07c_Miniuebungen_Loesungen

0.1 Mini-Aufgaben zur Überprüfung des Verständnis: Bedingte Ausführung

Wir codieren Wochentage als Zahlen. Schreiben Sie ein Programm, dass eine Zahl zwischen 1 und 7 auf die entsprechende textuelle Repräsentation überträgt. Dabei ist es völlig willkürlich, welche Zahl welchen Wochentag repräsentiert, solange die Zuordnung eindeutig ist.

[2]:
```
day = 5

if day==1:
    print("Heute ist Montag.")
elif day==2:
    print("Heute ist Dienstag.")
elif day==3:
    print("Heute ist Mittwoch.")
elif day==4:
    print("Heute ist Donnerstag.")
elif day==5:
    print("Heute ist Freitag.")
elif day==6:
    print("Heute ist Samstag.")
elif day==7:
    print("Heute ist Sonntag.")
else:
    print("Den achten Wochentag gibt es nur nach 25 Uhr.")
```

```
Heute ist Freitag.
```

Erweitern Sie die vorherige Miniübung, indem Sie ein "Kalenderblatt" für den Monat April des Jahres 2023 ausgeben, pro Zeile sollen die Informationen "Wochentag" und "Kalendertag" erscheinen:

Der 1. April ist ein Samstag.
Der 2. April ist ein Sonntag.
⋮

Versuchen Sie mit nur einer while Schleife auszukommen, und alle bedingten Entscheidungen in den Schleifenblock zu programmieren. Um die Ausgabe schöner zu gestalten, können Sie nach jedem Sonntag eine Leerzeile einfügen. Die Modulo-Rechnung mittels des % Operators ist hier hilfreich.

Zwei Ideen führen hier zum Erfolg. Erstens zählen wir die Tage der Woche separat von den Tagen des Monats, und setzen sie alle 7 Tage zurück. Zweitens kennen wir die Startbedingung, d.h. welcher Wochentag der 1.4.2023 ist. Der Rest ist smart copy-paste des vorherigen Beispiels.

[21]:
```
day = 1
day_of_week = 1
while day <= 30:
    if day_of_week == 1:
        # Einbau von Wissen, dass 1.4. ein Samstag ist
        print("Der {:d}. April ist ein Samstag.".format(day))
    elif day_of_week == 2:
        print("Der {:d}. April ist ein Sonntag.".format(day))
    elif day_of_week == 3:
        print("Der {:d}. April ist ein Montag.".format(day))
    elif day_of_week == 4:
        print("Der {:d}. April ist ein Dienstag.".format(day))
    elif day_of_week == 5:
        print("Der {:d}. April ist ein Mittwoch.".format(day))
    elif day_of_week == 6:
        print("Der {:d}. April ist ein Donnerstag.".format(day))
    elif day_of_week == 7:
        print("Der {:d}. April ist ein Freitag.".format(day))
    day += 1
    day_of_week += 1
    # Wochenwechsel
    if day_of_week % 8 == 0:
        day_of_week = 1
```

```
Der 1. April ist ein Samstag.
Der 2. April ist ein Sonntag.
Der 3. April ist ein Montag.
Der 4. April ist ein Dienstag.
Der 5. April ist ein Mittwoch.
Der 6. April ist ein Donnerstag.
Der 7. April ist ein Freitag.
Der 8. April ist ein Samstag.
Der 9. April ist ein Sonntag.
Der 10. April ist ein Montag.
Der 11. April ist ein Dienstag.
Der 12. April ist ein Mittwoch.
Der 13. April ist ein Donnerstag.
Der 14. April ist ein Freitag.
Der 15. April ist ein Samstag.
Der 16. April ist ein Sonntag.
Der 17. April ist ein Montag.
Der 18. April ist ein Dienstag.
Der 19. April ist ein Mittwoch.
Der 20. April ist ein Donnerstag.
```

```
Der 21. April ist ein Freitag.
Der 22. April ist ein Samstag.
Der 23. April ist ein Sonntag.
Der 24. April ist ein Montag.
Der 25. April ist ein Dienstag.
Der 26. April ist ein Mittwoch.
Der 27. April ist ein Donnerstag.
Der 28. April ist ein Freitag.
Der 29. April ist ein Samstag.
Der 30. April ist ein Sonntag.
dd
```

Das ist übrigens das Beispiel, an dem der Dozent chatGTP aus didaktischer Sicht gegen die Wand gefahren hat. Sie sind herzlich eingeladen, das auszuprobieren. Behalten Sie dabei immer im Hintergrund, was Sie bereits von Python durch diesen Programmierkurs wissen.

- Write a Python code that maps the days of April 2023 to Monday, Tuesday etc
- Do this without dictionary and the calendar module
- Do this without gregorian calendar and without lists
- Do this without lists, comprehensions, the gregorian calendar, simply in very basic python
- There was an error

Bis inklusive der vierten Anfrage sind die Ergebnisse beeindruckend. chatGTP und ähnliche Lösungen scheinen extrem hilfreich. Wir halten aber fest: Die Tools können nicht dazu "überredet" werden, sich an die Didaktik dieses Kurses anzupassen, mit anderen Worten, sich beliebig dumm zu verhalten aus der Sicht einer Python Gottheit.

- Wenn Sie bereits über Vorkenntnisse in Python verfügen, dann bieten chatGTP und Vs Code und gthub copilot Lösungen an, die nicht dem erwarteten Kenntnisstand entsprechen. In diesem Fall ist die Bewertung Ihrer Codes einfach, denn Sie haben die (implizite) Aufgabenstellung nicht erfüllt, das Problem mit existierenden Bordmitteln zu lösen. Stattdessen bringen Sie eine Lösung ein, die Ihnen von einem zuverlässigen und wohlinformierten Chatbot vorgeschlagen wurde.
- Wenn Sie dem Kurs ohne Vorkenntnisse folgen, dann beinhalten (vermutlich) alle KI-Vorschläge Python-Features, die Sie noch nicht kennen können. Ihre von chatGTP übernommene Lösung einer Aufgabe ist also offensichtlich kopiert.

In der Bewertung Ihrer Leistungen macht das keinen Unterschied. Mogeln kann detektiert werden.

Weiteres Verbieten von Sprachbestandteilen zeigt die Grenzen auf:

- Write a Python code that maps the days of April 2023 to Monday, Tuesday etc. Do not use any import, any list, dict, nor comprehension, and do not use functions nor for loops nor range

Die von chatGPT vorgeschlagene Lösung beinhaltet trotzdem viele dieser Sprachbestandteile.

0.2 Impressum

0.2.1 Programmierkurs Python, Dominik Göddeke https://www.ians.uni-stuttgart.de, Universität Stuttgart

Version vom April 2023

Lizenziert unter der Creative Commons Namensnennung 4.0 International Lizenz

Veröffentlicht auf https://zoerr.de, (alle Rechte am Logo vorbehalten)

Gefördert durch die Stiftung Innovation in der Hochschullehre. (alle Rechte am Logo vorbehalten)

Stiftung
Innovation in der
Hochschullehre

Gefördert mit Mitteln der Deutschen Forschungsgemeinschaft (EXC 2075 - 390740016) im Rahmen der Exzellenzstrategie.

[ ]: