

Informatik		Datum:
Name:	Klasse:	

## Lektion 2: Auswahl (if-else-selection)

### **Predict** (Vorhersagen)

```
# starter program

def conversation():
    print("Welcome to my conversation program")
    print()
    print("Do you like cycling? Answer yes or no")
    answer = input()
    if answer == "yes":
        print("That's good - you will get very fit")
    else:
        print("Perhaps you like some other sport. ")
    print("Goodbye")
```

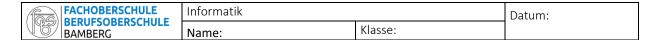
#### Run (Ausführen)

Führen Sie das Programm oben aus.

- Entspricht das Verhalten Ihrer Vermutung?
- Falls nicht: Was genau war anders? Schreiben Sie sich stichpunktartig Notizen.

### **Investigate** (Untersuchen)

- 1. Was würde passieren, wenn das Programm das Wort Print() statt print() verwenden würde?
- 2. Was passiert, wenn Sie beim Ausführen "Yes" statt "yes" oder "no" eingeben? (Probieren Sie es aus.)
- 3. Wozu wird der Befehl 'def' verwendet?
- 4. Erläutern Sie kurz, warum das Programm conversation () am Ende beinhaltet?



5. Warum steht am Ende der `if`-Anweisung und bei `else` jeweils ein Doppelpunkt (:)? Welche Rolle spielt die Einrückung?

6. Was ist der Unterschied zwischen = und ==? Geben Sie je ein Beispiel aus dem Code an.

7. Welchen Unterschied macht es, ob man answer = input() schreibt oder answer = input("Select 1 or 2...")?

8. Warum ist print ("Goodbye") links von der Code-Zeile darüber? Was passiert, wenn Sie die Einrückung **ändern**?

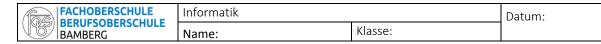
### M - Verändern (Modify)

- 1. Ändern Sie das Programm so, dass es eine **Unterhaltung über Sport** führt. Sie können eigene Fragen ergänzen.
- 2. Schreiben Sie eine Funktion 'cities()' und fragen Sie: "How many cities are there in England?" Verwenden Sie 'int(input(...))'. Die Funktion 'int(...)' wandelt die Eingabe (String) in eine ganze Zahl um. (Hinweis: Die korrekte Antwort ist 51)
- 3. Schreiben Sie ein Programm, das die Benutzerin/den Benutzer bittet, eine Zahl **größer als 10** einzugeben. Geben Sie eine Rückmeldung, ob die Eingabe gültig war (denken Sie an int (input ())).
- 4. Fragen Sie nach der **Punktzahl** in einem Test. Ist sie **kleiner als 50**, dann geben Sie die Rückmeldung, dass der Test nicht bestanden wurde.
- 5. Fragen Sie nach dem **Alter**. Ab **13 Jahren** gibt es die Rückmeldung, dass man eine **Zeitung austragen** darf. (Verwenden Sie >=.)
- 6. Bitten Sie um eine beliebige Zahl. Solange die Zahl nicht 99 ist (!= 99), ist alles in Ordnung.

#### M – Erstellen (Make)

Schreiben Sie ein Programm, das prüft, ob eine Zahl zwischen 1 und 20 liegt (verwenden Sie and).

- Im Bereich → positives Feedback
- < 1 → "too low"</li>
- > 20 → "too high"



## Schlüsselbegriffe – Lektion 2: Auswahl (if-else-Selection)

In dieser Lektion haben Sie gelernt, wie man Programme schreibt, die **Entscheidungen treffen**. Mit if und else können Programme **unterschiedliche Aktionen ausführen**, abhängig von der Eingabe oder der Bedingung. Diese Technik wird als **Auswahl (Selection)** bezeichnet.

## Schlüsselbegriffe - Key Concepts

Begriff	Bedeutung	Wie in Python?
Auswahl (Selection)	Ermöglicht es einem Programm, zwischen verschiedenen Handlungswegen zu wählen.	<pre>if answer == "Paris":     print("Correct") else:     print("Not correct")</pre>
Bedingung (Condition)	Ausdruck, der auf True oder False geprüft wird.	answer == "Paris" # to test equality number == 100 # test equality number > 100 # test greater than number < 100 # test greater than number >= 100 # test greater than or equal to number <= 100 # test greater than or equal to number != 100 # test greater than or
Verzweigung (Branching)	Teil des Programms, der je nach erfüllter Bedingung ausgeführt wird.	
Vergleichsoperatoren	Werden verwendet, um Werte zu vergleichen.	==, !=, <, >, <=, >=
Einrückung (Indentation)	Gibt an, welche Codezeilen zu einer bestimmten Anweisung gehören.	4 Leerzeichen (z.B. nach if:)
Testen	Überprüfen, ob jede mögliche Eingabe zu einem korrekten Ergebnis führt.	Systematisches Ausführen aller Pfade (hier der if-else-Pfade)

# **Zusammenfassung:**

Mit if und else können Programme intelligente Entscheidungen treffen.

Das Verstehen von Bedingungen, Vergleichen und Einrückungen ist entscheidend, um korrektes Verhalten in Programmen sicherzustellen