🌀 Kontrollstrukturen in Python – Ausführliche Erklärung

1. ◇ Bedingte Anweisungen (`if`, `elif`, `else`)

Kontrollieren den Programmfluss je nach Bedingung.

◇ Einfache `if`-Bedingung

```
x = 10
if x > 5:
    print("x ist größer als 5")
```

➡️ Der Codeblock wird **nur ausgeführt**, wenn die Bedingung `x > 5` **True** ergibt.

◇ `if` mit `else`

```
x = 3
if x > 5:
    print("x ist größer als 5")
else:
    print("x ist kleiner oder gleich 5")
```

➡️ Falls die erste Bedingung nicht zutrifft, wird der `else`-Zweig ausgeführt.

◇ `if` mit `elif`

```
note = 2

if note == 1:
    print("Sehr gut")
elif note == 2:
    print("Gut")
elif note == 3:
    print("Befriedigend")
else:
    print("Nicht bestanden")
```

➡️ `elif` bedeutet: "falls die vorherigen Bedingungen **nicht** zutreffen, dann prüfe diese".

◇ Verschachtelte `if`-Bedingung (nested `if`)

```
x = 10
if x > 0:
    if x % 2 == 0:
        print("x ist positiv und gerade")
    else:
        print("x ist positiv aber ungerade")
else:
    print("x ist nicht positiv")
```

➡ Eine `if`-**Anweisung innerhalb einer anderen**.

---

2. 🔁 Wiederholungen mit Schleifen

◇ `for`-Schleife: Iterieren über eine Sequenz

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

➡ Nützlich, wenn man weiß, **wie oft** etwas wiederholt werden soll.

◇ `while`-Schleife: Solange eine Bedingung wahr ist

```
i = 0
while i < 5:
    print("i ist:", i)
    i += 1
```

➡ Nutzt man, wenn man **nicht vorher weiß**, wie oft der Code laufen wird.

◇ Schleifen mit `break` und `continue`

`break` **– beende die Schleife**

```
for i in range(10):
    if i == 3:
        break
    print(i)
```

`continue` **– überspringe den aktuellen Durchlauf**

```
```

```python
for i in range(5):
    if i == 2:
        continue
    print(i)
```

---

3. 🎯 `match`-Anweisung (ab Python 3.10)

Ersetzt `switch-case` aus anderen Sprachen.

```python
befehl = "start"

match befehl:
    case "start":
        print("Programm startet")
    case "stop":
        print("Programm stoppt")
    case _:
        print("Unbekannter Befehl")
```

➡️ `case _:` ist der **Standardfall** (wie `default` in C/C++/Java).

---

4. 📌 Vergleichsoperatoren und logische Operatoren

Operator	Bedeutung	Beispiel
`==`	gleich	`x == y`
`!=`	ungleich	`x != y`
`<`, `>`, `<=`, `>=`	kleiner, größer, ...	`x < 10`
`and`	logisches UND	`x > 0 and x < 5`
`or`	logisches ODER	`x < 0 or x > 10`
`not`	logisches NICHT	`not (x == 5)`

---

5. 🧠 Wichtig: Einrückung ist Pflicht!

```python
if True:
    print("Dieser Code ist korrekt eingerückt")
# Kein { } – Python nutzt Einrückung (4 Leerzeichen empfohlen)
```

☑️ Fazit

- `if`/`elif`/`else`: Entscheidung je nach Bedingung
- `for`: Wiederhole über Sequenzen (z. B. `range`)
- `while`: Wiederhole solange eine Bedingung wahr ist
- `match`: Alternative zu vielen `if`-Abfragen bei bestimmten Werten