Project stemsysteem

Pyhton 23-24

AP Hogeschool

Paul Sandu 1ITCSC2

Contents

[Ontwikkelproces 3](#_Toc168591876)

[Entiteiten 3](#_Toc168591877)

[Bestand 1 – Kiezers, Kandidaten en Lijsten 4](#_Toc168591878)

[Bestand 2 – Stemproces 8](#_Toc168591879)

[Functies 13](#_Toc168591880)

[Main 16](#_Toc168591881)

[Veiligheid 18](#_Toc168591882)

[Bronnen 19](#_Toc168591883)

# Ontwikkelproces

## Entiteiten

Deze hoofdstuk zal gaan over de gemaakte entiteiten, deze zijn de entiteiten van het eerste versie van het projectopdracht. Ik zal uitleggen welke attributen ze hebben, wat de methoden allemaal doen en of er nutteloze/ongebruikte delen code zijn.

### Bestand 1 – Kiezers, Kandidaten en Lijsten

#### Kiezer klasse

Ten eerste zal ik beginnen met de Kiezer klasse. Deze klasse zal gebruikt worden om de kiezers te genereren, en is de basisklasse dat ook gebruikt wordt voor de Kandidaat klasse.

A screenshot of a computer program

Description automatically generated

Mijn klassen en hun bijhorende methoden hebben allemaal een docstring, wat de werking ervan al uitlegt op een simpele manier.

Een attribuut dat ik uiteindelijk niet gebruikte was de rijksregisternummer (ID), het reden dat ik het aanmaakte was om ervoor te zorgen dat er minstens iets tussen kiezers was dat hun kon onderscheid voor het geval dat ze hetzelfde naam hebben. Uiteindelijk was dit compleet nutteloos, ik gebruikte in mijn code de leeftijd attribuut ook niet.

De kiezers krijgen een ID bij generatie, de voor- en achternamen worden uit een lijst van 40 elk gekozen wat voor de mogelijkheid van 1600 unieke namen zorgt. Ze krijgen ook 2 attributen voor te stemmen, namelijk de gestemd en mogelijkeStemmen attributen, beide worden nagekeken en geüpdate wanneer een kiezer stemt voor een extra laagje veiligheid.

Ik zal nu over de methodes beginnen dat deze klasse heeft. Het eerste methode is getName, het accepteert geen externe parameters en retourneert een string dat de voor- en achternaam van een kiezer bevat. Het nood ervoor komt omdat beide attributen privé zijn.

Het tweede methode is gestemd, het accepteert geen externe parameters en wordt gebruikt om de 2 stemattributen te updaten. Als mogelijkeStemmen niet gelijk wordt aan 0 na het stemmen zal het code stoppen omdat er iets mis ging.

Het derde methode is heeftGestemd, het accepteert geen externe parameters en wordt enkel gebruikt om de status van de stem van de kiezer te returnen.

Het vierde (en laatste) methode van deze klasse is de \_\_repr\_\_ methode, deze methode wordt gebruikt om de instanties van deze klassen te representeren. In dit geval doet het ook bijna vrij letterlijk hetzelfde als de getName functie, ik gebruik het zodat er altijd een string representatie is, het maakte het debuggen ook gemakkelijk wanneer nodig.

#### Kandidaat klasse

Deze klasse zal gebruikt worden om de kandidaten te genereren, het maakt gebruik van overerving om ook alle methoden en attributen van de Kiezer klasse te hebben.

A screenshot of a computer error

Description automatically generated

Het reden dat de kandidaat klasse bestaat is omdat ik aan het project begon voor de 2de versie van het opdrachtenblad, ik had dit deel dus al voor een meerderheid af voor de update ervan. Ik heb daarna verkozen de klasse toch te behouden.

Het grote verschil tussen kiezers en kandidaten is dat de kandidaten ook een stemmen attribuut hebben, waar het hoeveelheid verkregen stemmen in opgeslagen wordt.

Deze klasse heeft ook 2 methoden. De stemOntvangen methode zorgt ervoor dat het hoeveelheid stemmen dat een kandidaat heeft verhoogd worden met 1, en de getStemmen methode returnt gewoon het hoeveelheid ontvangen stemmen.

#### Lijst klasse

Deze klasse is om de lijsten te maken, een lijst is een partij, iets dat in het begin niet wist.

A computer screen shot of text

Description automatically generated

Deze klasse heeft 3 attributen, maxKandidaten geeft aan hoeveel kandidaten elk partij kan hebben. Aangezien elk lijst een partij is vond ik het leuk hen een naam te geven. In het begin wou ik dat de partijen NBA teams waren, maar later verkoos ik de namen te laten genereren met een for-loop wat ze veel saaier maakte. Het laatste attribuut is een list om de kandidaten in bij te houden.

Deze klasse heeft 3 attributen, het eerste (addKandidaat) zal kandidaten toevoegen aan de kandidatenlijst. Het tweede (aantalKandidaten) zal aangeven hoeveel kandidaten al in de kandidatenlijst zijn. Het derde, en laatste, methode zal de kandidatenlijst returnen, daarom heet het ook getKandidaten. Buiten de eerste methode accepteren de methoden geen externe parameters.

### Bestand 2 – Stemproces

#### Stembus klasse

A screenshot of a computer program

Description automatically generated

Deze klasse wordt gebruikt om de stembiljetten te maken, eerst hield ik er meer in bij, net als de lijstnummer en het index van het gekozen kandidaat. Uiteindelijk heb ik verkozen dat allemaal weg te halen aangezien ik ze niet gebruikte. De attributen dat deze klasse heeft zijn de kiezer, 1 om na te kijken of de stembiljet geregistreerd is en een controleattribuut.

Deze klasse heeft 3 methoden, de registratiecheck methode zal de registratiestatus returnen. De controleer methode en de registreer klasse werken samen voor een meer veilige registratie van elk stembiljet. De controleer methode kijkt na of de scanner effectief iets wilt scannen en of de scanner een scanner is (om het gewoon doorgeven van een True of False te voorkomen). Als alles goed is zal het de registreer methode aanroepen wat dan de registratie voltooit.

#### Stembus klasse

A computer screen with text on it

Description automatically generated

De stembus heeft 2 grote taken, namelijk het bijhouden van de geldige stembiljeten en het geven van opstartcodes aan de USBstick dat dan wordt gebruikt om de stemcomputers te initialiseren. Deze 2 functies heb ik dan gemaakt door middel van methoden. Het enige attribuut dat de Stembus klasse heeft is een list om de verschillende geldige stembiljetten bij te houden.

#### Scanner klasse

A screenshot of a computer

Description automatically generated

De scanner heeft 1 taak, namelijk het scannen van geldige stembiljeten om ze te kunnen registreren. Deze functie wordt behandeld door de check methode. De getScan methode kijkt na of de scanner wel aan staat (scan=True), als deze wordt gebruikt om de scan attribuut ook te resetten naar False, wat ervoor zorgt dat de controleer methode van de stembiljet klasse niet meer zal werken buiten als de scanner weer aangezet wordt.

#### Stemcomputer klasse

A screenshot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generated  
A black screen with text

Description automatically generated

Deze klasse wordt gebruikt om te stemmen, te confirmeren dat een stem goed ging, stembiljetten te maken en de opstartcodes van beide de chipkaarten en de USB na te kijken. Het laatste wordt gedaan in de init methode. Er zijn ook een paar variabelen op de level van de klasse zelf, deze worden gebruikt om de 3 stemcomputers samen te laten werken. Net als besproken in de online lessen is het moeilijk om alle 3 stemcomputers te gebruiken, waardoor ik 1 van de 3 kies die het stemgedeelte zal doen, terwijl dat de andere 2 er zijn voor de juistheid van de opdracht te behouden.

Voor het stemmen koos ik ervoor elke kiezer 1 tot 3 mogelijke stemmen te geven, ze kunnen dus op meer als 1 kandidaat of partij stemmen. Partijstemmen zullen elk member van dat partij een stem geven, wat niet het beste optie is maar wel een werkende optie. Een probleem is dat een kiezer theoretisch gezien meerdere keren op hetzelfde kandidaat kan stemmen.

Voor de rest vind ik dat het code vrij simpel is en de bijhorende docstrings meer als genoeg uitleg bevatten voor de methoden.

#### Chipkaart klasse en USBstick klasse

A computer code on a black background

Description automatically generated  
A screenshot of a computer

Description automatically generated

Beide klassen werken met opstartcodes, hoewel hun gebruik verschillend is, is een meerderheid van de werking identiek. Het grootste verschil tussen de 2 is dat de usb zijn opstartcode ophaalt van de stembus, terwijl dat de chipkaart het zelf doet, ze gebruiken ook beide een verschillende file om een opstartcode te halen. De gehashte versie van de opgehaalde opstartcodes functioneren dan als opstartcodes. Beide hebben een getter methode om de opstartcode op te vragen.

## Functies

#### Parser

A computer screen shot of a black screen with white text

Description automatically generated

Deze methode neemt de dictionary “lijsten” aan dat de lijsten bevat en zorgt ervoor dat het opgedeeld wordt naar 3 verschillende lists. De eerste bevat de kandidaten, de tweede bevat het aantal stemmen dat alk kandidaat heeft behaald en de derde bevat het totale hoeveelheid stemmen per partij (gewoon de som van de stemmen van de kandidaten).

Na het opdelen van de dictionary wordt de volgende functie dat ik zal bespreken uitgevoerd.

#### ToHTML

A computer screen shot of a program code

Description automatically generated

Deze functie maakt gebruik van Airium om de HTML te genereren, het resultaat hiervan is te vinden in de index.html bestand in de forWeb folder. Ik zal ook een screenshot hieronder zeten van een deeltje ervan.

Net als u kan zien heb ik 2 dingen niet, ten eerste is er geen manier om een winnende partij te kiezen als 2 hetzelfde hoge hoeveelheid stemmen krijgen. Dat zorgt ervoor dat ze gewoon vrienden moeten zijn en de titel van winnaar moeten delen, alle andere partijen krijgen de “niet\_winnaar” klasse omdat ik losers een beetje te erg vond. Het tweede deel dat ik niet heb is de zetelverdeling, mijn heel simpele reden hiervoor is dat het niet in het projectdocument stond, waardoor ik er niet op focuste.

Ik heb ook een heel simpele style.css erbij gemaakt, ik zal dus ook een screenshot hieronder zetten met de output van de 2 bestanden.

HTML-code:  
A screen shot of a computer program

Description automatically generated

Output:A close-up of a white screen

Description automatically generated

Bron Airium: <https://pypi.org/project/airium/>

## Main

A computer screen with many colorful text

Description automatically generated with medium confidence  
A computer code on a black background

Description automatically generated  
A screen shot of a computer program

Description automatically generated

Normaal zou ik door heel mijn main functie gaan om alles uit te leggen, maar alles werd al uitgelegd dus ik vind dat het code dat gevonden kan worden op de screenshots relatief duidelijk moet zijn bij dit punt. De main functie maakt gebruik van alles dat tot nu toe besproken werd, en zorgt ervoor dat alle klassen geïmplementeerd worden op een fantastische manier dat de werking garandeert.

De main functie zorgt er ook voor dat er een hoeveelheid output is naar de terminal, spijtig genoeg is het hoeveelheid output te groot, dus ik verkoos ook alles te sturen naar een bestand, namelijk “output.txt”. Het inhoud van output.txt ziet er als volgt uit:   
A screen shot of a computer

Description automatically generated

# Veiligheid

Als een cybersecurity student zal ik spijtig genoeg ook de veiligheid moeten bespreken.

De klassen heb ik op een correcte object georiënteerde manier proberen te maken door de attributen dat belangrijk zijn als private te zetten door middel van een dubbele underscore (\_\_). Dit zou eigenlijk veiliger zijn in een ander programmeertaal, namelijk C++, dus op overheidsniveau zou het veiliger zijn het niet in pyhton te maken.

Door naar het Belgisch verkiezingssysteem te kijken zie ik nog wel de mogelijkheid op menselijke fouten, net als het verliezen van het stembiljet door de kiezer, in mijn code heb ik dat proberen te ontwijken door geen keuze te geven aan de kiezers over wat men met het stembiljet doet.

Iets dat ik interessant vond was dat een stembiljet nog steeds afgeprint wordt, zelfs als de stemming digitaal gebeurt (met een stemcomputer), wat ervoor zorgt dat de stemmen geverifieerd kunnen worden.

Een gevaar voor het Belgisch stemsysteem is nog steeds propaganda/desinformatie dat verspreid wordt door mensen met slechte intenties/mensen dat iets hebben te winnen als een bepaalde partij wint (*Desinformatie En Beïnvloeding Tijdens De Verkiezingen - Crisiscentrum*, n.d.).

Ik kon online niets vinden dat te maken had met het gebruik van blockchain in de verkiezingen, wat wel voor veel extra veiligheid zouden zorgen. Het grootste veiligheidspunt van een blockchainimplementatie is namelijk hoe alles dat erop gebeurt publiek is. Het is dus ook gemakkelijk om te verstaan waarom men dat niet zou doen. Aan het einde van de dag is de bedoeling dat stemmen anoniem blijven zodat iedereen een eerlijk stem kan indienen.

In het algemeen, in een ideale wereld waar wij ervanuit gaan dat er geen menselijke fouten zijn, is het Belgisch stemproces zeer veilig. Door het elimineren van het enig duidelijk probleempunt is het een voorbeeldig veilig verkiezingssysteem. Dat komt onder anderen door het super strikte hoeveelheid maatregelen dat men hier voor, tijdens en na het stemproces heeft.

# Bronnen

*- Desinformatie en beïnvloeding tijdens de verkiezingen - Crisiscentrum*. (n.d.). <https://crisiscentrum.be/nl/risicos-belgie/veiligheidsrisicos/desinformatie/desinformatie-en-beinvloeding-tijdens-de>