei) herunter
 - Installer starten → Anweisungen folgen
 - Danach im Terminal prüfen:
 - `python3 --version`
- 3. Python-Extension in VS Code
 - In VS Code links auf das Extensions-Symbol klicken
 - Nach Python suchen → offizielle Microsoft-Extension installieren
- 4. Erste Datei erstellen & ausführen
 - Neue Datei: `hello.py`
 - Beispielcode:
 - `print("Hallo, Welt!")`
- Ausführen:
 - Unten auf „▶ Run Python File“ klicken
- 5. Hilfe
 - fügt über die extensions ‘live share’ hinzu und schickt uns den Link per E-Mail

FUNKTIONSWEISE VON PYTHON

- Eine Zeile = ein Befehl
- Befehle werden von oben nach unten ausgeführt



```
befehl 1  
befehl 2  
befehl 3
```

DIE ERSTE ZEILE HELLO WORLD

- Datei erstellen: hello.py
- Code: `print("Hallo, Welt")`
- Ausführen



```
print('hello World')
```



```
hello world
```

VARIABLEN

- Werte im Speicher ablegen
- Machen Programme flexibel und wiederverwendbar
- kann sie einfach mit Print ausgeben

```
name = 'Anna'  
age = 20  
print(name, age)
```

```
Anna20
```

DATENTYPEN SPEICHER

- Int = Buchstaben in Zahlen umwandeln
- float = Buchstaben in Kommazahlen umwandeln
- bool = Wahrheitswerte speichern
(True oder False)
- str = Zeichenketten (Texte) speichern
- list = Listen erstellen

Ganze Zahlen

Dezimalzahlen

```
alter = 25 # int
temperatur = 18.5 # float
ist_sonnig = True # bool
stadt = "Berlin" # str
zahlen = [1, 2, 3, 4] # list
```

```
int (alter)
float (temperaturen)
bool (ist_sonnig)
stadt: str = "Berlin"
namen = ["java", "python", "HTML"]
```

ist_sonnig = True

PRINTEN 2.0

VARIABLEN UND F-STRINGS

- Ermöglichen es, Text und Variablen direkt zu verbinden
- Syntax: ein f vor dem String setzen
- Variablen in geschweiften Klammern {} einfügen

```
name = "Anna"  
alter = 20  
print(f"Hallo {name}, du bist {alter} Jahre alt.")
```

```
Hallo Anna, du bist 20 Jahre alt.
```

EVA UND ADAM

- Eingabe vom Benutzer: `input("Wie heißt du? ")`
- muss als Variable gespeichert werden
- Ausgabe auf den Bildschirm:
`print("Hallo,", name)`

```
name = input('Wie heißt du? ')\nprint('Hallo, ', name)
```

```
Wie heißt du? #Peter\nHallo, Peter
```

FUNKTIONEN

- Eingebaute Funktionen: len(), round()
- Eigene Funktionen mit def definieren
- hat einen Namen und ein Attribut

```
def begruessung(name):  
    print("Hallo,", name)  
begruessung('Anna')
```

Hallo Anna

BEDINGUNGEN

- Vergleichsoperatoren:

- ==
- !=
- >
- <

- Befehle:

- if (wenn)
- else(sonst)
- elif(sonst-wenn)

```
if alter >= 18:  
    print("Erwachsen")  
else:  
    print("Minderjährig")
```

SCHLEIFEN

- for-Schleifen: über eine Sequenz laufen
- while-Schleifen: solange Bedingung wahr ist
- for i in range(10)
- für jedes i (index/Zahl/Element) in range(10)
- range(10) ist Liste aus [1, 2, ..., 10]

```
if alter >= 18:  
    print("Erwachsen")  
else:  
    print("Minderjährig")
```

SAMMLUNGEN VON DATEN

- Listen
- Wörterbücher (Dictionaries)
- Praktisch zum Speichern mehrerer Werte

```
list = ["Apfel", "Banane"]  
dictionary = {"name": "Anna", "alter": 20}
```

UMGANG MIT DATEN

LISTEN

- Geordnete Sammlung von Werten
- Elemente können mehrfach vorkommen
- Zugriff über Index (Start bei 0)

```
fruechte = ["Apfel", "Banane", "Kirsche"]  
  
print(fruechte[0])  
fruechte.append("Mango")  
print(fruechte)
```

```
apfel
```

```
["Apfel", "Banane", "Kirsche", "Mango"]
```

UMGANG MIT DATEN

DICTIONARIES

- Sammlung von Schlüssel-Wert-Paaren
- Schlüssel (key) eindeutig, Werte frei wählbar
- Zugriff über den Schlüssel

```
person = {"name": "Anna", "alter": 20}

print(person["name"])
person["stadt"] = "Berlin"
print(person)
```

```
anna
```

```
{"name": "Anna", "alter": 20, "stadt": "Berlin"}
```


METHODEN FÜR LISTEN

- `append(x)` → hängt ein Element an
- `insert(i, x)` → fügt Element an Index ein
- `remove(x)` → entfernt erstes Vorkommen von `x`
- `pop(i)` → entfernt und gibt Element zurück (Standard: letztes)
- `len(liste)` → gibt Anzahl der Elemente zurück

```
zahlen = [1, 2, 3]
zahlen.append(4)      # [1, 2, 3, 4]
zahlen.pop()          # entfernt 4
```

METHODEN FÜR DICTIONARIES

- `dict["key"]` → Wert über Schlüssel abrufen
- `dict["key"] = x` → neuen Wert setzen / hinzufügen
- `dict.pop("key")` → entfernt Schlüssel und gibt Wert zurück
- `keys()` → alle Schlüssel
- `values()` → alle Werte

```
person = {"name": "Anna", "alter": 20}
person["stadt"] = "Berlin"
person.pop("alter")    # entfernt "alter"
```

DIE NORMEN VON PYTHON

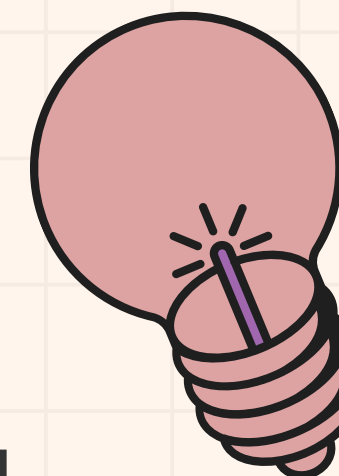
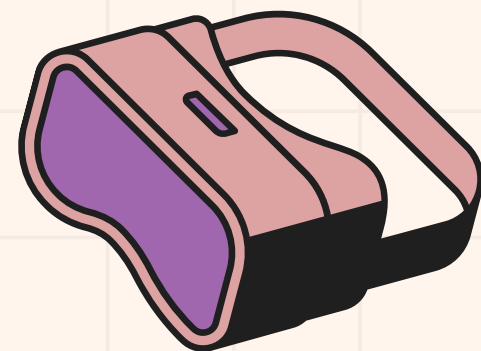
- Einrückungen (Spaces/Tabs) zeigen Blöcke (if, for, def)
- Leerzeichen vor und nach operatoren wie =, ==
- Sprechende Variablennamen statt a, b, c
- Kommentare mit # für Erklärungen
- Ordnung & Lesbarkeit › Abkürzungen
- Variablen- oder Dateinamen in Englisch

DIE NORMEN VON PYTHON

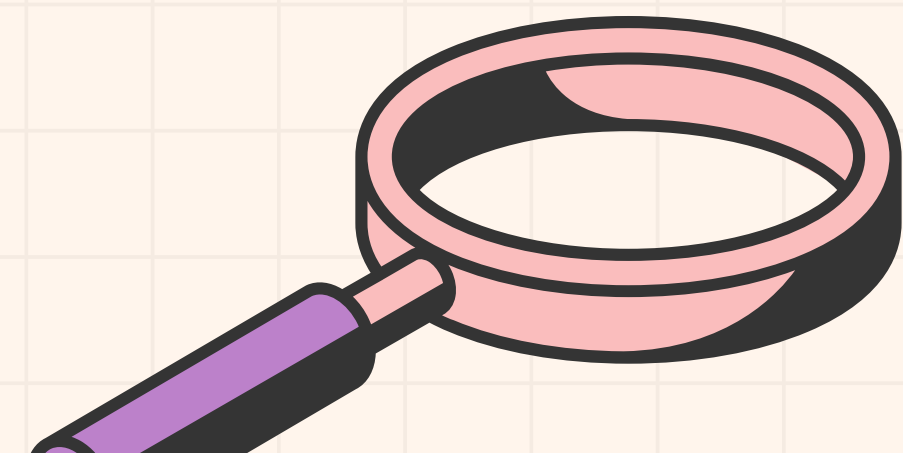
```
zahlen=[1,2,3,4,5]
for i in zahlen:
    if i%2==0:
        print(i,"ist gerade")
    else:
        print(i,"ist ungerade")
```

```
# Iteriere durch eine Liste von Zahlen
zahlen = [1, 2, 3, 4, 5]

for zahl in zahlen:
    if zahl % 2 == 0:
        print(zahl, "ist gerade")
    else:
        print(zahl, "ist ungerade")
```



THANK YOU



C