Week 6 (Document MCT)

Doelstellingen

Je bent in staat om te werken met tekstbestanden: inlezen, wegschrijven, aanvullen.
Je bent in staat om te werken met het aanwezige filesysteem.
Je bent in staat om een correct datatype (list, dictionary) te kiezen.

Afspraken

Eindniveau - oefeningen

Ben je een student MCT, dan beheers je de oefeningen tot moeilijkheidsgraad “D”

Ben je een student MIT, dan beheers je de oefeningen tot moeilijkheidsgraad “C”

GitHub

Alle oplossingen van week 6 dienen op Github geplaatst te worden. Volg hiervoor de procedure uitgelegd op Leho. Via Github krijg je ook alle bronmateriaal voor deze opgave.

Om je repository in onze Github-classroom aan te maken, klik je op volgende link:
https://classroom.github.com/a/wv7hLxC8

Na elke oefening kan je een 'commit & push' doen zodat jouw versie op GitHub steeds aangepast wordt. Geef telkens een gepaste message mee.

Niet afgewerkte oefeningen werk je thuis verder af: voer regelmatig een 'push & commit'-opdracht uit zodat alle oplossingen op je github-repository beschikbaar zijn.

Bij een programmeertaal zoals Python onder de knie krijgen is veelvuldig oefenen essentieel en een noodzakelijke voorwaarde. Daarom vind je in elk labo-document nog twee extra onderdelen. Deze worden als volgt aangeduid.

Uitbreidingsoefeningen - eigen onderzoek

Dit onderdeel gaat verder dan de geziene leerstof van deze week. Vaak zijn de opdrachten net iets moeilijker dan hetgeen je in het labo deed. Je zal de Python documentation en Google nodig hebben voor dit onderzoek.

We motiveren iedereen om dit (thuis) iedere week voor te bereiden. Je onderzoekt in dit onderdeel een onderwerp die de volgende weken terugkomt in de theorie of het labo.

Oefeningen voor thuis

In dit onderdeel vind je analoge oefeningen zoals je reeds in het labo maakte.
Deze oefeningen hebben dezelfde moeilijkheidsgraden zoals in het labo. Het is pas door de oefeningen thuis “alleen” te maken dat je de leerstof zich eigen maakt. Loop je vast bij een oefening? Herbekijk de theorie, kijk of je een analoge oefening terugvindt die je maakte tijdens het labo. Lukt het nog steeds niet? Kom met je voorbereiding naar het monitoriaat!

Feedback labo week 6

Lees nog even de algemene opmerkingen uit de theorieles na. Je vindt ze op Leho terug.

Reminder uit vorig labo: via de sneltoets ' **Shift+Alt+F** ' kan je jouw code onmiddellijk volgens de PEP8-stijl formatteren.

Voor wie fan is van tips & trics bij het gebruik van Visual Studio Code:
https://code.visualstudio.com/docs/getstarted/tips-and-tricks

Oefeningen

Vermeld in commentaar telkens de opgave!

Dit labo-document is specifiek voor studenten MCT. Ben je MIT student? Dan werk je deze week aan andere oefeningen.

Oef 01

Voeg een tekstbestand toe aan uw projectmap. Voorzie het tekstbestand van verschillende regels tekst. Maak nu een functie '**lees_bestand_in**' die het tekstbestand inleest en lijn per lijn afprint. De parameter van de functie is de bestandsnaam. Elke lijn wordt door een lijnnummer vooraf gegaan.

Voorbeeld:

```
Inhoud bestand:
Regel 1: Dit is een test
Regel 2: om meerdere lijnen
Regel 3: na elkaar
Regel 4: in te lezen.
```

Oef 02

Maak een applicatie die controleert of een bestand 'temp.txt' wel of niet bestaat.

Wanneer het bestand bestaat, dan wordt de functie '**lees_bestand_in'** uit de vorige oefening aangeroepen.

Wanneer het bestand nog niet bestaat, wordt de functie '**schrijf_input_naar_bestand**' aangeroepen. Deze functie maakt een nieuw bestand aan (met de bestandsnaam als parameter). Vervolgens vragen we meermaals aan de gebruiker om een lijn op te geven. Elke nieuwe lijn wordt onmiddellijk naar het bestand weggeschreven.

```
Geef een nieuwe regel op, of druk op enter om te eindigen:> dit is een test
Geef een nieuwe regel op, of druk op enter om te eindigen:> om meerdere lijnen
Geef een nieuwe regel op, of druk op enter om te eindigen:> naar een bestand op te slaan
Geef een nieuwe regel op, of druk op enter om te eindigen:>
Press any key to continue . . .
```

Oef 03

Maak de functie '**vul_bestand_aan**'. Deze functie is analoog aan de vorige functie 'schrijf_input_naar_bestand' behalve dat het bestand onderaan verder aangevuld wordt (en de bestaande inhoud dus niet gewist wordt). Test uit.

```
Geef een nieuwe regel op, of druk op enter om te eindigen:> Nog een extra lijn
Geef een nieuwe regel op, of druk op enter om te eindigen:> Om het af te leren
Geef een nieuwe regel op, of druk op enter om te eindigen:> the end
Press any key to continue . . .
```

Oef 04

Gegeven het bronbestand *spelers.txt*. Dit bronbestand staat in de submap 'doc' in. Analyseer het bronbestand: elke regel bestaat uit de gegevens van een voetbalspeler. Maak een applicatie die het bronbestand inleest, en vervolgens 11 verschillende spelers willekeurig selecteert om als ploegopstelling te kunnen fungeren. Deze 11 spelers worden in een nieuw bestand weggeschreven.

Bepaal vooraf welke datastructuur je zal gebruiken om alle data bij te houden.

Werk vervolgens in deelproblemen door oa. gebruik te maken van onderstaande functies:

- Functie '**inlezen_spelers**' die het bronbestand correct inleest en de data in een datastructuur teruggeeft
- Functie '**print_spelers**' die in staat is om alle spelers uit de gekozen datastructuur af te printen
- Functie '**selecteer_random_elftal**' die in staat is om 11 verschillende spelers willekeurig te selecteren en terug te geven.
- Functie '**opslaan_ploegopstelling**' die in staat is om een opstelling naar een tekstbestand weg te schrijven.

Test voldoende uit.

```
C:\Users\Stijn.Walcarius\AppData\Local\Programs\Python\Python3
Naam list: Geselecteerden
1: Gershon Rami
2: Messoudi Mohammed
3: Nfor Ernest
4: Oussalah Mustapha
5: Monrose Steven-Joseph
6: Zukanovic Erwin
7: N'Diaye Ismaila
8: Mulemo Landry
9: Pavlovic Nebojsa
10: Baptiste Martin
11: Ghesquiere Geoffrey
Geselecteerde spelers zijn opgeslagen naar geselecteerd.txt
```

Oef 05

Gegeven het bronbestand *scores.txt*.

Dit bronbestand staat in de submap 'doc' in. Analyseer het bronbestand: elke lijn bestaat uit de gegevens van een student gevolgd door 5 scores.

Maak nu een applicatie die na het inlezen van het bronbestand op basis van de naam de scores van een student(en) kan opzoeken en afprinten, en onmiddellijk ook de gemiddelde score afprint.

Bepaal vooraf welke datastructuur je zal gebruiken om alle data bij te houden.

Werk vervolgens in deelproblemen:

- Een functie '**inlezen_scores_studenten**' staat in voor het inlezen van het bronbestand
  - Welke parameter(s) heeft deze functie nodig?
  - Wat zal deze functie terug geven?
- Een functie '**zoek_en_print_student**' zoekt de scores van een student op en print deze af.
  - Welke parameter(s) heeft deze functie nodig?

De scores zijn correct ingelezen.

```
Geef de naam van een student op: Blancke
Resultaat zoekopdracht:
Gevonden student: Blancke Karen
De scores van deze student zijn: [16, 15, 17, 15, 18]
De gemiddelde score is 16.20/20
```

Oef 06

Gegeven het bronbestand *morse_vertaling.txt*.

Dit bronbestand staat in de submap 'doc' in. Analyseer het bronbestand: het bestaat uit een opsomming van letters met bijhorende morsevertaling. **Merk op dat de eerste lijn van het bestand hier niet bijhoort. *Wat zal je dus doen met de eerste lijn?***

Stap 1:
Maak een applicatie die bestaat uit:

- Dictionary 'morse_dictionary' (globale variabele)
- Functie '**inlezen_morse_bestand'**:
  Die het morse-bestand correct inleest en de data in de dictionary bijhoudt (waarom?).
  - Hoe kan je de eerste lijn 'overslaan'?
- Print nadien de dictionary morse_dictionary af.
- Functie '**vertaal_letter**' met als parameter een letter (string). Deze functie controleert of de letter in de dictionary aanwezig is.
  - Indien zo wordt de vertaling teruggegeven, indien niet zo wordt een '?' terug gegeven.

Voorbeeld:

```
MorseDictionary: {',': '--..--', '.'
 '..---', '3': '...--', '4': '....-'
'-.-.', 'd': '-..', 'e': '.', 'f': '
, 'o': '---', 'p': '.--.', 'q': '--.
'z': '--..', ' ': ' '}
Geef een karakter (letter) in > a
De vertaling van a is .-
```

Stap 2:
Functie '**vertaal_tekst_in_morse**' met als parameter een string (te_vertalen_tekst) die de doorgegeven tekst vertaalt en nadien de vertaling teruggeeft. Deze functie maakt gebruik van de net geschreven functie '**vertaal_letter**'.

- Hoe kan je letter per letter vertalen? Welke structuur heb je hiervoor nodig?

Voorbeeld:

```
MorseDictionary: {'0': '-----', '9': '----.', '.': '.-.-.-', 'y': '-.--', 'c': '-.-.', 'l':
Geef uw tekst in: sos
Morsevertaling: ... --- ...
```

Uitbreiding – Eigen onderzoek

-

Oefeningen voor thuis

Bereid je Q voor!

Bereid je Q voor!