Was bisher geschah…

- Algorithmus in 3 Schritte aufteilen:
  - Eingabe
  - Verarbeitung
  - Ausgabe
- Python setzt auf Duck Typing:
  - Datentyp durch Vorhandensein bestimmter Attribute & Methoden bestimmt
- Grundlegende Datentypen:
  - Boolean
  - Integer
  - Float
  - String

Verzweigungen

- Problem: Aktuell wird unser Code linear abgearbeitet
- Verzweigungen erlauben mehr Flexibilität:

„Mache etwas wenn eine bestimmte Bedingung erfüllt ist ansonsten mache etwas anderes.“

```
if Bedingung:
        etwas
else:
        etwas anderes
```

Verzweigungen - Beispiel

```
process_running = True


if process_running:
        stop(process)
else:
        start(process)

# auch möglich: if process_running == True
```

Vergleichsoperatoren

Syntax	Bedeutung	
==	Ist-Gleich	
!=	Ungleich	
<	Kleiner als	
<=	Kleiner als oder gleich	(Höchstens)
>	Größer als	
>=	Größer als oder gleich	(Mindestens)

- Ergebnisse sind immer boolescher Wert (True/False)

Vergleichsoperatoren - Beispiele

„Wenn a und b gleich groß sind, erhöhe a um 100. Ansonsten schreibe b in die Konsole.“

```
if a == b:
        a = a + 100
else:
        print(b)
```

Vergleichsoperatoren - Beispiele

„Wenn Lauras Gehalt niedriger ist als Marias, dann verdopple es. Ansonsten halbiere es.“

```
if gehalt_laura < gehalt_maria:
        gehalt_laura = gehalt_laura * 2
else:
        gehalt_laura = gehalt_laura / 2
```

Vergleichsoperatoren - Beispiele

„Wenn auf Antonios PC mindestens 10 Prozesse laufen, fahre ihn herunter. Ansonsten mache nichts.“

```
if countProcesses(Antonio_PC) >= 10:
        shutdown(Antonio_PC)
```

Nochmal zu Verzweigungen

- Einrückung beachten!
  - Alles was passieren soll, wenn Bedingung erfüllt -> Einrücken!
  - Alles was ansonsten (=else) passieren soll -> Einrücken!
- Zusatz: elif
  - Wenn erste Bedingung (=if) nicht wahr, prüfe weitere Bedingung (=elif)
  - Beliebig oft wiederholbar
- Ausnahme beim Einrücken: Kurzes if
  - Möglichkeit 1: One Liner
    if a > b: print("a ist größer als b")
    else: print("a ist nicht größer als b")
  - Möglichkeit 2: Ternary Operator
    print("a ist größer als b") if a > b else print("a ist nicht größer als b")

Mehrere Bedingungen

- Logische Operatoren: Erlauben mehrere Bedingungen zu prüfen

Syntax	**Bedeutung**	
and	Beide Bedingungen wahr	
or	Mind. Eine Bedingung wahr	Nicht verwechseln mit umgangssprachlichem oder!
not	Umdrehen des Wahrheitswerts	

Mehrere Bedingungen

- Wie prüft Python verknüpfte Bedingungen? Mit Wahrheitstabellen!

Aussage 1 (A1)	**Aussage 2 (A2)**	**A1 und A2**	**A1 oder A2**
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Logische Operatoren - Beispiele

„Wenn die Außentemperatur mind. 20 °C beträgt und die Sonne scheint, dann schreibe einen fröhlichen Smiley in die Konsole. Ansonsten schreibe einen traurigen Smiley.“

```
if current_temperature >= 20 and sun_shining == True:
        print(":)")
else:
        print(":(")
```

Logische Operatoren - Beispiele

„Wenn die Zahl a oder die Zahl b positiv ist, schreibe beide in die Konsole.“

```
if a > 0 or b > 0:
        print(a, b)


# Deutsche Sprache nicht eindeutig! Könnte auch so gemeint sein:
if (a > 0 and b <= 0) or (b > 0 and a <= 0):
        print(a, b)
```

Zusatz: Identitätsoperatoren

- Prüfen, ob zwei Werte die selbe Speicheradresse haben:

Beispiel	**Bedeutung**
A is B	A und B haben die gleiche Speicheradresse
A is not B	A und B haben unterschiedliche Speicheradressen

A is B ist äquivalent zu id(A) == id(B)

Übungen zu Verzweigungen, log. Operatoren etc.

- Siehe 05 – Übung.py