PO Informatica Galgje (met Python)

In deze opdracht gaan we het galgje-spel bouwen: een persoon speelt tegen een computer. Deel de opdracht op in stukjes. Probeer het steeds zo te programmeren dat je tussendoor zo veel mogelijk stukjes kunt testen zonder dat het spel helemaal af hoeft te zijn. Zo weet je zeker dat je altijd een werkend product hebt. Hoe verder je komt, hoe hoger je cijfer. Het is de bedoeling dat je zo veel mogelijk **zelfstandig** aan de slag gaan, zonder te veel hulp van de docent. Help elkaar als je vastloopt. Laat zien wat je geleerd hebt, dus ook hoe je fouten uit je eigen code kunt halen.

Toelichting opdracht

Inleveren op: donderdag 14-juni om 23:59 (Te laat inleveren betekend een half punt aftrek van je cijfer, met daarna per 24 uur weer een half punt aftrek)

Een **.zip** bestandje inleveren bij Montiplaza 'inleverdocumenten' met:

1. Verslag:
    - (een foto van) je ontwerp/plan en/of stroomdiagram
    - Korte uitleg van wat wel/niet werkt, en welke uitbreidingen je hebt toegevoegd.
2. Je python code.

Je mag elkaar helpen, maar iedereen moet zijn eigen code schrijven en inleveren. Kopieer niet van iemand anders! Plagiaat wordt niet getolereerd.

Beoordeling:

Je **proces** wordt beoordeeld op:	
Voortgang Plan/Ontwerp	Je werkt zelfstandig en lost zelf (of met andere leerlingen) fouten in de code op. Een plan/ontwerp is vooraf gemaakt en maakt duidelijk hoe en welke afzonderlijke brokken code je in welke volgorde aan gaat werken/testen (zie **Deel A**)
Je **product** wordt beoordeeld op:	
Volledigheid & originaliteit	Hoe verder je komt qua functionaliteit (uitbreidingen, originele invulling), hoe hoger je cijfer. Je uitbreidingen staan kort beschreven in je verslag.
Correctheid Stijl	Code werkt precies zoals verwacht (volgens omschrijving **Deel C**). Je programma is robuust en geeft duidelijke foutmeldingen bij onvoorziene omstandigheden (bv. ongeldige invoer). In je verslag staat wat wel/(nog)niet goed werkt. Code is makkelijk te lezen en begrijpen, voorzien van uitstekende commentaar en logische naamgeving (conventies).
Constructie	De kwaliteit van de code: het is duidelijk, efficiënt, elegant, logisch en goed gestructureerd. Er wordt gebruik gemaakt van loops, condities. Globale variabelen staan bij elkaar boven aan je code. VWO: Er wordt gebruik gemaakt van functies en parameters. Functies staan bij elkaar, bovenaan de code. Gebruik van 'harde' waardes en globale variabelen worden zo veel mogelijk vermeden.

Goed programmeren gaat dus **niet alleen** om een werkende **oplossing**, maar juist om een **degelijke aanpak en oplossing. Laat zien wat je geleerd hebt!** Ga gestructureerd te werk, bedenk uit welke losse onderdelen je programma bestaat en pak ze los van elkaar aan.

DEEL A: ONTWERP

Voordat we gaan beginnen met het spel ga je eerst een stroomdiagram tekenen waarin je het spel in logische delen opbreekt. Gebruik het onderstaand uitleg van deel B om je ontwerp te maken. Hieruit blijkt welke onderdelen achter elkaar uitgevoerd worden, en waar herhaling zit en wat de voorwaarden voor herhaling zijn. Ook blijkt daar uit welke dingen je moet onthouden, oftewel, welke variabelen of constanten je hebt. Mogelijk zul je gaandeweg aanpassingen maken en tot nieuwe inzichten komen. Pas je stroomdiagram daar op aan.

Voor het (digitaal) tekenen van een stroomdiagram kun je gebruik maken van: http://course.cs.ru.nl/greenfoot/flowchart/flowcharttool.html (Je mag het ook op papier schetsen en een foto maken).

DEEL B: PROGRAMMEEROMGEVING OPZETTEN

Voor de PO maak je gebruik van Pycharm. Als je wat uitgebreidere code gaat schrijven zoals nu, is een uitgebreidere editor ook wel handig. PyCharm Edu kun je hier gratis downloaden: https://www.jetbrains.com/pycharm-edu/download/#section=windows.

1. Maak een mapje 'Galgje' in je **Montessori OneDrive** aan waar je gaat werken. Hier zet je al je bestanden. Zo kun je zowel vanuit thuis als op school bij, en even belangrijk, als je laptop kapot valt dan heb je je werk nog.
2. Start PyCharm op. Kies voor **New Project** en blader naar het mapje 'Galgje' in je Montessori OneDrive dat je net hebt aangemaakt. Dit kan eventjes duren.
3. Maak een Python bestand aan voor je code. Met je rechtermuisknop klik je op linksboven op je 'Galgje' map en kies voor een New -> Python File. Noem je bestand Galgje.

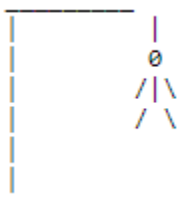
DEEL C: BASISPROGRAMMA

a) *Geheim woord:* Omdat we wel hebben geleerd om met lijsten te werken, maar nog niet met strings, gaan we het geheimwoord in een lijst onthouden. Maak een lijst en zet daarin de letters van het woord.
b) *Geraden woord:* Maak een 'geraden woord' met een juiste hoeveelheid puntjes.
c) *Gok:* Laat de gebruiker een gokletter in voeren.
d) *Controle:* Controleer of een gokletter in het geheime woord voorkomt. Zo ja, print dit. Vervang ook op **elke** juiste plek in het geraden woord het puntje door de letter. Zorg ervoor dat alle voorkomens van het letter vervangen wordt.
e) *Foute gok:* geef de gebruiker ook terugkoppeling als de gok niet in het woord voorkomt.
f) *Beurten:* Het spel zelf komt eigenlijk neer op een grote loop. Bij elke iteratie van de loop is de ´gebruiker een keer aan de beurt´. Aan welke condities moet zijn voldaan voordat de gebruiker nog een keer aan de beurt mag komen? Hoe bepaal je of 'ie niet al verloren heeft? En hoe bepaal je of 'ie gewonnen heeft?

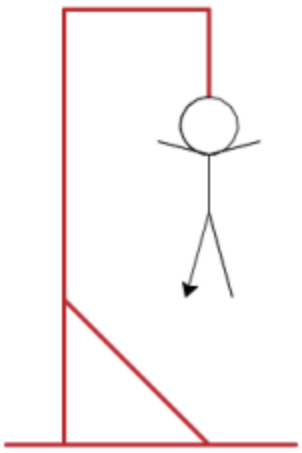
g) *Spelstand:* Toon de huidige spelstand (aantal pogingen, geraden woord tot nu toe) aan de gebruiker, en vraag om een nieuwe letter. Je hoeft (nog) geen rekening te houden met foutieve input (getallen, hele zinnen, of letters die al geraden zijn), dit mag je als uitbreiding doen.
h) *Einde spel:* Toon een bericht aan het einde wanneer de gebruiker heeft gewonnen of verloren. Waar kan je aan zien of de gebruiker won of verloor? Onderscheid deze twee gevallen. Laat ook nog even zien wat het te raden woord was.
i) *Bereken woord lengte:* Indien je dit nog niet gedaan hebt, pas je code zo aan dat het werkt voor een woord van elke willekeurige lengte. Gebruik dus geen getal voor de waarde, maar bereken deze.
j) *Willekeurige woorden*: Voeg meer geheime woorden toe (in een lijst) en laat het programma er eentje willekeurig kiezen. Tip: Tijdens het testen kan het handig zijn om de willekeurig gekozen woord af te drukken.
k) *Onvoorziene omstandigheden*: Zorg voor een goede afhandeling van fouten. Controleer, geef een duidelijke foutmelding en geef de gebruiker de mogelijkheid om de fout te herstellen. Bijvoorbeeld, bij een foutieve invoer (getallen, hele zinnen of helemaal geen invoer).

DEEL D: UITBREIDINGEN

- Wanneer de gebruiker een bepaalde letter al geraden heeft, is het een beetje zonde als het nog een gok kost wanneer hij of zij die letter per ongeluk herhaalt. Hoe kan je handig bijhouden welke letters al geraden zijn? Bedenk wanneer je een letter toe moet voegen, en voorkom dat een letter twee keer fout gerekend wordt.
- Je kunt ook een tweede speler vragen om een geheimwoord in te vullen. Zo kan je het spel met twee mensen spelen. Zorg er wel voor dat je het scherm even leeg maakt zodat de ander niet ziet wat je hebt ingevuld. Dat kan natuurlijk met een aantal lege regels te printen, maar dat kan ook netter. Met de juiste zoektermen in Google moet je het juiste commando wel kunnen vinden.
- Teken een galg door een combinatie van streepjes, sterretjes, nulletjes, etc. af te drukken.



- Om je code voor het tekenen van de galg overzichtelijk te houden is het goed om hier een aparte functie van te maken. Welke informatie heeft deze functie allemaal nodig? Hoe kan je deze functie op een nette manier opschrijven?
- Combineer wat je geleerd hebt bij TurtleGraphics met het spel wat je gemaakt hebt om een galg te tekenen. Ga na waar je precies een stukje moet tekenen. Om de venster open te houden moet je aan het **einde van je code** turtle.mainloop() aanroepen (maak geen gebruik van turtle.done()).



Figuur 1: Een voorbeeld van een leuke galg

- Voeg zelf iets origineels of iets slims toe waardoor je programma meer kan of waardoor je code eleganter of efficiënter wordt.