Testing und Debugging: Übung 09

Abgabetermin 04.06.2019 13:59

Name

Matrikelnummer

Gruppennummer

1 Fehlerreport und *TRAFFIC* Prinzip (100 Punkte)

Gegeben sei das folgende Programm

middle.py

```
#!/usr/bin/env python3
# :product: sorting suite
# :module:  middle
# :version: v0.1.0
#
# :desc: Die Funktion erhaelt beliebige ganze Zahlen als Eingabe.
#        Zurueckgegeben wird die zweitgroeßte Zahl.


def middle(x, y, z):
    assert isinstance(x, int)
    assert isinstance(y, int)
    assert isinstance(z, int)

    m = z
    if y < z:
        if x < y:
            m = y
        elif x < z:
            m = y
    else:
        if x > y:
            m = y
        elif x > z:
            m = x

    return m

if __name__ == "__main__":
    import sys

    if len(sys.argv) == 4:
        print(middle(int(sys.argv[1]), int(sys.argv[2]), int(sys.argv[3])))
    else:
        print("Programm mit drei ganzen Zahlen als Argument aufrufen.")
## vim: et ai ts=4
```

Beim Testen wird Ihnen auffallen, dass das Programm (aufgerufen mit gültigen Parametern) nicht in jedem Fall die zweitgrößte Zahl zurück gibt.

1.1 Schreiben Sie einen Fehlerreport dafür (Traffic)(30 Punkte)

Gehen Sie beim Schreiben des Reports so vor, wie sie es in der Vorlesung gelernt haben.

1.2 Stellen Sie den Fehler nach und erstellen Sie einen Testcase (tRAffic) (20 Punkte)

Welcher (gültige) Aufruf der `middle` Funktion (mit drei ganzen Zahlen!) ergibt ein fehlerhaftes Ergebnis.

Schreiben Sie eine kurze Testfunktion dafür (die Benutzung des unittest-Moduls ist nicht nötig).

1.3 Finden Sie die Fehlerursache (traFFIc) (30 Punkte)

Benutzen Sie hierfür die wissenschaftliche Methode und das Verfolgen der Änderungen der Zustände des Programms (vgl 1.1 und 1.2 aus Übung08)

Tipp
Benutzen Sie die Lösung der Programmieraufgabe für die Variablenverfolgung.

1.4 Korrigieren Sie den Fehler (traffiC) (20 Punkte)

Geben Sie die Zeilennummer und den korrigierten Programmaufruf an.

```
.
.
.
.
.
```

2 Programmieraufgabe: Automatische Variablenverfolgung (100 Punkte)

In der Übung 08 haben Sie gesehen, das das händische Verfolgen von Zustandsänderungen während des Programmablaufs sehr mühsam sein kann.

Glücklicherweise ist einer der Wahlsprüche von Python „Batteries included“. In unserem Falle heißt, dass es bereits einen in Python direkt eingebauten Mechanismus gibt, der uns hilft die Ausführung eines Programm(-teils) zu überwachen.

Unter http://docs.python.org/3.5/library/sys.html bei der Funktion `sys.settrace(tracefunc)` ist beschrieben, wie eine Tracefunktion aussehen muss und wie die Parameter bei den einzelnen Events belegt sind.

Lesen Sie unbedingt nach, wie die gesetzten Werte des *frame* Objekts beim Event *line* zu interpretieren sind!

In der Datei *uebung09.py* ist die Tracefunktion `traceit` und die Funktion `main` die den Trace benutzt bereits implementiert.

Dateien für die Programmieraufgabe

Grading/*init*py	Hinweisdatei für Python: Das Verzeichnis ist ein Modul
Grading/Grading.py	Quellcode der Testklasse, die in *run_test.py* verwendet wird
***uebung09.py** **	In dieser Datei müssen Sie die Funktion `print_trace()` gemäß Aufgabenstellung implementieren

Die mit [*] markierten Dateien müssen **bearbeitet** und **abgegeben** werden.

2.1 Ihre Aufgabe

Vervollständigen Sie die Implementation der Funktion `print_trace` so, das durch den Aufruf der Funktion `main` eine Ausgabe der Zustandsänderungen im CSV Format entsteht. Eine genauere Beschreibung und ein Beispiel für die Ausgabe finden Sie in der Quelldatei direkt über dem Aufruf von `main` ganz am Ende der Datei.

2.2 Bewertung

Wie gewohnt soll die Ausgabe mit ihren Namen etc. in gewohnter Form beginnen:

```
====================
Aufgabe:       09
Gruppennummer: 99

Matrikelnummer, Name
--------------------
        99999, Sebastian Stigler
====================
```

Es dürfen keine Assertions geworfen werden.

Bewertet werden die Ausgaben von 10 Verschiedenen Aufrufen von `main`. Dabei wird darauf geachtet, das die Ausgabe von `main` als CSV Datei z.B. in *LibreOffice Calc* (oder *Excel*) geladen werden kann. Natürlich müssen auch die Werte der überwachten Variablen stimmen.

2.3 Abgabe der Programmieraufgabe

1. Falls noch nicht vorhanden, dann erstellen Sie sich auch https://github.com einen Account. (Pro Gruppe ist nur ein Account und eine Abgabe nötig!)
2. Schicken Sie bei Ihrer ersten Abgabe den **Namen** des github Accounts (ohne Passwort) an sebastian.stigler@htw-aalen.de mit dem Betreff: *Testing und Debugging Github Account*
3. Gehen Sie in Ihrer virtuellen Maschine in das Aufgabenverzeichnis (wo sich Ihre bearbeitete Aufgabe und die Datei `submit.cfg` befinden).
4. Tippen Sie in der Konsole den Befehl *submit*. Dieser wird beim ersten Ausführen nach den Zugangsdaten Ihres github Accounts fragen. Anschließend werden die bearbeiteten Aufgaben verschlüsselt auf https://gist.github.com abgelegt.
5. Diese Datei wird nach dem Abgabetermin automatisch zur Korrektur heruntergeladen.

Viel Erfolg