Programmierrichtlinien

zur Verbesserung der Qualität der Abgaben und zur erleichterten Korrektur

– Kommentare, Variablenbenennung, etc. –

A) Verbindliche Regeln (sonst ggf. nicht sinnvoll bearbeitet)

A.1) Kommentare:

Eigener Programmcode sollte immer kommentiert sein, insbesondere, wenn er von einer Person gelesen werden soll, die ihn nicht entwickelt hat. Kommentare schaffen Übersicht, machen Code zu einem lesbaren Fließtext und helfen signifikant beim Finden und Beheben von Fehlern (sowohl für die Entwickler als auch für die Korrigierenden).

Die folgenden beiden Arten von Kommentaren sind sinnvoll:

- **Strukturkommentare:** Sie ordnen den Programmcode durch Überschriften zu Beginn der verschiedenen Abschnitte eines Programms (siehe **Programmabschnitte**).
- **Erklärende Kommentare:** Sie erläutern ergänzend zum Code (der ja nur beschreibt, *was* gemacht wird), *warum* und *wie* Sachen im Programm gemacht werden.

Positionen von Kommentaren:

Kommentare sollten über der betreffenden Codezeile stehen. Sollte ein Kommentar mal länger werden, kann man ihn durch Zeilenumbrüche auf mehrere Zeilen verteilen.

In seltenen Fällen kann es auch sinnvoll sein, einen kurzen Kommentar an das Ende der betroffenen Codezeile zu schreiben.

Programmteile, die fast immer zu kommentieren sind:

- **Kontrollstrukturen:** `if`, `while`, `for` (was passiert grob im Rumpf; warum wurde die gegebene Bedingung gewählt, bzw. wann soll verzweigt/abgebrochen werden)
- **Programmabschnitte** (Initialisierung, Benutzereingaben, Fehlerfälle, Hauptprogramm, etc.)
- **Verhaltensbeschreibung** (was macht der Algorithmus im Ganzen, was sind Ein- und Ausgaben)
- **Fehleranalyse** (warum/wo arbeitet der Code noch nicht richtig; was passieren für Fehler, für welche Eingaben treten Fehler auf)
- **Funktionsdefinition**, **Prozedurdefinition**, **Methodendefinition** (erwartete Datentypen für die Parameter, kurze Beschreibung der Parameter, Datentyp und Beschreibung des Rückgabewertes)
- **Klassendefinition** (kurze Beschreibung der Klasse; was repräsentiert die Klasse im Programm)

Man sollte sich aber auf sinnvolle Kommentare beschränken, die nicht nur 1:1 den Code beschreiben, z. B. ist der folgende Kommentar nicht hilfreich:

```
z = z + 1    # z wird um eins erhöht
```

Besser:

```
# Der nächste zu testende Teiler wird berechnet.
teiler = teiler + 1
```

(angelehnt an die Aufgabe zur Primfaktorzerlegung)

A.2) Variablenbenennung:

Um ein Programm lesbar zu halten, sollten möglichst alle Variablen mit beschreibenden Begriffen benannt werden; Einzelbuchstaben werden schnell unübersichtlich und sollten vermieden werden. Sie sind aber für Indizes und Zählvariablen geeignet, bei denen die Verwendung klar ist.

Gegebenenfalls kann die Bedeutung einer Variablen zum besseren Verständnis bei ihrer Initialisierung beschrieben werden:

```
gefunden = False    # Hilfsvariable um die innere Schleife abzubrechen
```

Beispiele für sinnvolle „sprechende“ Variablennamen:

`teiler`, `zaehler`, `summe`, `kandidat`, `result`, `istPrimzahl`

(angelehnt an die Aufgabe zur Primfaktorzerlegung)

A.3) Textformatierung:

Der Programmcode soll immer korrekt eingerückt werden, z. B. `if`, `elif` und `else` auf einer Höhe, und Rümpfe ebenfalls auf einer Höhe (setzt Python ja ohnehin zur syntaktischen Korrektheit voraus):

```
def gibWert(liste, index):
    if index >= 0 and index < len(liste):
        wert = liste[index]
    else:
        wert = 0
    return wert
```

Ebenso bei `while`, `def`, Klassendefinitionen usw. Diese Art der Formatierung hilft bei der Fehlersuche (welcher Rumpf gehört zu welcher Kontrollstruktur) und schafft Übersicht.

A.4) Abgabeformalitäten:

- Möglichst komplette Abgaben: Es müssen mindestens 50% einer Aufgabe bearbeitet werden, damit sie als sinnvoll bearbeitet zählt.

- Kein Abschreiben: Auch wenn eine Aufgabe zusammen mit anderen Gruppen gelöst wurde, sollte jede Gruppe selbstständig ihre eigene Version der Lösung aufschreiben und abgeben. Im Zweifelsfall solltet ihr bei der Abgabe in einem Kommentar mit angeben, dass die Aufgabe mit anderen Gruppen zusammen gelöst wurde. Abgaben, die unkommentiert 1:1 identisch mit den Abgaben anderen Gruppen sind, werden nicht gewertet.

B) Optionales und Hinweise

B.1) Häufige Fehler / Fehlerquellen:

- Überambitionierte Abgaben:

Es kommt regelmäßig zu Abgaben, die weit im Stoff vorgreifen. Das ist in Ordnung, nur leider sind diese Abgaben in den meisten Fällen fehlerhaft. Selbstverständlich dürfen alle Werkzeuge benutzt werden, die man beherrscht – meist sind die Lösungen mit den Mitteln, die bisher in der Vorlesung behandelt worden sind, aber sicherer, lesbarer und weniger fehleranfällig.

Zum anderen muss immer gewährleistet bleiben, dass die Lösung sowohl in ihren Ausgaben als auch in der Form des Codes bei dem bleibt, was die Aufgabe fordert. Häufig hat man sich bei überambitionierten Abgaben weit von der Aufgabenstellung entfernt.

- Aufgabenstellung nur teilweise erfüllt / Aufgabenstellung nicht richtig gelesen:

In den allermeisten Fällen, wenn eine Lösung nicht ganz korrekt ist, liegt der Fehler darin, dass Teile der Aufgabenstellung nicht berücksichtigt wurden. Das sollte leicht zu vermeiden sein und bitte auch verstärkt beachtet werden, da solche Fehler sehr häufig vorkommen.

Ergebnisse, Ausgaben und die Art des Algorithmus sind häufig in der Aufgabenstellung durch konkrete Beispiel beschrieben und können bzw. sollten mit der eigenen Lösung verglichen werden.