INFRASTRUCTUUR

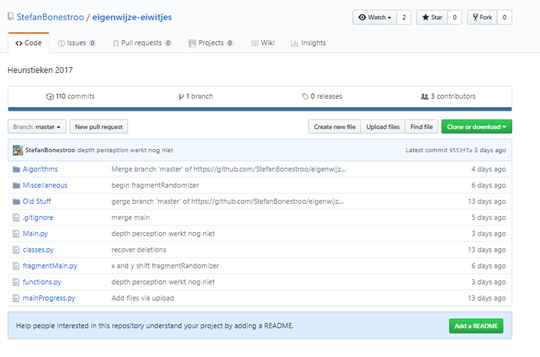
**Github:** [**https://github.com/StefanBonestroo/eigenwijze-eiwitjes**](https://github.com/StefanBonestroo/eigenwijze-eiwitjes)

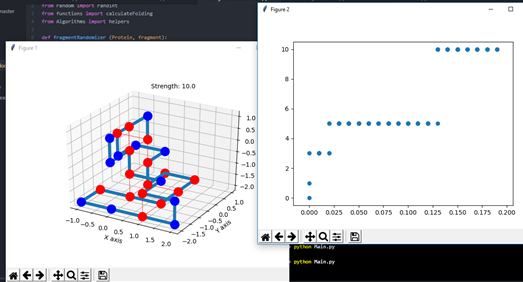
**Git is opgezet/schikking van files:**

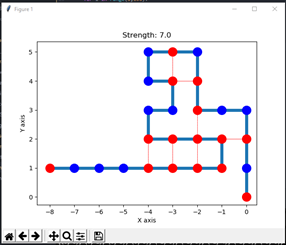
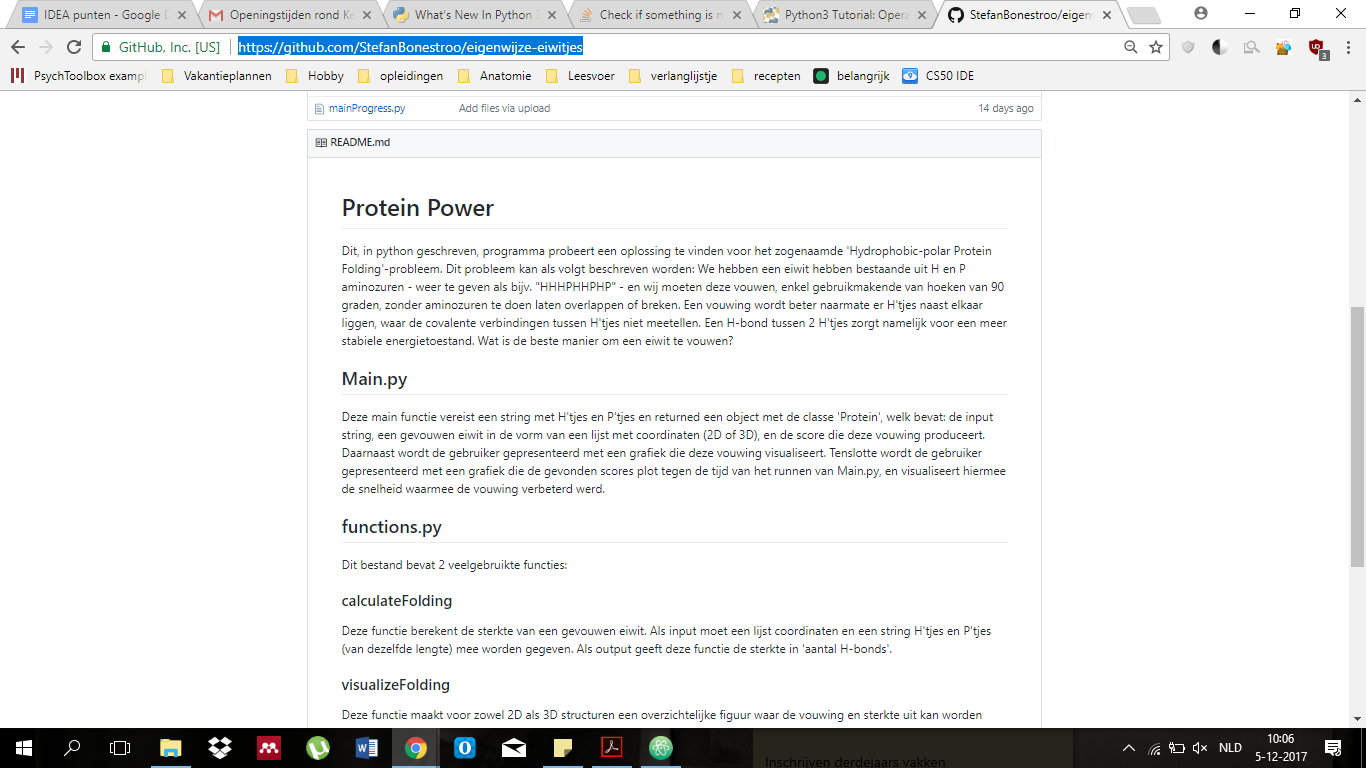
**Visualisatie is bijna in orde:**

- **Opzetten en gebruiken van Git**

- **Mappenstructuur: er is een speciale map voor alghoritms**

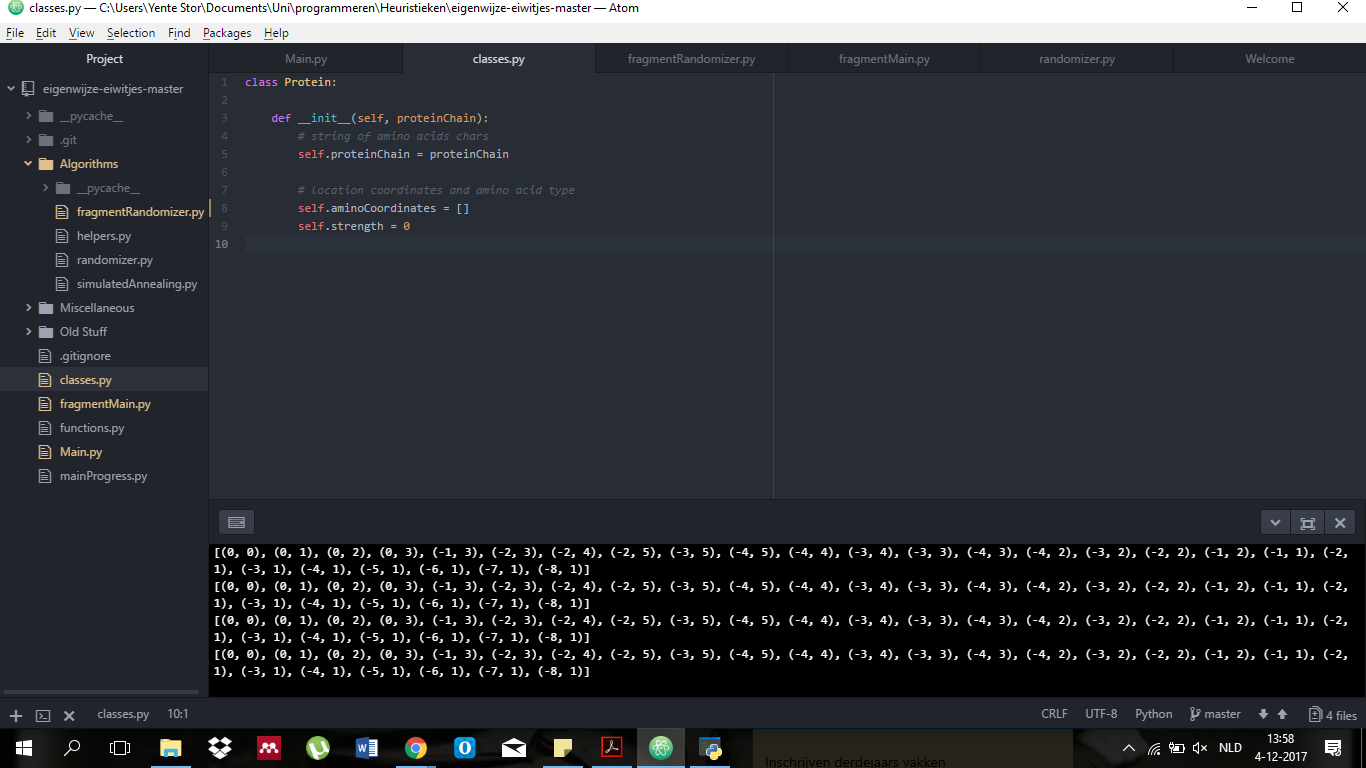
****

- **Visualisatie voor 2D is af, visualisatie voor 3D is af maar nog niet ideaal (de diepte is nog niet heel goed te zien). Daarnaast hebben we een visualisatie van de tijd die het kost om bij een beter vouwing te komen.**

* **2D visualisatie:**
* **Readme af :**

Punten Infrastructuur: 4

DATASTRUCTUUR

* **Eiwitclasse   
  **
* **Alleen de beste vouwing wordt steeds opgeslagen in de randomizer functie.**
* **Alle algoritmes kunnen gebruik maken van de functies in de helpers file. Over het algemeen zijn ze breed toepasbaar, maar kleine verschillen in output kunnen met een extra argument zoals gewenst verkregen**

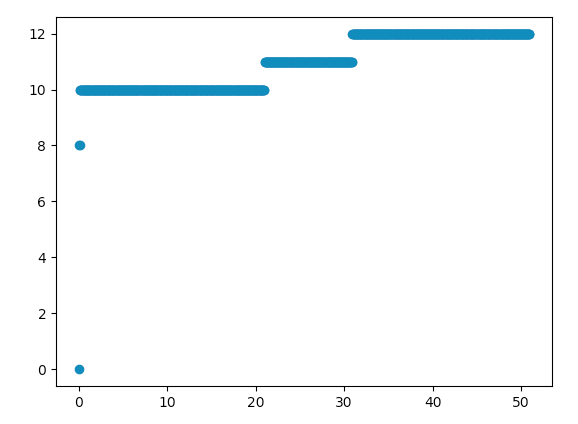
**worden. -> possibilityCheck & validityCheck bijv.**

* **Ook de functies in functions.py kunnen vaker gebruikt worden. In de algoritmes wordt de calculateFolding gebruikt als scorefunctie. In Main.py wordt visualizeFolding aangeroepen, deze geeft in een figuur de resultaten weer.**

Punten Datastructuur: 4

EXPERIMENTATIE

* **Om de snelheid en efficiëntie van de verschillende algoritmes te analyseren, runt Main.py ook een timer en verkrijgt deze info over het gerunde algoritme (wanneer welke verbetering op de score gevonden is). Deze resultaten worden in een grafiek weergegeven. Zo kunnen we zien hoelang het duurt voordat er een betere oplossing gevonden is, en hier onze ‘tries’ op afstellen. Hier is het random algoritme bijv. 1000x gerund met 100 tries per run. resultaten, vergelijkingen, en conclusies**

****

**Punten Experimentatie: 1**

ALGORITME

**Alleen nog een randomizer functie**

**Punten Algorimte: 1**