Ein Vorschlag zur Dokumentation von Programmen

Dies ist nur ein könnte. Diese Vorlage zu nutzen ist **nicht** verpflichtend. Beachten Sie auch eventuelle Vorgaben Ihrer Tutoren!

Verwendete Aufgabenstellung:

Erstelle ein Programm zur Berechnung des Body-Mass-Index (BMI).

1. Analyse: Analysieren Sie die Problemstellung /die Aufgabe:

- Wo ein- und ausgeben?
- Ein- und Ausgabeformat festlegen,
- weitere Annahmen angeben,
- Zerlegung in Funktionen/Prozeduren und Signatur (formale Schnittstellen) angeben und eventuell Aufteilung in Module.
- Geben Sie an, nach welchem Entwurfsmuster (design pattern) Sie arbeiten.
- Ggf. Angaben zum Algorithmus, falls dieser nicht naheliegend ist:

Beispiel für BMI-Aufgabe

1. Analyse: Das Programm wird gemäß des EVA-Entwurfsmuster entwickelt.Zwei Eingaben werden nacheinander eingelesen:

1. Das Gewicht (genaugenommen die Masse) in kg.
2. Die Größe in cm.

Die Ein-/ Ausgabe erfolgt an der Konsole.
Als Eingaben sind zugelassen Python Literale für Integer **oder** Float!
Der Abschluss der Eingabe erfolgt durch <return>.
Prüfen, ob beide Zahlen positive Größen sind!

Weitere Annahmen: Für beide Eingaben sind nur positive Größen erlaubt!
Zerlegung: Umsetzung als eine Funktion!
Algorithmus: Größe in Meter umrechnen und in die BMI-Formel einsetzen.
Ausgabe: Der errechnete BMI wird auf eine Stelle hinter dem Komma gerundet und auf der Konsole ausgegeben.

2. Coding

Ein Programmkopf wird erwartet, wie im Style Guide (Programmierhandbuch) angegeben.

Implementiert wird in Python 3.10.

3. Test des Programms

Je nach Anforderung der Aufgabe sollen eine bestimmte Anzahl geeigneter Testfälle (*test cases*)angegeben werden und auch durchgeführt werden. Wir unterscheiden:

- **Positivtests** (versucht die Anforderungen (siehe Aufgabentext) zu verifizieren). Der Testfallprüft also **die korrekte Verarbeitung bei korrekter Handhabung.**
- **Negativtest** (prüft die Robustheit einer Software). Beim Negativtest werden absichtlich ungültige Eingabewerte eingegeben, Schnittstellen werden mit falschen Werten beliefert etc. Der Negativ-Testfall prüft also auf **"korrekte" Verarbeitung bei fehlerhafterHandhabung ab**.

„Vollständigkeit“ der Testcases (nach einer bestimmten Methodik) wird anfänglich nicht angestrebt, sondern das Bewusstsein für die Bedeutung und logisches Überlegen dazu. Es ist anzustreben, dass Positivtests und Negativtest genutzt werden.

Oft kann man das erwartete Verhalten der zu entwickelnden Software in sogenannte Äquivalenzklassen (oft Werteintervalle) einteilen, bei denen man davon ausgeht, dass sich die Software gleich verhält. Man kann die Testfälle gut und übersichtlich in einer Tabelle darstellen:

Beispiel für BMI-Aufgabe

#	Äquivalenzklasse	Positivtest Negativtest	Repäsentant	Soll-Ergebnis	Kommentar	Erfüllt
1	Größe in cm > 0, Gewicht in kg >= 0	+	Größe = 180 Gewicht = 75	23,1		✓
2	Größe in cm <= 0	-	0	XError		no
3	Gewicht in kg < 0	-	- 60	XError		✓
4	…					

(In dem BMI-Beispiel kann (und sollte man ggf. gültigere Grenzen: Größe > 100 cm, BMI > 10) wählen. Eventuell also zurück zu 1: sinnvolle Annahmen machen und programmieren! Diese Rückschritte/Schleifen sind normal, tritt häufig auf.

So wäre es super. Gefordert wäre minimal: **Repräsentant, Soll -Ergebnis, ggf. Kommentar** (z.B. warum gewählt). Wichtig ist zunächst nur, dass n sinnvolle Testfälle ausgewählt wurden und diese mit richtigem Ergebnis getestet wurden. Testen wird später in der Vorlesung noch einmal genauer behandelt.

4. Dokumentation

Hier sind wir nicht formal!

Ein Beispiel für den Inhalt einer README-Datei (vgl. https://www.ionos.de/digitalguide/websites/web-entwicklung/readme-datei/): „

1. *Eine generelle Kurzbeschreibung des Systems oder Projektes.*

- *Der Projektstatus ist vor allem dann wichtig, wenn sich das Projekt noch in der Entwicklung befindet. Erwähnen Sie in der Datei geplante Änderungen und die Entwicklungsrichtung oder geben Sie direkt an, wenn ein Projekt fertig entwickelt ist.*
- *Die Anforderungen an die Entwicklungsumgebung für die Integration.*

2. *Eine Anleitung für die Installation und Bedienung.*

- *Eine Auflistung der verwendeten Technologien und gegebenenfalls Links zu weiteren Informationen zu den Technologien.*
- *Open-Source-Projekte, die Entwickler eigenständig verändern oder erweitern, solltenin der Readme.md einen Abschnitt zur „gewünschten Zusammenarbeit“ enthalten. Wie geht man mit Problemen um? Wie sollen Entwickler die Änderungen vorantreiben?*

3. *Bekannte Bugs und eventuelle Fehlerbehebungen.“*

- FAQ-Bereich mit allen bisher gestellten Fragen.
- Copyrights und Lizenzinformationen.

Wir beschränken uns auf die drei wichtigen (nummerierten Angaben)

Beispiel füreine README-Datei für das BMI- Programm:

> Das Programm „bmi.py“ berechnet aus Körpergröße und Gewicht den Body-Mass-Index(BMI).
>
> Installiert sein müssen ein Python-Interpreter 3.10 oder neuer mit den Bibliotheken und Support-Programmen der Standard-Installation von www.python.org .
>
> Man starte das Programm „bmi.py“ in der für das jeweilige Betriebssystem üblichen Art und Weise,aus der Interpreter-Shell **oder** in einer IDE, z.B. IDLE.
>
> Weitere Angaben zur Dokumentation / zum Benutzungshandbuch siehe **1. Analyse**
>
> Es sind keine Bugs bekannt.

Hier gibt es potenziell viele Möglichkeiten. Dieser Teil darf Teil eines Gesamt-PDF oder separat als REDME im .txt-Format beigefügt sein.

Fokussieren Sie sich bitte auf die drei angegebenen Punkte.

Viele Teile können Sie bei anderen Programmieraufgaben wieder benutzen! Erstellen Sie sich doch Ihre persönliche Vorlage.