ZUYD

Casus Programmeren

Introductie voor Python

Naam: _____

Auteur: Jup Wijnen
Zuyd Hogeschool

-

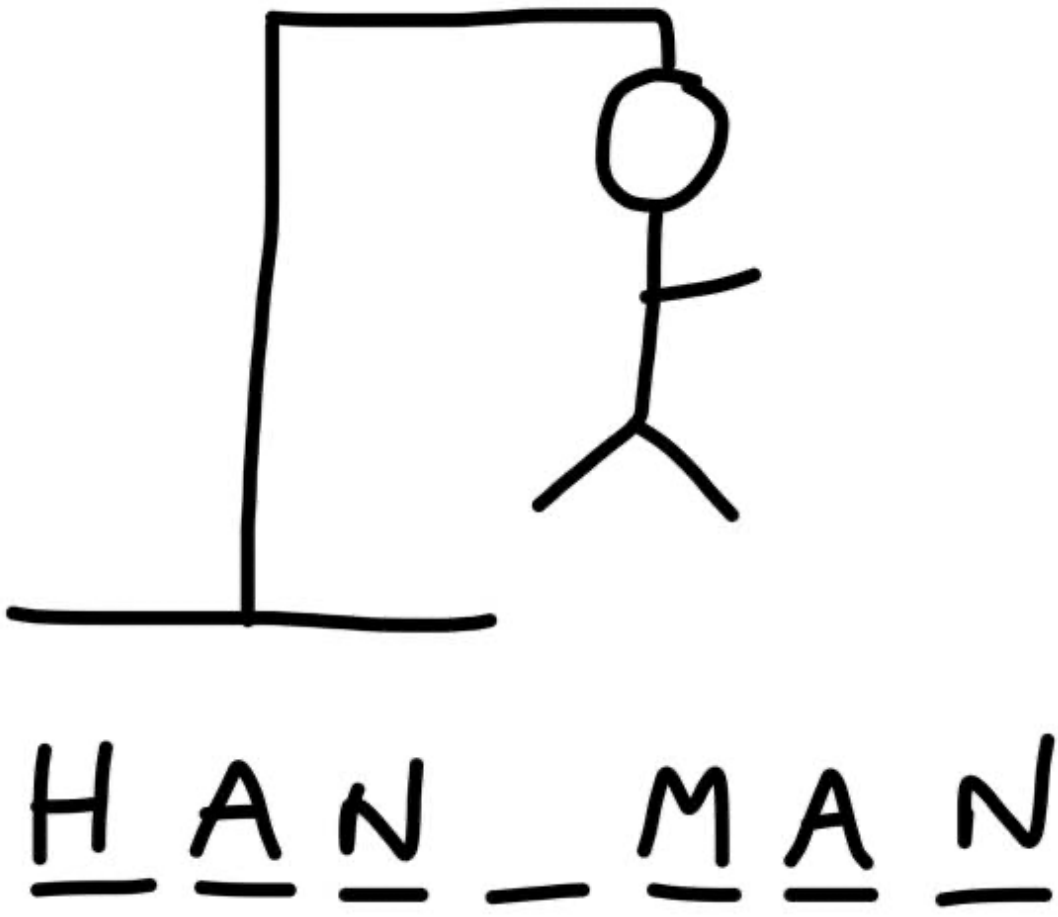
Proefstuderen
Datum:____/____/_____

Inhoudsopgave

1. Introductie

In deze casus wordt een simpele vorm van het spel 'Galgje' geprogrammeerd met de taal **Python**. Python maakt gebruik van een '**syntax**'. Dit is de manier waarop we de code moeten schrijven, de **bouwstenen** van ons programma. Python is ontworpen met het oog op '**leesbare' code** te ontwikkelen. De casus wordt in klassikale setting uitgevoerd met begeleiding van een docent. Om de opdracht zo goed mogelijk te kunnen volgen, is het belangrijk de stappen samen met de docent te doorlopen op de website **colab.research.google.com**. Op deze manier is het mogelijk te coderen zonder programmatuur te hoeven installeren of eigen proceskracht te gebruiken. Meer informatie volgt hierover in de komende hoofdstukken.

In deze casus wordt de basisvorm van het spel '**Galgje'** geprogrammeerd. Het spel kan altijd uitgebreid of aangevuld worden met andere functionaliteiten. Bij deze opdracht richten we ons op het leren programmeren.



2. Doel

Het doel voor deze casus om de meest simpele vorm van het spel galgje te realiseren. Eerst zal globaal een '**plan van eisen'** opgesteld worden met de meest belangrijke eisen voor het programma. Vervolgens zal woordelijk de code opgezet worden zonder programmeertaal (dit noemt men **pseudocode**) en zal op basis van deze pseudocode de **pythoncode** geschreven worden.

Het programma kan later uitgebreid worden of eventueel 'mooier' gemaakt worden.

```
Give Word: debug
_
_
_
_
_
Next Guess: d
d
_
_
_
_
Next Guess: e
d
e
_
_
_
Next Guess: b
d
e
b
_
_
Next Guess: u
d
e
b
u
_
Next Guess: g
Congratulations!, you guessed the word: debug
```

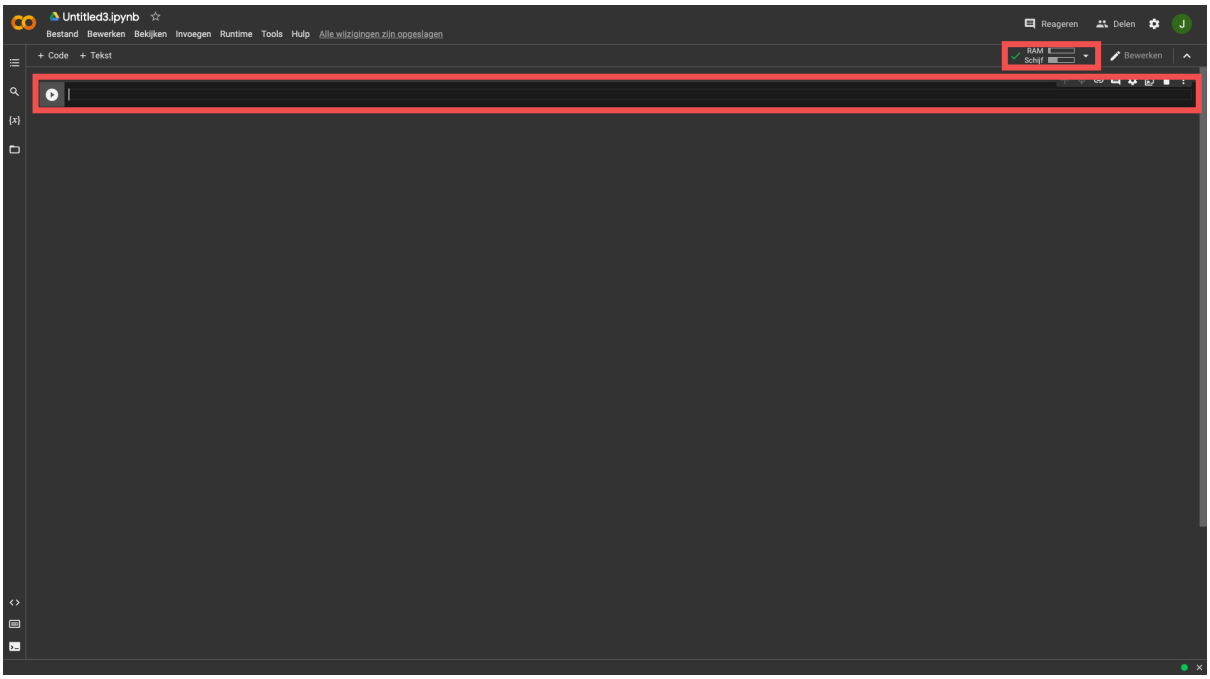
3. Voorbereiding

1. → Ga naar **colab.research.google.com**
2. → Als je nog niet ingelogd bent met je googleaccount, druk rechtsboven op **inloggen**
3. → Druk op **bestand > Nieuw Notebook**

4. → Zorg er als laatste voor dat de run-time geactiveerd is

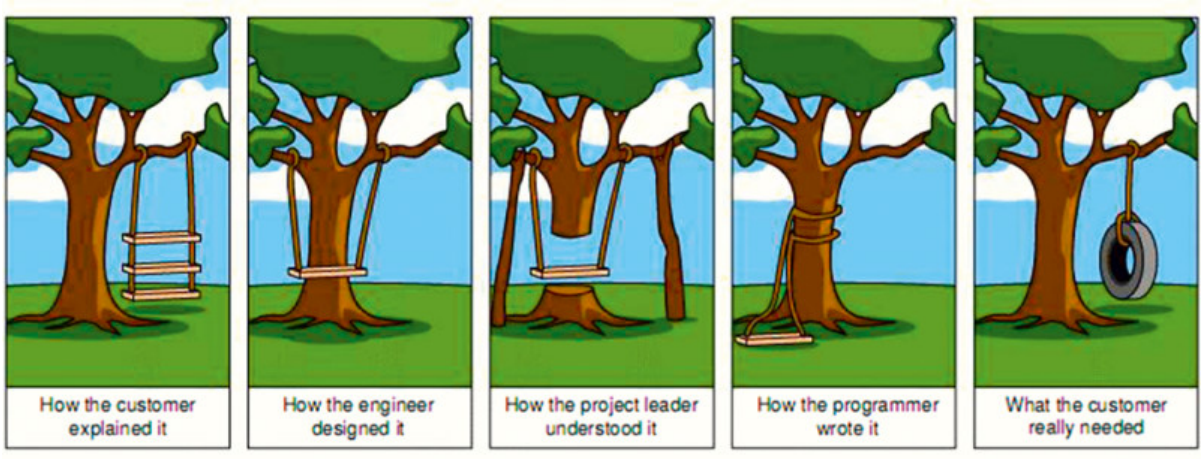
Runtime zijn resources (systeem-RAM en schijfruimte) die google gratis ter beschikking stelt om dit soort projecten uit te voeren!

5. → Zorg je ervoor dat je werkplaats eruitziet als volgt (let op de rood gearceerde gebieden):



4. Plan van Eisen

Om de wensen van een opdrachtgever het beste te begrijpen wordt een plan van eisen opgesteld. Dit plan van eisen helpt erbij om een beeld te krijgen van hetgeen dat de opdrachtgever bedoeld. Het plan van eisen komt vaak terug binnen de opleiding bij Zuyd Hogeschool Academie ICT.



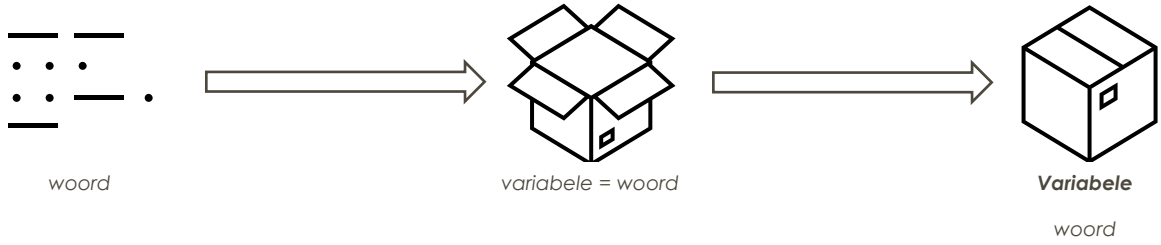
De onderstaande eisen noemen we **'must have'** eisen. Het project is niet succesvol als deze eisen niet gerealiseerd zijn.

Nr.	Eis
1	De gebruiker dient gevraagd te worden om een woord te geven dat vervolgens geraden moet worden
2	De gebruiker dient in staat te zijn letters in te geven met als doel het woord te raden
3	Het programma dient weer te geven hoeveel karakters geraden moeten worden
4	Het programma dient de geraden karakters op de juiste plaatsen (in het woord) weer te geven
5	Als de gebruiker meer dan X keer een foutieve letter invoert is het spel voorbij en verliest de gebruiker en krijgt hij of zij een melding (game over)
6	Bij het raden van een goede letter, wordt deze als correct geteld en weergegeven in het te raden woord.
7	Bij een foutief gerade letter, dient deze opgeteld te worden bij het aantal foutief geraden letters
8	Als de speler het woord geraden heeft binnen de perken van het aantal toegestane foutieve letters, dient het spel te stoppen en de gebruiker te informeren dat hij het spel heeft gewonnen

5. Pseudocode

5.1. Vraag de gebruiker voor het woord

In het begin van ons programma dient aan de gebruiker gevraagd te worden een woord in te voeren. Dit woord dient vervolgens opgeslagen te worden als 'variabele'. Dit doen we zodat we deze **variabele** (de dataset dus) later kunnen aanroepen. Zie een variabele als een doos waar data, code of andere dingen in worden opgeslagen.



5.2. Geef het speelveld weer

Het programma dient het speelveld weer te geven. In het geval van een 5 letterig woord dient het programma 5 lege plaatsen weer te geven. Om de code niet te complex te maken, gaan we geen extra opmaak toevoegen. Daarom ziet het speelveld als volgt uit:

_

_

_

_

_

Voor iedere letter in het woord:

Indien letter zich bevindt in de lijst met gerade letter, zet de letter op de plaats.

Zet anders een "_" neer voor een onbekende letter

5.3. Vraag de gebruiker om een letter te raden

Vervolgens moet het mogelijk zijn voor de gebruiker/speler om een letter in te geven om zo het woord te kunnen raden.

Gebruiker voert de 'guess' in:

Programma voegt de gerade letter toe in een lijst met gerade letters:

Als de letter niet in het woord zit:

Haal 1 van het 'aantal te maken fouten' af

Als het 'aantal te maken fouten' op 0 staat:

Game Over voor de gebruiker

5.4. Valideer de letter

Nu we alle gegevens hebben, is het belangrijk om te verschillende inputs met elkaar te vergelijken (het woord en de gerade letter)

Voor iedere letter in het woord:

Als de letter niet in de lijst met 'gerade letters' zit en de speler is nog niet game over:

De gebruiker is nog niet klaar en mag een nieuwe poging doen

Als elke letter in de lijst met 'gerade letters' zit en de speler is nog niet game over:

De gebruiker heeft het woord geraden en heeft het spel gewonnen

5.5. Einde van het spel

Bij deze fase dient het programma de gebruiker een melding te geven of hij verloren of gewonnen heeft.

Als de gebruiker het woord geraden heeft:

Geef een overwinning bericht

Als de gebruiker game over is:

Geef een game over bericht

5.6. Eindresultaat

Zie hieronder het eindresultaat van de pseudocode. Houdt deze code constant naast de programmeertaal om de relatie te kunnen zien.

Vraag Gebruiker voor woord en sla dit op als 'woord'

Voor iedere letter in het woord:

Indien letter zich bevindt in de lijst met gerade letter, zet de letter op de plaats.

Zet anders een "_" neer voor een onbekende letter

Gebruiker voert de 'guess' in:

Programma voegt de gerade letter toe in een lijst met gerade letters:

Als de letter niet in het woord zit:

Haal 1 van het 'aantal te maken fouten' af

Als het 'aantal te maken fouten' op 0 staat:

Game Over voor de gebruiker

Voor iedere letter in het woord:

Als de letter niet in de lijst met 'gerade letters' zit en de speler is nog niet game over:

De gebruiker is nog niet klaar en mag een nieuwe poging doen

Als elke letter in de lijst met 'gerade letters' zit en de speler is nog niet game over:

De gebruiker heeft het woord geraden en heeft het spel gewonnen

Als de gebruiker het woord geraden heeft:

Geef een overwinning bericht

Als de gebruiker game over is:

Geef een game over bericht

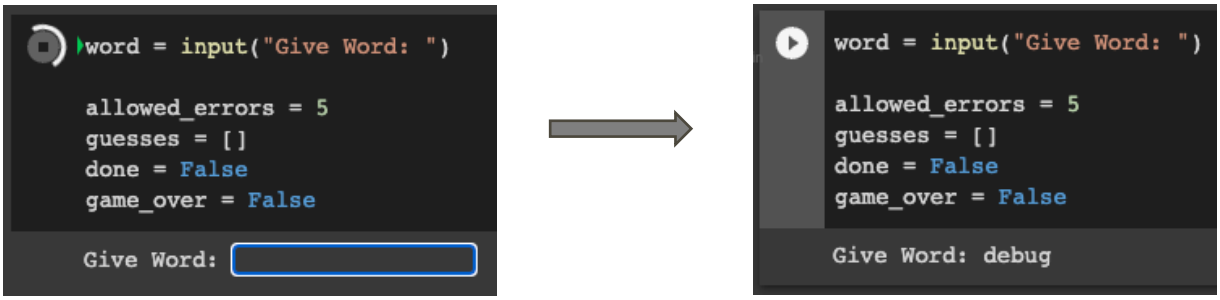
6. Code

6.1. Game Setup

Als allereerste dient de gebruiker gevraagd te worden voor een woord dat vervolgens geraden dient te worden. Dit woord wordt opgeslagen als 'variabele' met de naam '**word**'. Hierna worden nog een aantal andere variabelen opgezet die wel later in de code nodig gaan hebben.

`word = input("Voer woord in: ")`	Door te zeggen '**word =** ', vertellen we de computer dat we een variabele aanmaken genaamd 'word' en we er iets willen instoppen (=). Vervolgens zeggen we **input()** omdat we willen dat de gebruiker iets ingeeft. Hetgeen dat tussen aanhalingstekens staat tussen de haakjes is de instructie die de gebruiker ziet.
`allowed_errors = 5`	Maximaal toegestane fouten
`guesses = []`	Lege lijst waarin de gerade letters komen
`done = False`	Indicator of het spel voorbij is
`game_over = False`	Indicator of de speler game over is

*Druk vervolgens op **shift + enter** om deze regel te runnen en een nieuwe 'cel' aan te maken. Standaard wordt bij het programmeren de code volledig gerund, maar omdat nu gewerkt wordt in de Colaboratory van Google kunnen we lossen gedeeltes van de code runnen. Deze gedeeltes delen we op door shift + enter te doen. Nu vraagt de computer om een woord in te geven. In dit voorbeeld gebruiken we het woord '**debug**'.*



6.2. Speelveld

Vervolgens dient het speelveld opgezet te worden. We willen één blanco plaats ("_") per letter, die opgevuld wordt indien de letter geraden is (en dus in de lijst met gegokte letters staat). Het belangrijkste voor dit gedeelte van de code is dat dit vaker moet gebeuren. Dit noemen we een loop. Een loop stellen we in aan de hand van voorwaarden. In dit geval is de voorwaarde dat iedere keer een nieuwe beurt komt (en dus ook een nieuw speelveld) zolang dat het spel nog niet klaar is (daarvoor hebben we variabele '**done**' en op **False** gezet). Alles wat in deze 'while loop' (voorwaardelijke herhaling) valt, zetten we met een tabje insprong ónder deze while loop.

```
while not done:
```

Vervolgens stellen we dat we iets willen doen voor iedere letter in het woord. We willen per letter bekijken of we een blanco plaats moeten neerzetten ("_") of dat de letter geraden is en deze getoond moet worden. En omdat deze checks ook plaats vinden pér letter (dus in een loop), moeten deze ook een tabje ónder de 'for loop' (voor iedere letter in het woord)

```
    for letter in word:
```

Vervolgens gaan we per letter het volgende doen. Als de letter zich in de lijst 'guesses' bevindt dan laten we de letter zijn (print(letter)) en als de letter niet in de lijst 'guesses' staat, laten we een "_" zijn voor een blanco plaats die nog geraden moet worden. Achter '**letter**' schrijven we '**.lower()**'. Zoals je misschien al is opgevallen is Python hoofdlettergevoelig (zowel de code als de input die we vragen) daarom zetten we (als we werken met het controleren van letters) de letter in het klein. Als gecontroleerd wordt of een hoofdletter '**D**' in de lijst guesses staat terwijl er een kleine letter '**d**' in staat, zal de code deze letter niet vinden.

```
        if letter.lower() in guesses:
            print(letter)
        if letter.lower() not in guesses:
            print("_")
```

```
while not done:
  for letter in word:
    if letter.lower() in guesses:
      print(letter)
    if letter.lower() not in guesses:
      print("_")
```

LET OP! We doen hier geen ***shift + enter*** omdat de code binnen de while loop nog aangevuld moet worden.

Voor het woord **debug** zullen we dus het volgende speelveld nu krijgen:

_

_

_

_

_

6.3. Invoerbeurten van de speler

Net zoals in het begin (bij het invoeren van het woord) gaan we nu om invoer vragen voor een letter. **LET OP!** We zijn nog steeds aan het werken in de while loop, dus er moet een tab voor de code.

```
guess = input("Raad een letter: ")
```

Vervolgens gaan we deze ingevoerde letter toevoegen aan het de lijst met gerade letters (guesses). Ook hier doen we de letter transformeren naar een kleine letter in plaats van een grote letter voor het geval dat de gebruiker een hoofdletter ingeeft.

```
guesses.append(guess.lower())
```

Nu gaan we controleren of de gerade letter fout is, want in dat geval moet er een punt afgetrokken worden bij het aantal fouten die nog toegestaan worden (variabale allowed_errors). Als de gerade letter (guess), niet voorkomt in het woord (word) dan wordt een punt afgetrokken (allowed_errors -1).

```
if guess.lower() not in word.lower()
    allowed_errors = allowed_errors - 1
```

Op dit punt willen we ook controleren of de gebruiker zijn aantal toegestane fouten heeft opgebruikt. Indien hij of zij te vaak fout geraden heeft, is het game over. Hierbij controleren we of de variabele allowed_errors gelijk staat (==) aan 0. Omdat we bij iedere fout -1 doen bij allowed_errors komt deze automatisch op 0 uit als het totaal aantal toegestane foute antwoorden bereikt is. In dat geval zetten we variabele game_over op True. Deze wordt later

```
if allowed_errors == 0
    Game_over = True
```

```
word = input("Give Word:")
allowed_errors = 5
guesses = []
done = False
game_over = False

while not done:
  for letter in word:
    if letter.lower() in guesses:
      print(letter)
    if letter.lower() not in guesses:
      print("_")

  guess = input("Next Guess: ")
  guesses.append(guess.lower())
  if guess.lower() not in word.lower():
      allowed_errors = allowed_errors - 1
      if allowed_errors == 0:
          game_over = True
```

LET OP! We doen hier geen ***shift + enter*** omdat de code binnen de while loop nog aangevuld moet worden. De code zal op dit moment het speelveld weergeven en om input vragen. De letters zullen, indien correct getoond worden op het speelveld. Echter hebben we nog geen code geschreven voor de validatie om te kijken of de gebruiker gewonnen of verloren heeft.

6.4. Validatie

Bij de validatie van het spel gaan we kijken of het spel klaar is. Er zijn twee verschillende manieren waarom het spel kan eindigen. Het spel kan eindigen: de speler heeft alle letters geraden en heeft zijn maximaal aantal fout gerade letters niet overschreden, of het aantal maximaal fout te raden letters is overschreden. In dat geval is het game over. Omdat nu bepaald moet worden of het spel klaar is, wordt ervanuit gegaan dat het spel klaar is (done = True), en als de code een letter vindt die nog niet geraden is of de speler is game over wordt deze aanname terug veranderd (done = False). Dit willen we na iedere gak doen, dus ook dit komt terug in de while loop.

```
done = True
```

We gaan er dus vanuit dat het spel klaar is. Vervolgens gaan we controleren of er letters zijn die de speler nog niet geraden heeft (if letter.lower() not in guesses). Als er nog letters zijn die de speler nog moet raden en de speler is nog niet game over, dan mag de speler doorgaan met raden (done = False en game_over = False). Echter als de speler game_over is stopt het spel (done = True en game_over= True) en heeft de speler verloren.

```
for letter in word:
    if letter.lower() not in guesses and not game_over:
        Done = False
```

Druk nu op ***Shift + Enter*** om de code te starten en een nieuwe sectie te openen.

```
while not done:
  for letter in word:
    if letter.lower() in guesses:
      print(letter)
    if letter.lower() not in guesses:
      print("_")

  guess = input("Next Guess: ")
  guesses.append(guess.lower())
  if guess.lower() not in word.lower():
      allowed_errors = allowed_errors - 1
      if allowed_errors == 0:
          bgame_over = True

  done = True
  for letter in word:
      if letter.lower() not in guesses and not game_over:
          done = False
```

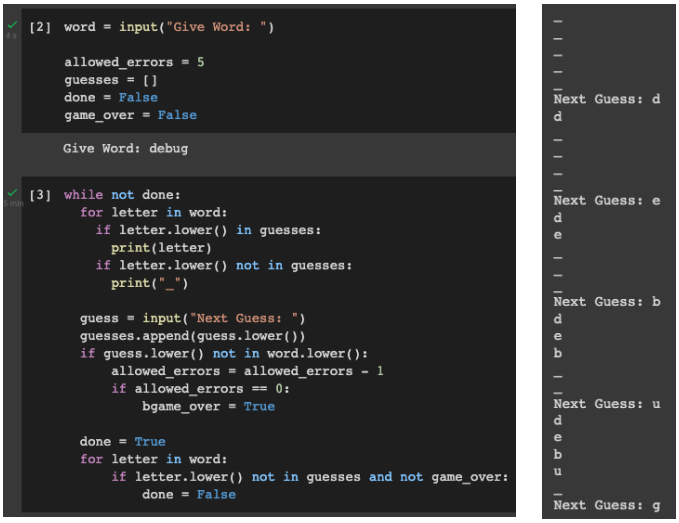
```
_
_
_
_
_
Next Guess:
```

6.5. Berichtgeving

Als laatste rest ons nog de speler te vertellen of hij gewonnen heeft. Als je op het einde van 6.4. het spel speelt en het woord raadt of game over bent, stopt het spel, maar weet je niet wat er is gebeurd. Om de speler te informeren over het einde van het spel hebben we twee meldingen: Een voor winst en een voor verlies. Als de speler klaar is (done = True) en hij is niet game over (not game_over) krijgt hij een melding dat hij gewonnen heeft.

```
if done and not game_over:
    print("Congratulations! You Won!")
if game_over:
    print("Game Over...")
```

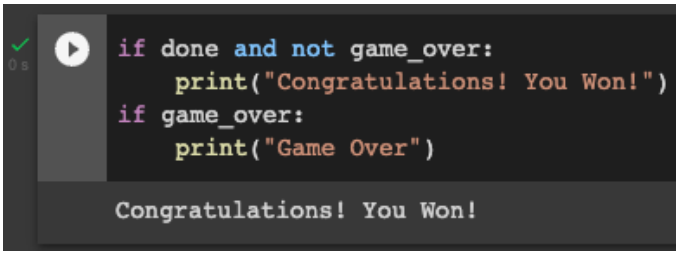
Als je beide secties gedaan hebt (het invoeren van het woord en het raden van het woord) staan groene vinkjes bij beide secties (zie onderstaande afbeelding).



```
[2] word = input("Give Word: ")

    allowed_errors = 5
    guesses = []
    done = False
    game_over = False

    Give Word: debug

[3] while not done:
      for letter in word:
        if letter.lower() in guesses:
          print(letter)
        if letter.lower() not in guesses:
          print("_")

      guess = input("Next Guess: ")
      guesses.append(guess.lower())
      if guess.lower() not in word.lower():
          allowed_errors = allowed_errors - 1
          if allowed_errors == 0:
              bgame_over = True

      done = True
      for letter in word:
          if letter.lower() not in guesses and not game_over:
              done = False
```



```
_
_
_
_
_
Next Guess: d
d
_
_
_
_
Next Guess: e
d
e
_
_
_
Next Guess: b
d
e
b
_
_
Next Guess: u
d
e
b
u
_
Next Guess: g
```



```
if done and not game_over:
    print("Congratulations! You Won!")
if game_over:
    print("Game Over")

Congratulations! You Won!
```

6.6. Eindresultaat

```
word = input("Give Word: ")

allowed_errors = 5
guesses = []
done = False
game_over = False

while not done:
  for letter in word:
    if letter.lower() in guesses:
      print(letter)
    if letter.lower() not in guesses:
      print("_")

  guess = input("Next Guess: ")
  guesses.append(guess.lower())
  if guess.lower() not in word.lower():
      allowed_errors = allowed_errors - 1
      if allowed_errors == 0:
          bgame_over = True

  done = True
  for letter in word:
      if letter.lower() not in guesses and not game_over:
          done = False

if done and not game_over:
    print("Congratulations! You Won!")
if game_over:
    print("Game Over")
```

Nr.	Eis	V
1	De gebruiker dient gevraagd te worden om een woord te geven dat vervolgens geraden moet worden	
2	De gebruiker dient in staat te zijn letters in te geven met als doel het woord te raden	
3	Het programma dient weer te geven hoeveel karakters geraden moeten worden	
4	Het programma dient de geraden karakters op de juiste plaatsen (in het woord) weer te geven	
5	Als de gebruiker meer dan X keer een foutieve letter invoert is het spel voorbij en verliest de gebruiker en krijgt hij of zij een melding (game over)	
6	Bij het raden van een goede letter, wordt deze als correct geteld en weergegeven in het te raden woord.	
7	Bij een foutief gerade letter, dient deze opgeteld te worden bij het aantal foutief geraden letters	
8	Als de speler het woord geraden heeft binnen de perken van het aantal toegestane foutieve letters, dient het spel te stoppen en de gebruiker te informeren dat hij het spel heeft gewonnen	

7. Verbeteringen aan de code

Nr.	Eis
1	Het programma laat het speelveld horizontaal (letters naast elkaar) in plaats van verticaal zien (letters boven elkaar)
2	Het programma laat de gebruiker weten als hij een foutieve invoer geeft bij het gokken van letters (meerdere letters, tekens of cijfers)
3	Het programma laat in een speelveld zien met een hangman die verschijnt bij foutief gerade letters
4	Als het spel afgelopen is, krijgt de gebruiker de keuze om te stoppen of het spel nogmaals te spelen
5	De gebruiker is in staat de moeilijkheidsgraad in te vullen (allowed_errors)
6	Als een letter een tweede keer ingevoerd wordt (een letter is al geraden), krijgt de gebruiker een melding dat deze letter al geraden is.