Prüfung
Einführung in die Programmierung (Python)

Wintersemester 2020/2021

Regeln:

- Bearbeitungszeit:
  - die Bearbeitungszeit beginnt mit Ausgabe der Aufgabe und endet mit Ablauf des 19.03.2021 (also 24:00 Uhr),
  - Belege, die nach Ablauf dieser Frist abgegeben werden, werden nicht mehr berücksichtigt und mit „nicht bestanden bewertet“.
- Abgabe:
  - Senden Sie bis spätestens zum Ablauf der Abgabefrist eine E-Mail mit Ihrem Namen, ihrer Matrikelnummer und der Bezeichnung Ihres Studienganges an: **felix.kettner@hsw.tu-chemnitz.de**.
  - Fügen Sie als Anhang Ihren Programmierbeleg („grading_bot.py“), ggf. die Bearbeitung der Zusatzaufgabe („asimov.py“) sowie die geforderte Dokumentation im PDF- oder Markdownformat an.
  - Hinweis: Alle bis 19.03.2021 um 18 Uhr abgegebenen Arbeiten erhalten bis spätestens zu diesem Zeitpunkt eine Bestätigung der Abgabe per E-Mail.
- Hinweise zur Bewertung:
  - Einfluss auf die Bewertung haben:
    - die Lauffähigkeit und syntaktische Korrektheit des abgegebenen Codes,
    - die aussagekräftige Bezeichnung von Variablen und Funktionen,
    - ggf. notwendige Kommentare in sinnvollem Umfang,
    - Orientierung an den Programmierprinzipien KISS, DRY,
    - PEP8 Konformität und
    - die Dokumentation.
  - Identische Programmierbelege verschiedener PrüfungsteilnehmerInnen werden als Betrugsversuch mit „nicht bestanden“ bewertet.
- Bitte beachten Sie, dass während der Bearbeitungszeit keinerlei Fragen bzgl. des Lernstoffs des Semesters und zum inhaltlichen Teil der Prüfung beantwortet werden können.

Aufgabenstellung:

Szenario
Die Professur für Futuristische Bewertungsmethodik möchte ein Pilotprojekt starten, in welchem Programmierbelege von Studierenden (jeweils mit der Bezeichnung „input.py“) vollautomatisch korrigiert und bewertet werden.
Dazu wurde eine wissenschaftliche Hilfskraft mit Erfahrung in der Programmierung mit Python angestellt. Stellen Sie sich vor, Sie wären diese Hilfskraft.
Sie können davon ausgehen, dass jede „input.py“:

- als Kommentarzeichen lediglich die Raute (#) verwendet und jeder Kommentar in einer eigenen Zeile steht,
- PEP8 konform ist (z.B. Leerzeichen nach Kommentarzeichen),
- syntaktisch korrekt und lauffähig ist,
- in Zeile 1 den Vor- und Nachnamen des Studenten als Kommentar enhält (z.B.: `# Arthur Dent`),
- in Zeile 2 die 6-stellige Matrikelnummer des Studenten (z.B.: `# 235813`) als Kommentar enthält.

Aufgabe
Lesen Sie sich vor Beginn der Bearbeitung auch alle Hinweise gründlich durch!
Schreiben Sie ein Python Programm, welches folgende Kriterien und Funktionalitäten erfüllt:

- Der Name des Programms soll „grading_bot.py“ lauten.
- Das Programm soll eine externe Datei namens „input.py“ schreibgeschützt öffnen und deren Inhalt weiterverarbeiten, ohne das Original zu verändern.
- Zunächst sollen Vorname, Nachname und Matrikelnummer ausgelesen und in einem Tupel (mit drei Einträgen) gespeichert werden.
- Schreiben Sie nun eine Funktion, welche den Code aus „input.py“ bereinigt und ohne Kommentarzeilen unter „no_comments.py“ speichert. Die Funktion soll anschließend die Anzahl der Codezeilen und Kommentarzeilen als zwei Integer-Werte zurückgeben.
- Schreiben Sie eine Funktion, welche die Anzahl der verwendeten if-Anweisungen ermittelt und zurückgibt.
- Schreiben Sie eine Funktion, welche die Summe der verwendeten for-Anweisungen und while-Anweisungen ermittelt und als einen Wert zurückgibt.
- Schreiben Sie eine Funktion, welche überprüft, ob alle normalen geöffneten Klammern „(“ auch wieder geschlossen wurden „)“ und entsprechend True oder False zurückgibt.
- Schreiben Sie eine Funktion, welche überprüft, ob es sich bei der Matrikelnummer um eine Zahl aus der Fibonacci-Folge handelt und entsprechend True oder False zurückgibt.
- Schreiben Sie eine Funktion, welche die Benotung des Programmierbelegs nach folgender Formel ermittelt:

$$Note = \frac{Quersumme\ von\left(\frac{Anzahl\ Buchstaben\ Vorname + Anzahl\ Codezeilen}{\left(Anzahl\ Kommentarzeilen\right)^{\left(wenn\ Fibonacci\ True:2,\ sonst:1\right)}}\right)}{2+\left(wenn\ alle\ Klammern\ geschlossen\ wurden:1,\ sonst:0\right)}$$

Hinweise zur Formel:

- Realisieren Sie die Division im Zähler als Ganzzahldivision!

    - Bilden Sie auch von der Quersumme jeweils wieder die Quersumme solange das Ergebnis nicht einstellig ist
- Wenn die Anzahl der if-Anweisungen mindestens so hoch ist wie die Summe der for- und while-Anweisungen soll die Note auf die nächste Ganzzahl aufgerundet werden (Bsp. 2,4 wird zu 3).
  Andernfalls soll die Note auf die nächste Ganzzahl abgerundet werden (Bsp. 3,7 wird zu 3).
- Schreiben Sie eine Funktion, welche eine Datei mit Namen „notenliste.txt“ öffnet und Vorname, Name, Matrikelnummer und Note in einer neuen Zeile einträgt, ohne bereits vorhandenen Inhalt zu überschreiben.
- Schreiben Sie zu Ihrem Programm eine Kurzdokumentation im Umfang von mindestens einer A4 Seite. Die Dokumentation soll eine kurze Beschreibung der einzelnen Funktionen und der Programmstruktur enthalten sowie eine Erläuterung des gedachten Programmablaufs (welcher Programmteil soll was tun, warum so und nicht anders implementiert).
  Erwähnen Sie außerdem die von Ihnen verwendete Python Version (3.?).

Zusatzaufgabe (optional)

- Informieren Sie sich zu den drei Gesetzen der Robotik von Isaac Asimov.
  Gehen Sie von einer wohlwollenden Interpretation des ersten Gesetzes aus und schreiben Sie eine Funktion, welche eine Bewertung von 5,0 stattdessen mit dem Eintrag „ohne Wertung“ als Note in der Notenliste vornimmt und einen entsprechenden Hinweis auf dem Bildschirm (normale Konsolenausgabe ist ausreichend) ausgibt.
  Schreiben Sie diese Funktion in eine eigenständige Pythondatei namens „asimov.py“ und importieren Sie diese zur Verwendung in „grading_bot.py“.

Allgemeine Hinweise

- Dieses Szenario ist rein fiktiv und entspricht NICHT dem tatsächlichen Bewertungsablauf.
- Versuchen Sie Funktionalitäten des Programms sinnvoll in einzelne Funktionen zu unterteilen.
- Es wird keine „input.py“ zur Verfügung gestellt. Erstellen Sie sich für Testzwecke ggf. selbst eine entsprechende Datei.
- Die Zusatzaufgabe ist optional. Für ein bestehen mit der Note „sehr gut“ ist deren Lösung nicht notwendig, ggf. kann eine korrekte Lösung allerdings die Note verbessern.
- Eventuell ist die Methode .strip() für die Verarbeitung von Strings hilfreich.
- Zulässig ist ausschließlich die Verwendung von Bibliotheken, welche zum Standardumfang von Python gehören (z.B. math, os, time…) und nicht erst zusätzlich installiert werden müssen.