Outlook

Re: Opdracht Pygame Blackjack

Van Thomas Laisnez <thomaslaisnez@live.be>
Datum Di 31-12-2024 15:32
Tot Lore Laisnez <lore.laisnez@student.ucll.be>

Dag Lore,

Ik heb eens naar de code gekeken en heb hierbij wat feedback hieromtrent! Aangezien ik zelf weinig kennis heb van pygame zelf is het denk ik voornamelijk iets algemenere feedback, maar dat is naar mijn mening daarom ook niet per se minder waardevolle feedback... Dus bij deze:

- Je stuurde het project door via een zipje, en daar ga ik meteen mijn eerste feedback over geven: leer zo snel mogelijk een version control systeem gebruiken (lees: git) en hou je project bij op een repository (lees: github). Dit is een veilige en simpele manier om code bij te houden, op een handige manier je geschiedenis ook bij te houden van updates die je gemaakt heb, bij uitstek de ideale manier om samen te werken (wat je ongetwijfeld in de toekomst ook nog zal moeten doen) en, voor gevallen zoals dit, ook een handige manier om je code te delen met andere mensen en feedback te vragen. In het algemeen zou ik stellen dat git een industrie-standaard is op dit moment (en als git niet gebruikt wordt, zal het wel een ander systeem zijn dat gelijkaardig is). Helemaal niet zo moeilijk om te leren, maar maakt je leven een pak simpeler! Je hebt een aantal interactieve sites die je de basis van git leren (en de basis is zeker al voldoende om te starten, de rest leer je dan wel al doende), zoals https://learngitbranching.js.org/
- Er zijn een hele hoop dingen in programmeren die specifiek tot een bepaalde taal toebehoren, of op een bepaalde manier gedaan worden binnen een soort 'ecosysteem'. Python is daar zeker en vast een van, dat op veel vlakken dingen heeft die door praktisch iedereen als 'standaard' aanvaard worden, deze worden vaak de 'pythonic' manier genoemd. Als je dus zoekt naar hoe je iets best kan doen in Python (maar je weet wel hoe het moet vanuit bvb een andere taal), dan kan je altijd eens op zoek gaan naar hoe de meeste mensen het in Python zouden aanpakken... Hieronder zijn een aantal dingen die ik opmerk, die daaronder zouden vallen:
    - Je filename is Blackjack.py, in python gebruiken we zelden (of nooit) hoofdletters in filenames of veel andere dingen. In talen zoals Java en C# is PascalCase de standaard, in Python gebruiken we meestal snake_case.
    - Als je in Python comments schrijft, dan gebruiken we meestal docstrings: https://peps.python.org/pep-0257/. Ik zie dat je hier en daar wat commentaar bij je functies schrijft (wat zeker en vast iets goed is!) maar ik raad je dus aan om te kijken naar hoe dat normaal in Python aangepakt wordt. Een korte one-liner per functie die duidelijk toont wat er gebeurt in de functie is altijd handig om te hebben.
    - Nog iets handig om te hebben, en misschien nog niet meteen een standaard (maar zou het naar mijn mening beter wel zijn): Type Hinting! Net zoals je in talen als Java en C# al hebt, kan je tegenwoordig in Python type hinting doen, wat betekent dat je eigenlijk in je functieargumenten en returns al aangeeft wat de functie verwacht en zal teruggeven... Een dom voorbeeldje:

```
def get_score_string_for_player(score, player):
    """Return a string with the score of the player."""
    return f"{player}'s score: {score}"
```

```
def get_score_string_for_player(score: int, player: str) -> str:
    """Return a string with the score of the player."""
    return f"{player}'s score: {score}"
```

Maar zoals je zelf ook wel kan zien, geeft dit veel beter aan welke argumenten je kan meegeven en wat je terug zal krijgen uit deze functie. Vele IDE's kunnen hier ook al goed mee overweg en gaan hier ook rekening mee houden bij automatische aanvullingen e.d.

- In het algemeen is het best om variabelen zoveel mogelijk te beperken in scope. Je declareert er een hele hoop in het begin van je file, en daar staan er een stuk bij (zoals player_score, dealer_score, de hands, ...) die eigenlijk enkel binnen de game-loop nodig zijn. Ik zou die dus daar ergens in steken, in plaats van zo een grote lijst aan variabelen te initialiseren in het begin, dit zou normaal ook de leesbaarheid van de code wat moeten verbeteren.
- De functie show_cat aanvaard een parameter 'key', maar dat zijn hier zowel strings als integers. In de meeste gevallen is het beter om je te beperken tot een bepaald type, zowel voor leesbaarheid als voor mogelijke errors die je zou kunnen tegenkomen. Je kan altijd een 2de 'tussen'functie maken die de omvorming doet van 1 type naar het andere, om het toch mogelijk te maken om iets op te roepen. Wat ook iets handig is voor een gelijkaardige functie is een enum-type. Je kan dit eens opzoeken en eventueel gebruiken in de plaats van de strings en int's die je nu gebruikt. Met zo een enum kan je eigenlijk gemakkelijk een soort 'contract' maken van wat er geldige opties zijn, en daarop reageren.
- In de functie deal_cards, gebruik je de pop() functie. Dit returnt ook de value die je uit de lijst popt, dus je zou gemakkelijk op 1 lijn zowel de value kunnen pop-pen, en deze dan in current_hand appenden.
- Probeer functies zoveel mogelijk klein te houden en specifiek hun taak te geven - dit is een stuk persoonlijk, maar voor de leesbaarheid verkies ik liever 4 kleine functies van 6 lijnen, dan 1 grotere van 20. Ook al is dat in totaal meer lijnen code, je wilt er vooral voor zorgen dat andere mensen dit ook kunnen lezen zonder dat ze alles gaan moeten uitzoeken. Een voorbeeld hiervoor is check_endround - hier gebeuren een hele hoop dingen, zoals kijken wat de scores zijn, bepalen wat het resultaat is, er worden totals bijgehouden, geld wordt aangepast, er worden wat dingen getoond, en een hele hoop dingen die teruggegeven worden. Ik denk dat er hier wel wat opkuis mogelijk is dat het uiteindelijke resultaat leesbaarder gaat maken.
- Ook de gameloop kan wat opgesplitst worden in verschillende kleinere functies die allemaal hun eigen taak hebben, want dat is ook een serieuze blok code.

Dit is mogelijks nog een interessante resource om te lezen: https://testdriven.io/blog/clean-code-python/ Er zijn daar zeker nog honderden andere artikels (en meningen) over te vinden, maar over veel algemene dingen is het grootste deel van de Python community het wel eens, en dat zijn zeker geen slechte dingen om rekening mee te houden.

Voor de rest wel een tof projectje, en ik zou nu ook wel eens met pygame willen gaan prutsen! Hopelijk ben je wat met mijn feedback (zowel de feedback direct over de code, als de meer algemene) en leer je er iets van hier of daar!

Met vriendelijke groeten,

Thomas

Van: Lore Laisnez <lore.laisnez@student.ucll.be>
Verzonden: maandag 30 december 2024 12:33

Aan: thomaslaisnez@live.be <thomaslaisnez@live.be>
Onderwerp: Opdracht Pygame Blackjack

Dag Thomas

Zoals afgesproken stuur ik in bijlage mijn project door.
De opdracht was om aan de hand van deze video: https://www.youtube.com/watch?v=e3YkdOXhFpQ een werkend Blackjack spel te programmeren en hier nadien zelf nog uitbreidingen aan toe te voegen.

Zou je het zien zitten om deze code kritisch na te kijken en je bevindingen door te sturen?

Alvast heel erg bedankt voor de moeite!

Met vriendelijke groeten
Lore