Python Cheat Sheet für Einsteiger

1. Konsolen Ein- und Ausgabe

- **Ausgabe** mit `print()`:

```
print("Hallo, Welt!")  # Gibt einen Text auf der Konsole aus
```

- **Eingabe** mit `input()`:

```
name = input("Wie heißt du? ")  # Liest einen String vom Benutzer ein
print(f"Hallo, {name}!")
```

---

2. Variablen & Datentypen

- **Deklaration:** Dynamisch typisiert, kein `;` nötig:

```
zahl = 5         # Integer
text = "Python"  # String
pi = 3.14       # Float
wahr = True     # Boolean
```

- **Typumwandlung:**

```
int("42")    # -> 42
float("3.14") # -> 3.14
str(100)      # -> "100"
bool(0)       # -> False
```

---

3. Listen (`list`)

- **Erstellen** und **Zugriff**:

```
fruits = ["Apfel", "Banane", "Kirsche"]
print(fruits[0])    # "Apfel"
print(fruits[-1])   # "Kirsche" (letztes Element)
```

- **Methoden:**

```
fruits.append("Orange")  # Hinzufügen
fruits.insert(1, "Mango")  # Einfügen an Index 1
fruits.remove("Banane")  # Entfernen nach Wert
fruits.pop()             # Letztes Element entfernen
fruits.sort()            # Sortieren
```

---

4. Dictionaries (`dict`)

- **Erstellen** und **Zugriff**:

```
person = {"Name": "Max", "Alter": 30}
print(person["Name"])  # "Max"
```

- **Methoden:**

```
person.keys()     # Alle Schlüssel
person.values()   # Alle Werte
person.items()    # Schlüssel-Wert-Paare
person.get("Alter", 0)  # Sicherer Zugriff mit Default
```

---

5. Bedingte Anweisungen

5.1 `if` / `elif` / `else`

```
if bedingung:
    # Code, wenn wahr
elif andere_bedingung:
    # Code, wenn andere wahr
else:
    # Code, wenn keine wahr
```

5.2 `match` / `case` (ab Python 3.10)

```
match wert:
    case 1:
        print("Eins")
    case 2:
        print("Zwei")
    case _:
        print("Anderer Wert")  # Default-Fall
```

---

6. Schleifen

6.1 `while`-Schleife

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

6.2 `for`-Schleife

```
for i in range(5):  # 0,1,2,3,4
    print(i)
```

6.3 `for each` (Iterieren über Collections)

```
for fruit in fruits:
    print(fruit)
```

---

7. Funktionen

- **Definition** & **Aufruf**:

  ```
  def begruessung(name: str) -> None:
      """Gibt eine Begrüßung aus"""
      print(f"Hallo, {name}!")

  begruessung("Anna")
  ```

- **Parameter** und **Rückgabe**:

  ```
  def addiere(a: int, b: int) -> int:
      return a + b

  ergebnis = addiere(3, 4)  # 7
  ```

8. Wichtige String-Funktionen

Methode	Beschreibung	Beispiel
`.upper()`	In Großbuchstaben umwandeln	`"hallo".upper()` → "HALLO"
`.lower()`	In Kleinbuchstaben umwandeln	`"HALLO".lower()` → "hallo"
`.strip()`	Leerzeichen am Anfang/Ende entfernen	`" text ".strip()` → "text"
`.split(sep)`	In Liste anhand Separator aufteilen	`"a,b,c".split(",")` → ['a','b','c']
`.replace(a,b)`	Teilstring ersetzen	`"Hi".replace("H","J")` → "Ji"

9. Operatoren & Vergleich

- **Arithmetik:** `+`, `-`, `*`, `/`, `//` (Ganzzahl), `%` (Modulo), `**` (Potenz)
- **Vergleich:** `==`, `!=`, `<`, `>`, `<=`, `>=`
- **Logisch:** `and`, `or`, `not`
- **Mit `` prüfen:**

```
"a" in "Hallo"  # True
3 in [1,2,3]      # True
```

10. Weitere wichtige Keywords

- `import`, `from ... import`: Module einbinden
- `as`: Alias für Module/Funktionen
- `lambda`: anonyme Funktion

```
quad = lambda x: x**2
print(quad(5))  # 25
```

- `class`: Klassen definieren (OOP)
- `try / except / finally`: Fehlerbehandlung
- `with`: Kontextmanager (z.B. Dateizugriff)

```
with open("file.txt") as f:
    data = f.read()
```

Dieses Cheat Sheet bietet einen schnellen Überblick – zum tieferen Eintauchen lohnt es sich, die offiziellen Python-Dokumentationen zu erkunden.