# Versiebeheer hand-out 1

Versiebeheer is een belangrijk begrip binnen de IT. Zonder versiebeheer ontstaat chaos binnen projecten en bij grote projecten kan dit het einde van het project betekenen. Online versiebeheer bevorderd ook samenwerking en professionalisering. Gedurende dit vak zullen jullie een set repositories maken die jullie portofolio uit zullen maken, dit portofolio zal ook jullie eindoplevering zijn.

Gedurende de lessen krijgen jullie hand-outs met daarin opdrachten en stap bij stap uitleg. Bij iedere opdracht staan tijdsinschatting van hoelang geschat wordt dat de opdracht zou duren, dit kan langer of korter maar geeft een indicatie.

## Inhoudsopgave



## Opdracht 1: Achtergrond informatie (5 – 10 minuten)

Om enige kennis te hebben over wat versiebeheer is lees dit artikel
https://www.atlassian.com/nl/git/tutorials/what-is-version-control

Lees alleen het stuk van “Wat is versiebeheer?” meer is niet nodig, de rest van de artikelen gaan over de basis vorm van Git via console commands en zijn dus overbodig voor dit vak. De informatie wordt niet getoetst maar is wel handig om te hebben.

## Opdracht 2: Maak je eigen GitHub account (5 – 10 minuten)

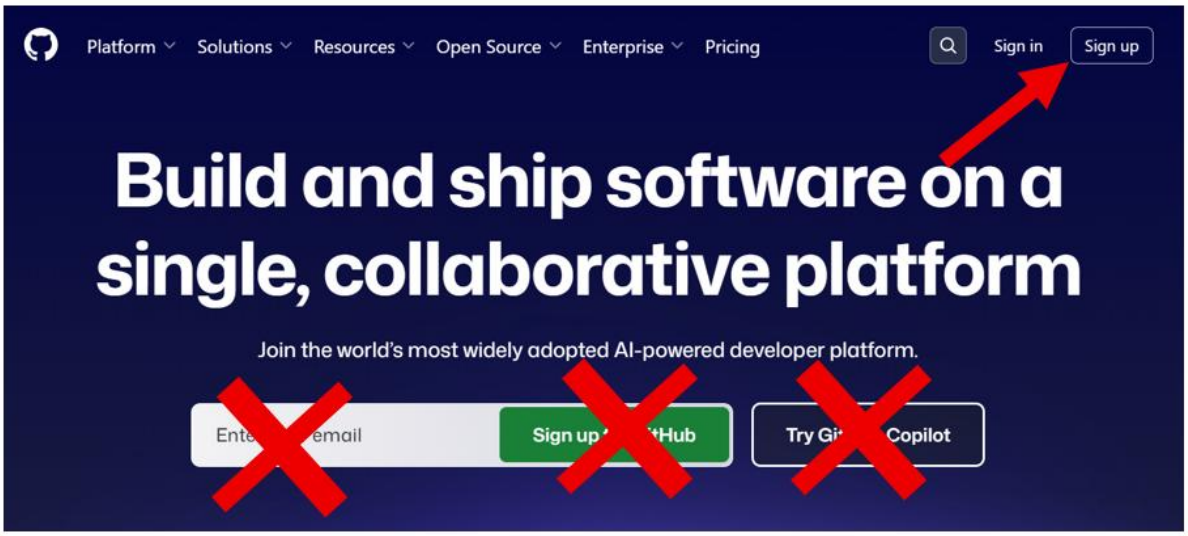
Omdat GitHub een van de meest gangbare versiebeheer systemen is gaan we GitHub gebruiken voor het vak versiebeheer. Een van de doelstellingen is om jullie een portofolio te laten opbouwen dat jullie later met trots kunnen laten zien bij sollicitatie gespreken.

**Stap 1: Zorg ervoor dat je toegang hebt tot een privé email/Google/Apple account**

Je gaat een van deze drie nodig hebben om een GitHub account te maken. Google en Apple is zeer makkelijk om te gebruiken en laat je toe het te koppelen. Het enige nadeel is dat als je de controle account wil wijzigen, dit meer stappen vereist dan met privé. Daarentegen kost inloggen op devices minder energie met Google of Apple. Alle drie hebben voor en nadelen, geen is beter dan de ander.

**Stap 2: Ga naar https://GitHub.com/ en klik op Sign up**

Als je liever een ander programma gebruikt mag dit in overleg, het moet equivalent zijn aan GitHubs functies. Bijvoorbeeld BitBucket kan maar google drive niet




**Stap 3: Gebruik je privé email om een account te maken of gebruik je Google/Apple account**

Als je een email adres gebruikt hou rekening met het feit dat je een email gebruikt dat ook na je opleiding beschikbaar is. Je kan je school account gebruiken maar hou rekening met het feit dat deze na je studie tijd niet meer beschikbaar is.

Je kan je privé email later koppelen aan je account maar doe dit bijtijds anders ben je alles kwijt!

**Stap 4: zorg ervoor dat je gebruikersnaam iets is dat mensen makkelijk kunnen koppelen aan jou of goede representatie heeft.**

Je GitHub zal in de toekomst functioneren als je portofolio met daarin je werkwijze, ervaring en gemaakte projecten. Een naam met scheldwoord of alleen onzin geeft geen goede indruk.

| Goed voorbeeld: | Slecht voorbeeld: |
|---|---|
| ArteGit | Afnjewkolnfoq |
| MariePortofolio | 13ra912eh |
| MillyDesmond | Username1234557 |
| HFDeVrees | John826496482 |

**Stap 5: Wachtwoord instellen**

In het geval dat je alleen een privé email gebruikt zorg ervoor dat er een sterk wachtwoord gebruikt wordt. GitHub forceert al een redelijk sterk wachtwoord maar het is aangeraden om een zeer sterk wachtwoord te gebruiken, bij voorkeur een random selectie van 15 symbolen/karakters. GitHub accounts staan vaak onder vuur van hackers omdat er vaak slechte spelers op het systeem willen. Een voorbeeld van een slechte speler is iemand die met toegang van je account je projecten delete. Een andere slechte speler zou je privé projecten op publiek zetten.

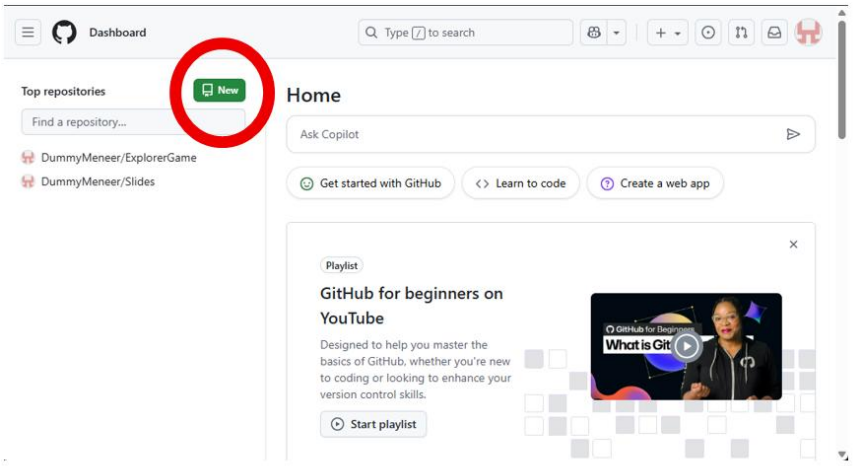

*Figure 1: credit aan hivesystems.com*

## Opdracht 3: een repository maken (10 minuten)

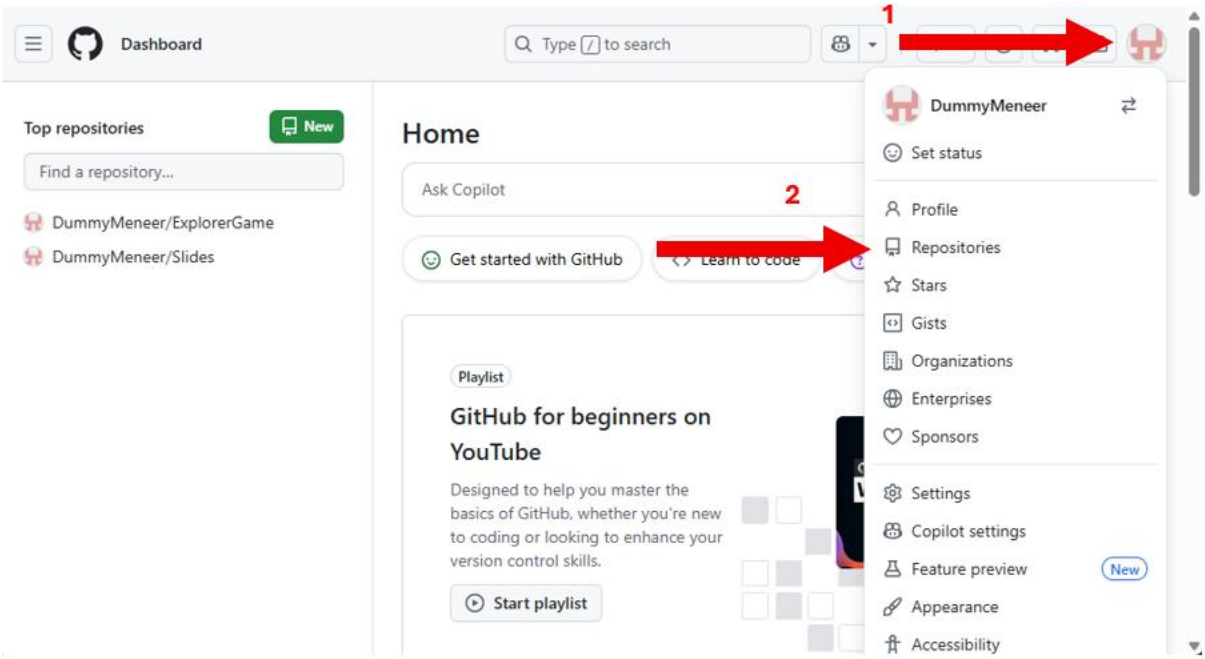
Een repository (of Repo voor kort) bevat vaak 1 project waar over een periode aan gewerkt wordt en waar soms verschillende versies nodig zijn. Een andere reden voor een repo is als een groep projecten een gedeeld onderwerp hebben. Als je je huiswerk van C# moet bewaren zou dat in een repo kunnen. Voor deze opdracht gaan we 1 repo maken, het onderwerp van de repo is een vrije keuze, dit mag een nieuw project zijn waar je aan gaat werken of over de opdrachten van een voorgaand vak, zolang het maar gerelateerd is aan een ICT project/programeer code project.

**Stap 1: maak een nieuwe repo**

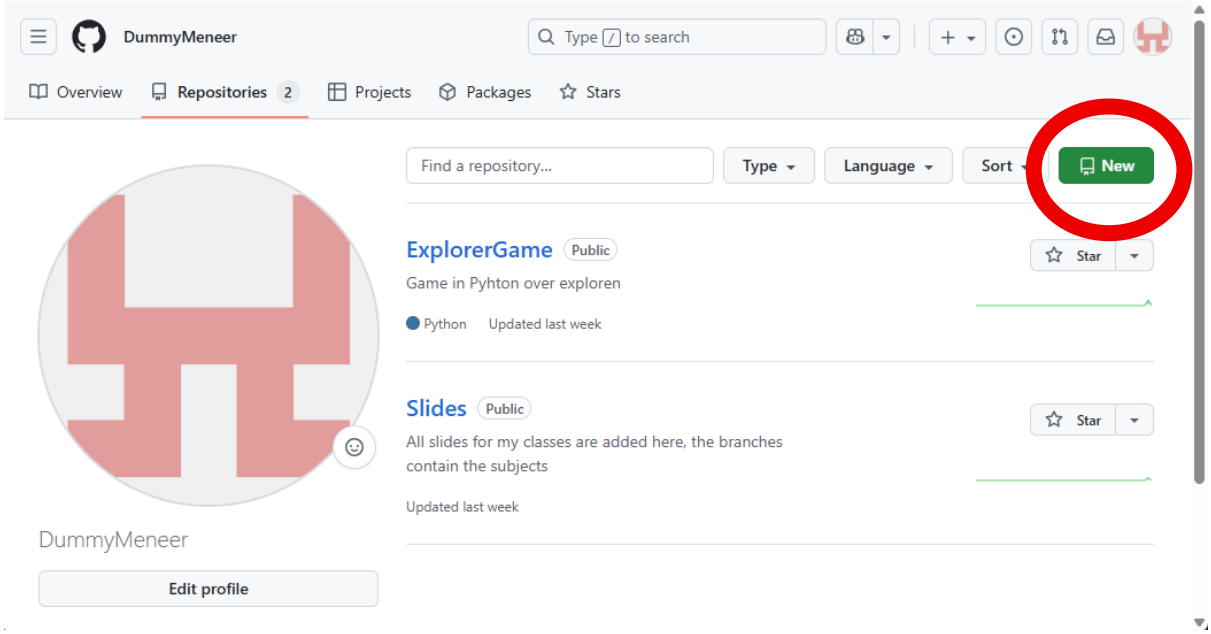
Druk op het GitHub logo (Zwarte cirkel met witte kat.) links bovenin. Je komt op het dashboard met een aantal vakken aan de linkerkant staat een lijst met top repositories. Aan de bovenkant van dit vak staat de groene knop met “New” erin. Deze klikken we aan.




Mocht het vak met top repositories er niet zijn kan je via je profiel rechts bovenin (Klik hierop dit opent een drop down menu) op repositories klikken.

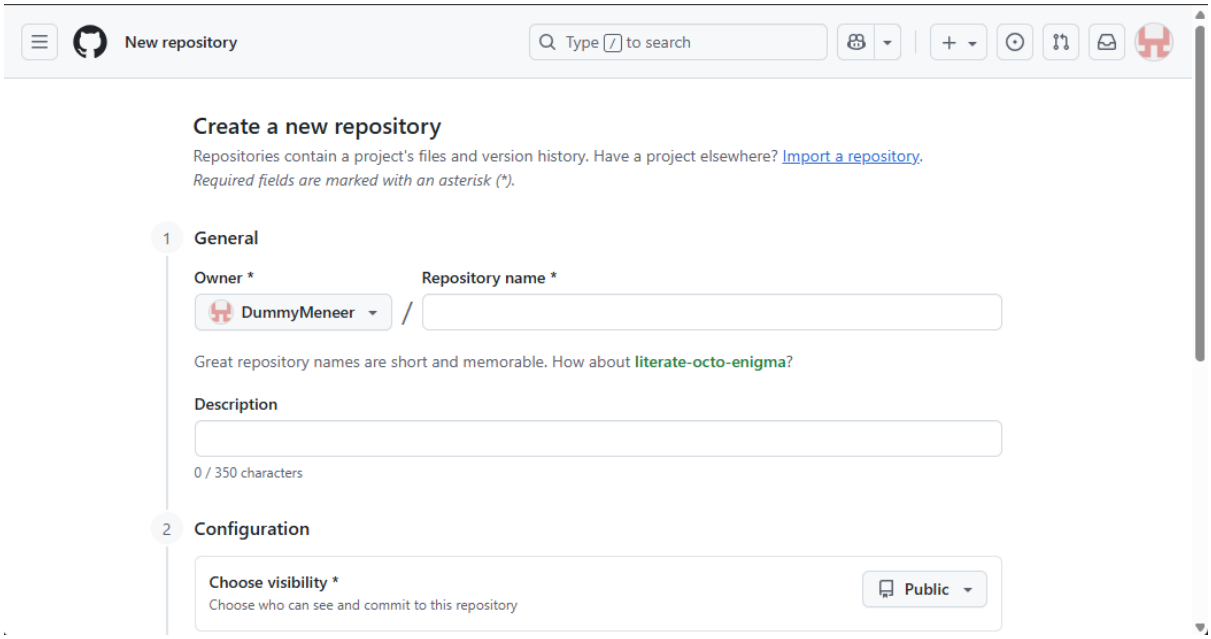


Dit opent de lijst met repositories:




Bij dit voorbeeld zitten er twee in maar bij nieuwe accounts zal er niks in zitten. Klik op de groene knop met het woord “New” erin.

Dit brengt je naar de volgende pagina:



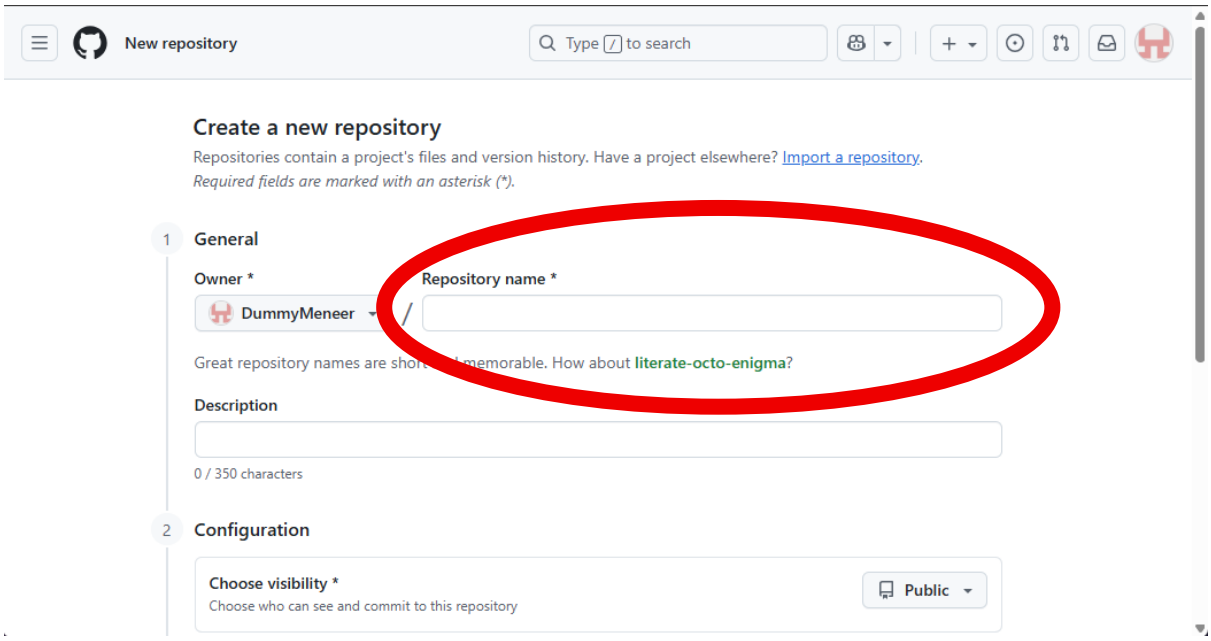
**Stap 2: geef de Repo een duidelijke naam**

Publieke Repos kunnen gezocht worden door bedrijven of andere programmeurs die op zoek zijn naar inspiratie, voorbeeld code of implementatie van een plug-in. Dit is een van de grootste redenen waarom er een duidelijke naam aan de repo moet zitten. De andere is om er voor te zorgen dat 3 jaar later je nog weet wat in welke repo zat. Het gebruik van Code namen voor projecten kan, als je dit doet zorg ervoor dat er dan in de beschrijving duidelijk staat wat dit project in houdt.

De naam van een repo wordt geschreven als: Gebruikersnaam/RepoNaam. Dit zorgt ervoor dat er meerdere repos met hetzelfde onderwerp kunnen bestaan. De gebruikersnaam staat al ingevuld je hoeft alleen de repo naam in te vullen.

| Goed voorbeeld | Slecht voorbeeld |
|---|---|
| Huiswerk python1<br>MySQL database Pokémon<br>Project Unity leren<br>Project ASML webserver<br>SAP Remote monitor plug-in<br>Snake game in Java | Py<br>Database<br>Leuk<br>Projectje<br>CSS<br>“Project over het verplaatsen van een slang die een appel eet en dan langer wordt en dood gaat als het zijn eigen staart eet” |

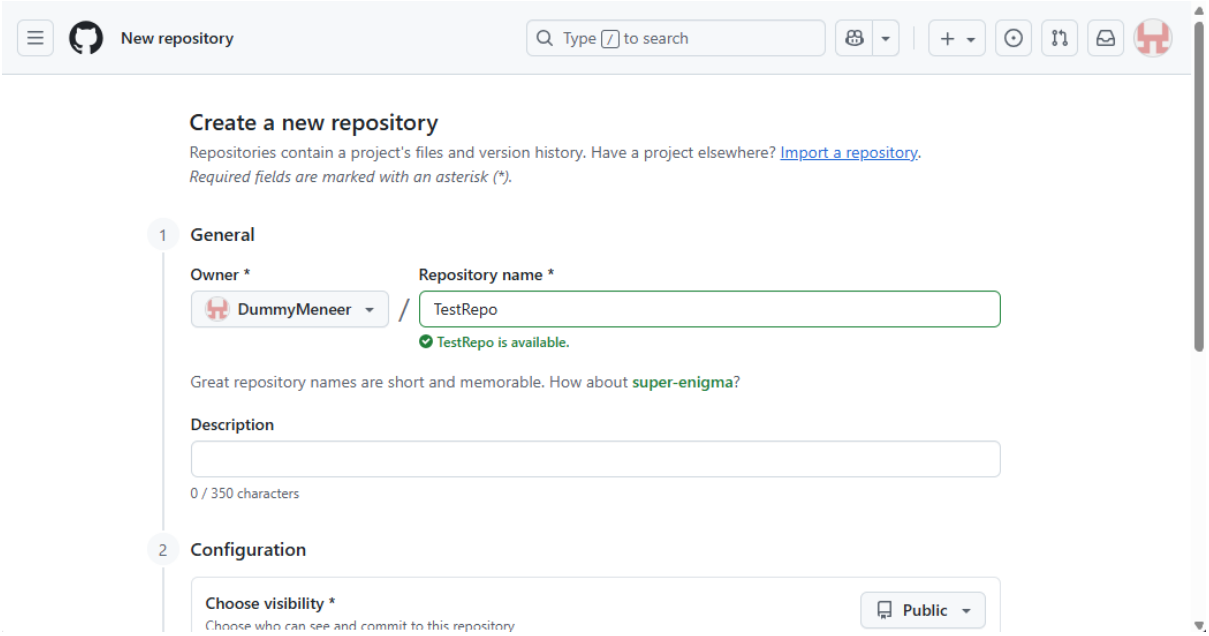
Voor deze opdracht noemen we de Repo “Mijn eerste repo” of “TestRepo”. In week 3 leer je hoe je een repo kan verwijderen. Voor wie al GitHub eerder heeft gebruikt en publieke repos heeft staan mag voor de opdracht een oude repo gebruiken voor week 1.



**Stap 3: Zorg ervoor dat je beschrijving logisch en nuttig is**

De beschrijving is optioneel maar het is wel verstandig om deze te schrijven aangezien we hieraan kunnen aflezen wat de details zijn van dit project. Details die mogelijk niet in de title staan zijn: Programmeer taal, welke OS gebruikt kan worden, of het project in een volwaardige staat is, installatie instructies en wat de geschiedenis is.

Voor deze opdracht zetten we er een simpele uitleg bij dat dit een repo is om te leren hoe Git werkt.



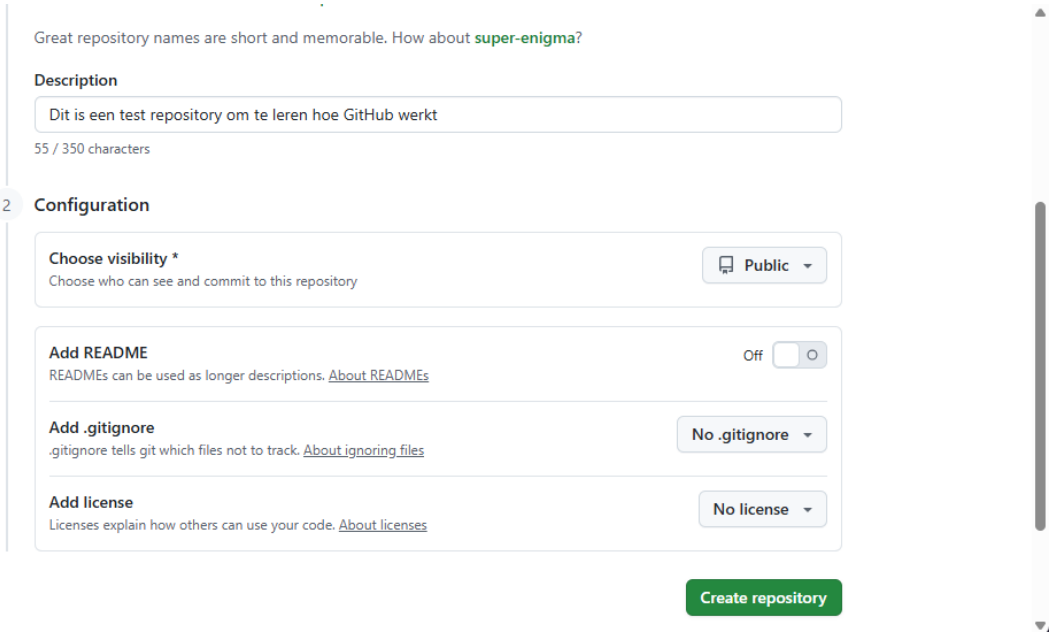

Mocht je beschrijving langer nodig zijn dan dat je ruimte voor hebt is dat geen probleem, zet hier een beknopte samenvatting en dan kan je later in de ReadMe de volledige beschrijving zetten.

**Stap 4: Maak de Repo publiek**

Een **private** repo is handig in sommige gevallen vooral als je een **privé** project gaat maken of als je een project hebt voor een bedrijf met algoritmes of beveiliging die geheim moeten blijven, denk aan bank apps of YouTube algoritmes. Voor dit vak gaan we alle oefen opdrachten en eindopdrachten **publiek** zetten. Dit betekent dat iedereen de repos kan zien en naar zoeken.

Belangrijk: **publiek** betekent niet dat iedereen er iets op kan plaatsen, alleen zichtbaar.

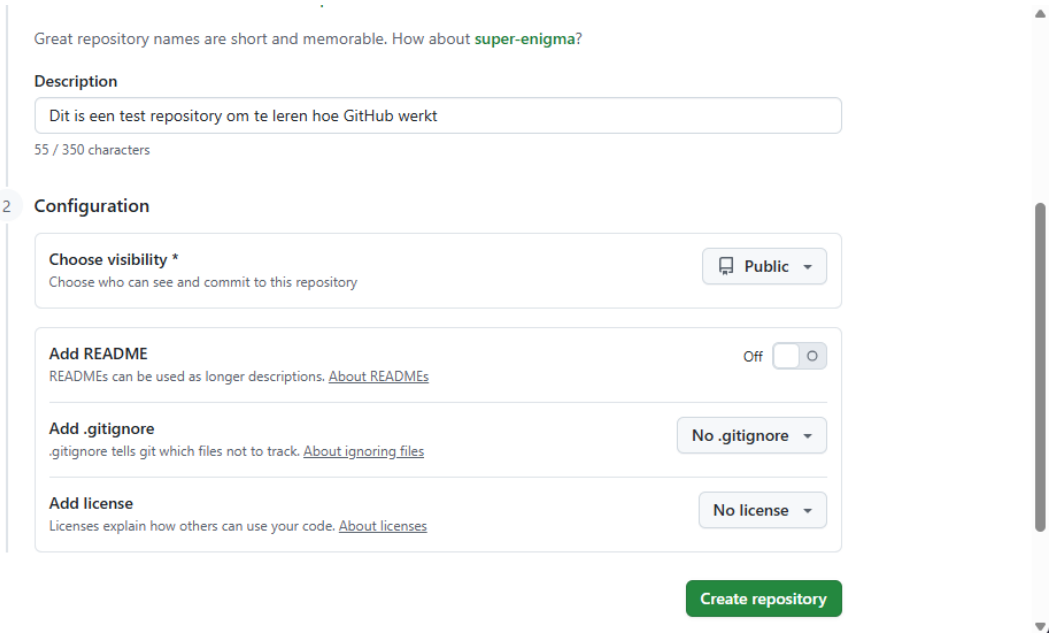
Onder het vak voor de beschrijving vind je de configuratie. Voor deze opdracht zetten we de **visibility** op **publiek**, later kan je het op private zetten als je dat fijn vindt.



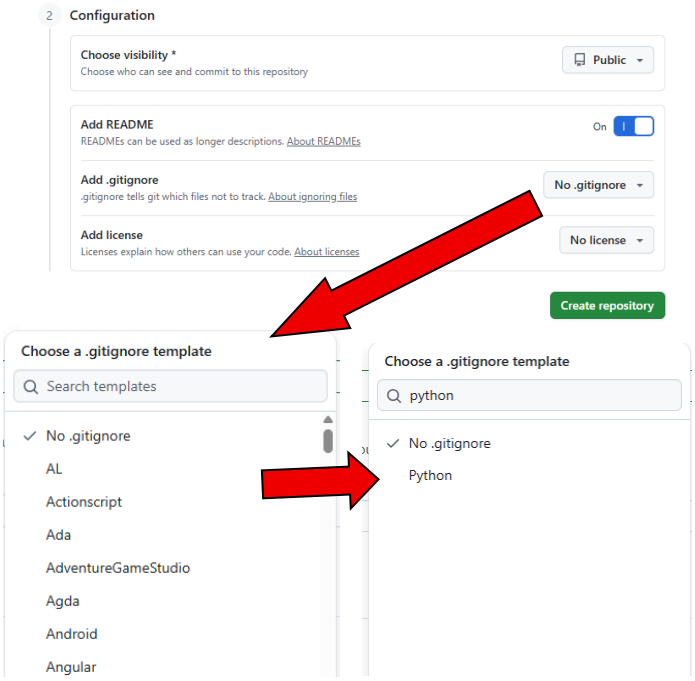

**Stap 5: Zorg ervoor dat er een ReadMe wordt gemaakt.**

Zoals in stap 3 al kort werd benoemd kan de **ReadMe** een belangrijk document zijn dat bezoekers meer kan vertellen. De **ReadMe** bevat alle instructies, detail informatie en opmerkingen over het project. Voor zeer kleine projecten kan er overwogen worden om de **ReadMe** niet toe te voegen echter is het aangeraden om dit standaard te doen.

Voor deze opdracht hoeven we niks in de **ReadMe** te zetten maar wel aan te laten maken.



### Stap 6: zorg ervoor dat er een gitignore wordt aangemaakt

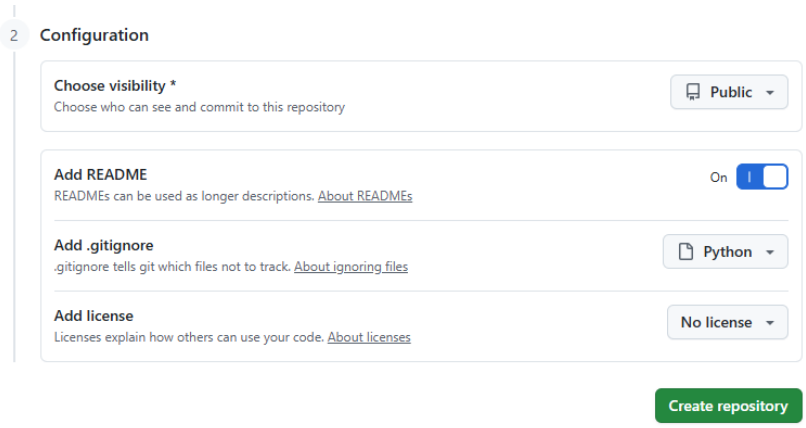



Een **gitignore** is een document wat je toelaat om bepaalde documenten te skippen die je anders wel zou meesturen. Voor grote projecten wordt er een compiler bijvoorbeeld aangemaakt en deze wil je niet meesturen. (Dit zijn soms honderden tot duizenden files die niet goed overdraagbaar zijn en onnodig zijn om op een repo te gooien.) Met een **gitignore** kan je het project gewoon uploaden en gaat Git uitfilteren wat wel en niet relevant is. Het is aangeraden om GitHub deze te laten genereren bij aanmaak van de repo aangezien het later toevoegen van de gitignore niet altijd goed werkt.

Voor deze opdrachten gaan we python code uploaden dus de **gitignore** zetten we op python. Klik op de drop down menu waar nu staat “No .gitignore” dit opent het menu. Hier kan je allerlei programmeer talen en omgevingen vinden. De lijst is lang dus het is aangeraden om de zoekbalk te gebruiken. Vervolgens moet python aangeklikt worden. Hierna staat python in de **gitignore**.

### Stap 7: Hou de license op “No license”

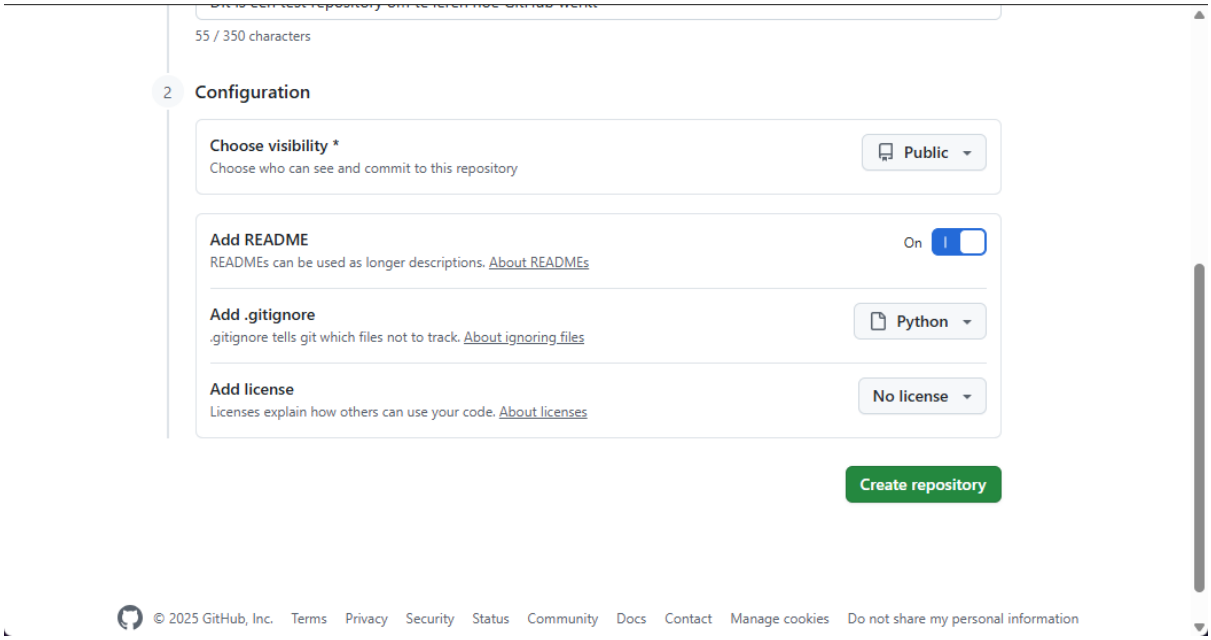
De **license** van een repo kan zeer belangrijk zijn voor projecten die door bedrijven zijn gemaakt of als het een open-source plug-in is. In het bedrijfsleven zal de **license** of gegeven worden via een tekst document of zal de teamleider deze erin plaatsen. De **license** wordt vaak ook besproken met een juridisch team om de juiste te kiezen. Voor individuele projecten zonder een bedrijf kan een eigenaar forceren dat bedrijven de code in de repo niet mogen gebruiken zonder geschreven toestemming van de eigenaar. Er zijn vele **licenties** die gebruikt kunnen worden echter voor deze repos zullen we geen licenties toevoegen.




Mocht je het leuk vinden kan je je erin verdiepen echter is dit niet een cruciaal element van het vak. We zullen dus de License houden op “No License”.

**Stap 8: Rond de repo af**

De Repo is nu in principe klaar om gemaakt te worden, druk op de “Create repository” knop. Dit kan enkele seconden duren.

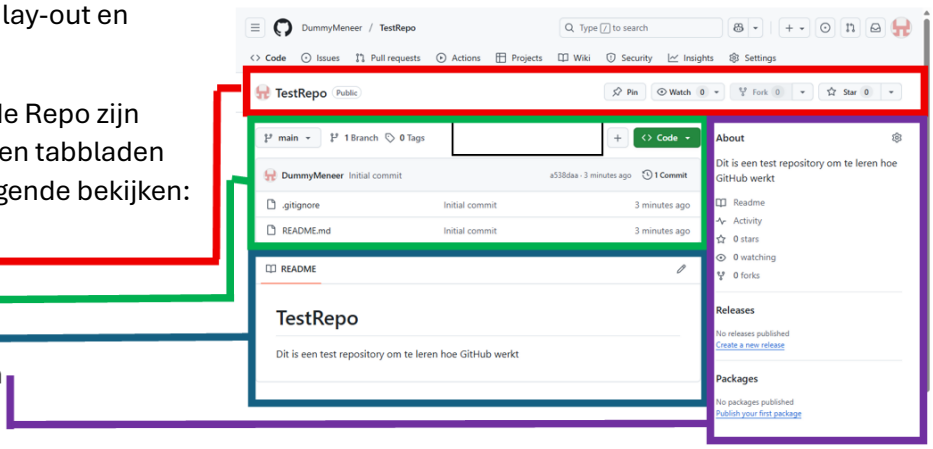


## Opdracht 4: Bekijk de repo pagina (5 -10 minuten)

In deze opdracht wordt de repo pagina voor een deel doorlopen om bekend worden met de lay-out en informatie.

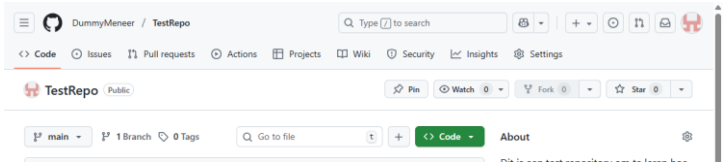
Op de voorpagina van de Repo zijn een aantal onderdelen en tabbladen te zien. We gaan de volgende bekijken:

- Repo kop
- Het code vak
- De ReadMe
- De statistieken



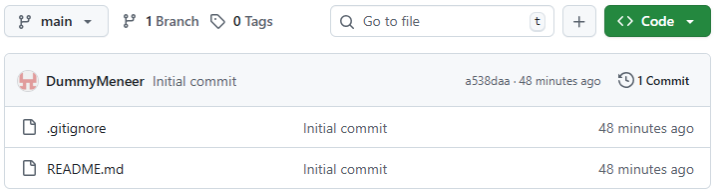

Een groot deel van de andere onderdelen worden in latere hand-outs besproken.

### De Repo Kop



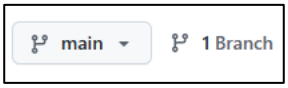

De repo kop bevat van links naar rechts: het icoon van de eigenaar, de naam, de **visibility**, of de gebruiker de repo heeft **gepind**(dit maakt het makkelijker om Repos te zoeken als je veel repos hebt), hoeveel mensen de Repo in de gaten houden (deze mensen krijgen notificaties bij pushes, discussies en issues), de **forks** en als laatste de **stars** (mensen die een repo een star geven, laten waardering blijken en kunnen de repo blijven volgen zonder de vele notificaties die kunnen ontstaan bij de watch functie).

### Het code vak




Het code vak is de kern van de repository, hier komt alle inhoud uiteindelijk te staan. Dit vak kan folders, documenten en assets

Links bovenin staan de **branches**, dit zijn takken waardoor er tegelijk twee of meer versies van code ontwikkeld kunnen worden naast elkaar. Hier krijgen jullie in week 4 meer les over.



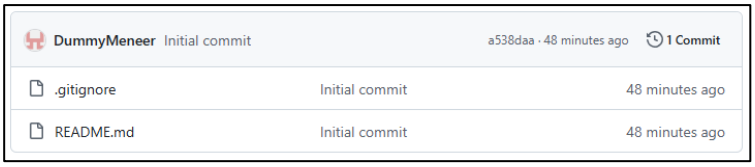

De **tags** extra markeringen die je kan plaatsen bij je je toevoegingen zodat ze makkelijker zijn om te vinden.


De search balk laat je toe om documenten en folders te zoeken, dit maakt het makkelijker om een specifiek document te vinden waar je iets van wil laten zien.


De knop naast de search balk is voor het toevoegen en updaten van de repository, hier gaan we in opdracht 5 verder mee.

De groene code knop laat je toe om de code te downloaden of te **clonen**, dit is vooral belangrijk voor ordening en om samenwerking te bevorderen. Hier meer over in week 2.

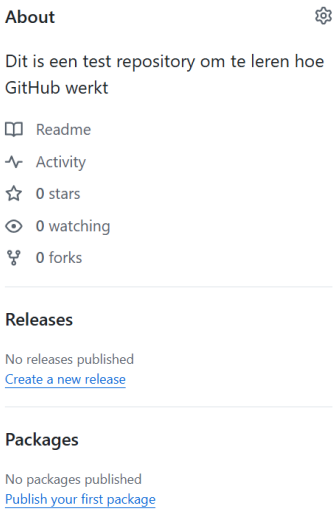
Als laatste hebben we het venster waarin de code staat, dit venster laat de hoeveelheid commits/versies zien, en de stand van de laatste commit/versie op de main branch. Deze box functioneert als een file explorer en laat je toe om door folders en bestanden te zoeken.




**De ReadMe**


Omdat we bij de vorige opdracht een **ReadMe** hebben gemaakt is er een informatie pagina gemaakt die op de voorkant van de repo geplaatst is. Als default wordt de beschrijving in de **ReadMe** geplaatst Je kan deze aanpassen waardoor je meer informatie zoals instructies over het installeren en gebruiken erbij kan plaatsen. Het aanpassen hiervan kan via een nieuwe ReadMe toevoegen die je zelf hebt geschreven of door het pen icoon te gebruiken.

**De statistieken**




De statistieken of **About** deel van de pagina gaat over de details van de repository. De korte beschrijving staat hier samen met een korte link naar de **ReadMe** en een overzicht van hoe actief de Repo is.

Hieronder staan nogmaals de watching en stars. **Forks** zijn hierin een tot nu toe onbekend begrip. Dit zijn de hoeveelheid keren dat mensen de code als basis hebben gebruikt om zelf er verder op uit te breiden.

**Releases** zijn de officiële afgeronde versies die hier geplaatst worden om sneller opgepakt kunnen worden. Hou rekening met het feit dat releases dus moeten functioneren zoals bedoeld. (onbekende bugs mogen maar geen half afgeronde programma's) de **releases** worden gevormd aan de hand van een type tag dat je aan je projectdelen heb gezet.

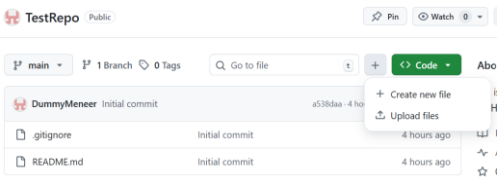
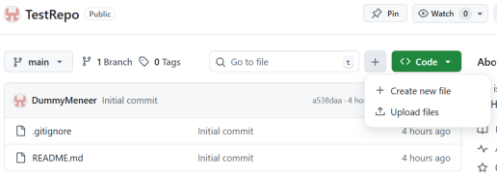
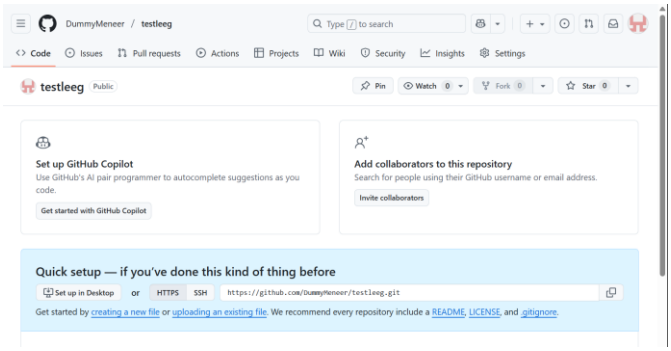
**Packages** zijn vergelijkbaar echter hebben deze vaak meer samenhang met andere elementen zoals een server. De packages zorgen ervoor dat je je dependencies samen hebt met het project.

Zowel **Packages** als **Releases** zullen we niet gebruiken gedurende dit vak maar zijn wel handig om enige kennis van te nemen.

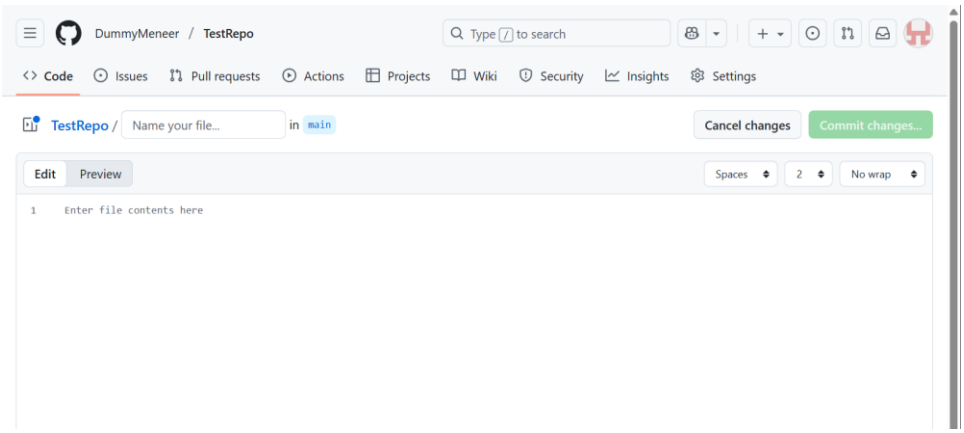
## Opdracht 5: Pushes en commits (20 – 25 minuten)

We hebben nu een repo maar geen inhoud met uitzondering van de gegenereerde bestanden als je deze hebt gegenereerd. In deze opdracht gaan we de inhoud van de test repo aanmaken. Het uploaden van bestanden naar een Repo heet **pushen**, als je een bestand **pusht** naar de repo wordt er gekeken of het bestand al bestaat, zo niet wordt deze toegevoegd, als deze wel bestaat wordt deze bijgewerkt. Iedere **Push** die is doorgezet naar de Repo heet een **Commit**.

Indien je niks hebt gegenereerd zie de rechter kolom voor de vervolgstappen

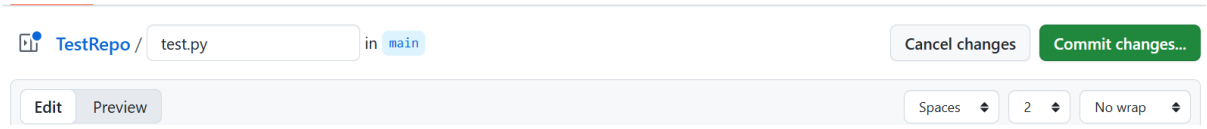
| Met inhoud gegenereerd | Zonder inhoud gegenereerd |
|---|---|
| **Stap 1: klik op [+] of [Add File] knop**<br>  <br><br>Stap 1b: Uit de dropdown kies<br>“+ create new file”<br>  <br> | **Stap 1: in het tekstblok klik op de tekst “Creating a new file” in het blauwe vak met Quick setup.**<br>  <br> |

Dit is het beeld dat zichtbaar is als je een nieuw bestand gaat maken:



**Stap 2: geef je bestand een naam EN een extensie**

Bovenin ga je het bestand een naam geven deze naam mag je zelf kiezen maar zorg ervoor dat je bestand een extensie heeft dit maakt voor projecten heel veel uit aangezien een type loos bestand vaak niet functioneel meer is. Dus geen “Index” of “Main” maar “index.php” of “Main.py”

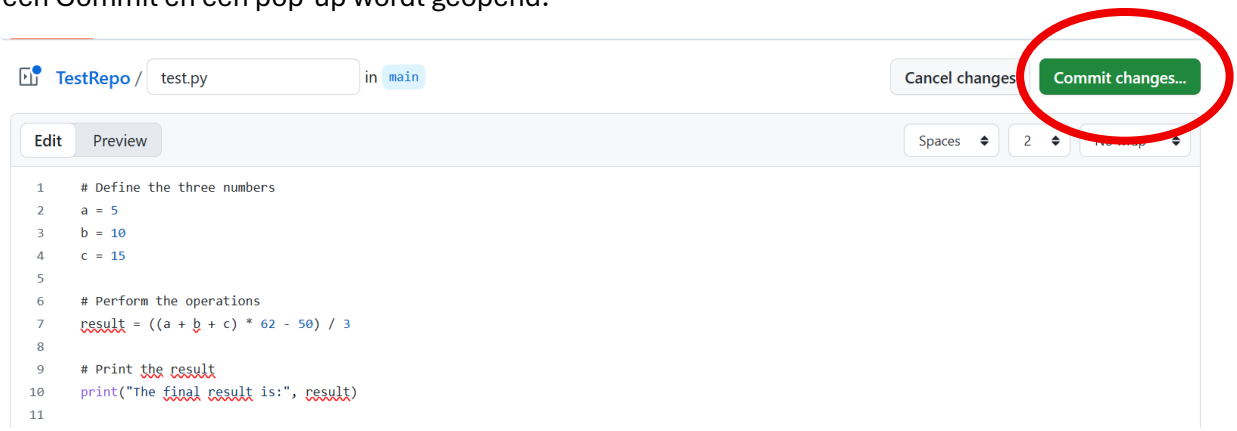



**Stap 3: geef het bestand inhoud en rond het bestand af**

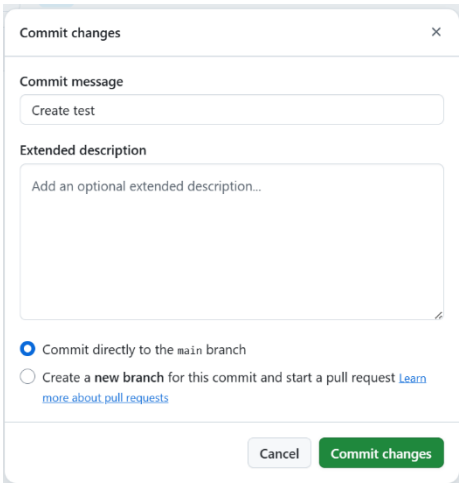
Nadat de titel is gemaakt ga je het bestand vullen. Normaal gesproken zou je of hierin programmeren (Niet aangeraden) of zou je het erin kopiëren. Voor deze opdracht is er een stuk code voor geschreven die gebruikt mag worden:

```
# Define the three numbers
a = 5
b = 10
c = 15

# Perform the operations
result = ((a + b + c) * 62 - 50) / 3

# Print the result
print("The final result is:", result)
```

Nadat de code in het bestand staat kunnen we op de knop commit changes drukken. De changes worden dan verpakt en samengevat in een Commit en een pop-up wordt geopend.




Deze commit moet nog een naam krijgen en een beschrijving. GitHub genereerd op deze wijze van uploaden al een Commit message in de vorm van “Create Bestandnaam”. De beschrijving is leeg gehouden door GitHub en is optioneel. Mochten er nog taken zijn of belangrijke opmerkingen over de file dan kan er hier iets over geschreven worden.



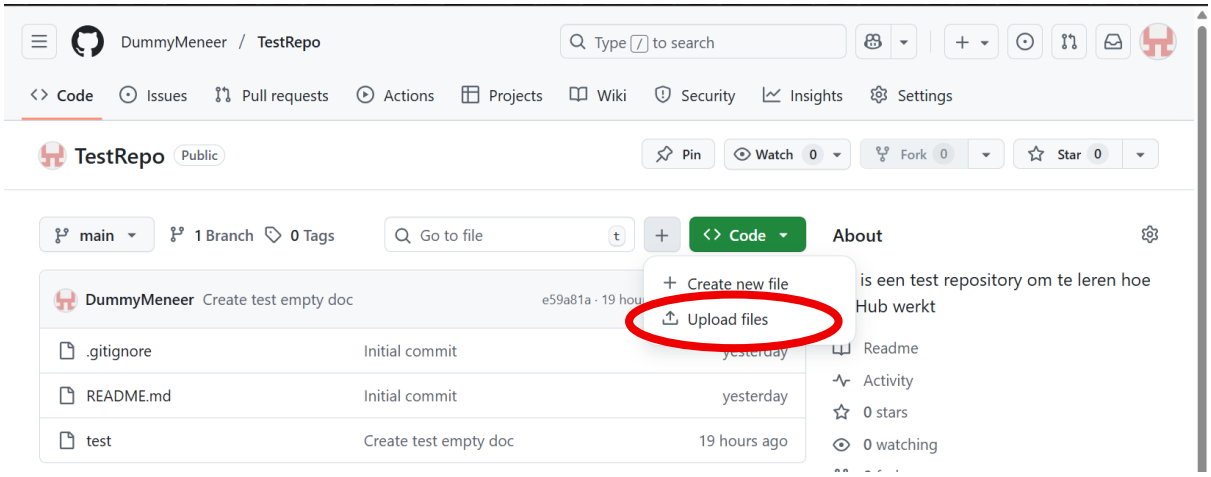

Als laatste is er een keuze die gemaakt kan worden tussen “Commit directly to the main branch” en “create a new branch…”. Voor deze opdrachten blijven we “commit directly to the main branch” gebruiken, tot week 4, dan gaan we met branching werken.

Klik op commit changes en dan word je gebracht naar de hoofdpagina van de repo waar de wijziging direct zichtbaar is.

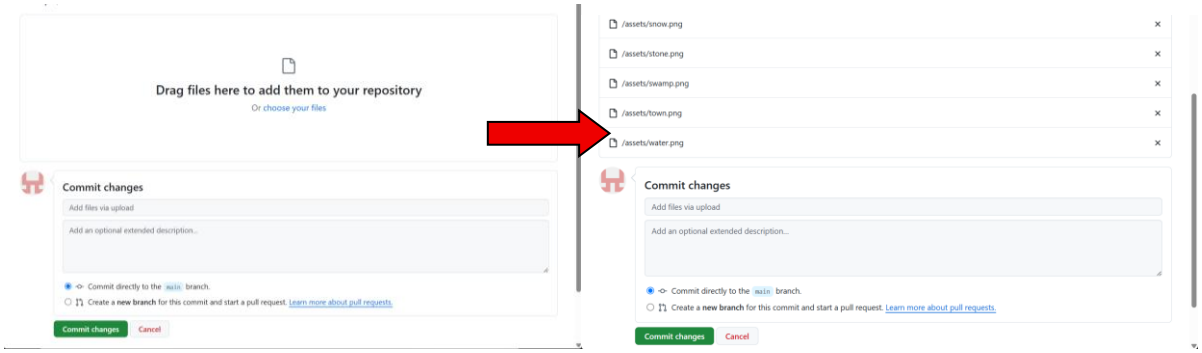
**Stap 4: Folders en bestanden in bulk uploaden**

Voor grote projecten is het zeer onhandig om een commit te doen voor iedere losse file. Het is ook mogelijk meerdere bestanden en folders te pushen naar de repo. Dit doen we als volgt:

Ga weer naar de drop down bij het code blok op de repo hoofdpagina en kies “Upload files”




Hierna krijg je een upload vak te zien waar je de bestanden en folders in kan slepen. Hou er rekening mee dat de bestanden die in de Gitignore vermeld staan vervolgens er uitgefilterd worden zodat de GitHub niet vol komt met onnodige bestanden.

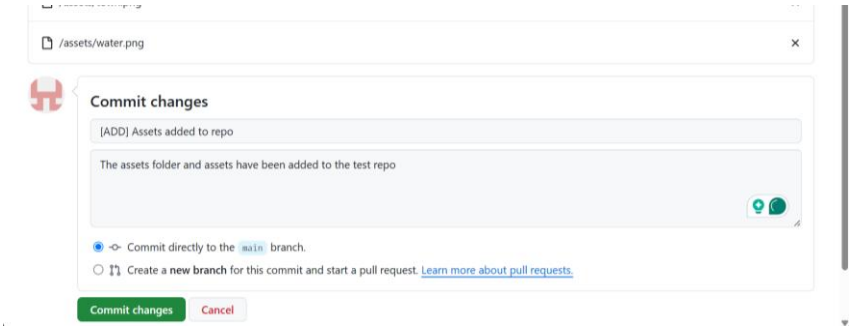



Nadat de files erin staan kan er een naam gegeven worden aan de Commit. Omdat er meerdere bestanden worden gepusht kan er geen commit bericht/naam gegenereerd worden dus moet er een bedacht worden.

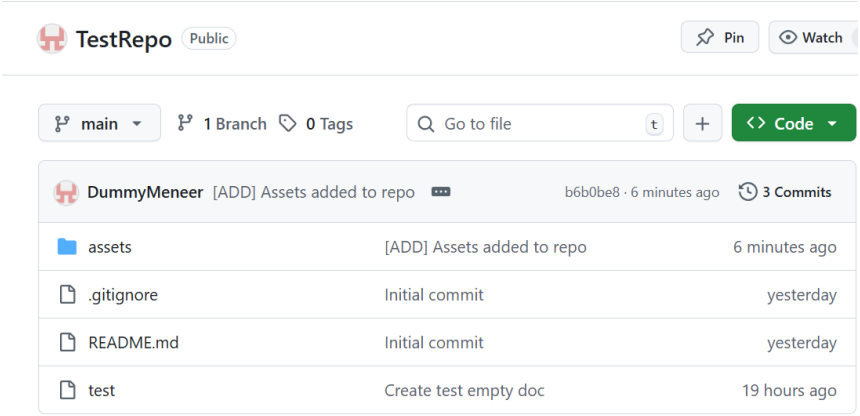
Zorg er voor dat de commit bericht duidelijk omschrijft wat het doet. Create of Add kan gebruikt worden voor het toevoegen terwijl remove of delete gebruikt wordt voor het verwijderen. Commits worden vaak in het Engels geschreven zodat de gehele community en een mogelijk internationaal team gebruik kan maken van jouw commits. Het is ook aangeraden om een uniforme manier van Commits aan te houden, dit maakt het zoeken straks makkelijker. Meer hierover in week 2.

| Suggesties voor commit namen: |
| --- |
| [ADD] Onderwerp, [UPDATE] Onderwerp, [DELETE] onderwerp |
| Create onderwerp, change onderwerp, remove onderwerp |
| Onderwerp made, onderwerp updated, onderwerp deleted |

Hierna is er nog een keuze die gemaakt moet worden voor hoe de commit gestuurd moet worden. Ook voor een groep upload zullen we voor nu op de main branch pushen.




Hierna klikken we op de “commit changes” knop, dit brengt de pagina weer terug naar de hoofdpagina van de repo, hier zien we de folder die is toegevoegd.



## Week 1 inleveren

Een tekst bestand met 2 dingen:

1. De datum en tijd (op halve uren afgerond is prima)
2. De link of naam van je GitHub account

Aan de hand hiervan ga ik kijken of je eerste repo in orde is en geef ik feedback op de repo. Indien je al langer GitHub gebruikt en al repos hebt staan zorg ervoor dat er een publieke Repo is waar ik naar kan kijken en dat de naam en beschrijving hiervan een indicatie is van wat er in zit. Geef dan ook aan in het tekstbestand wat de naam is van de Repo die je wilt dat ik ga bekijken.

## Volgende week

Volgende weke worden de volgende onderwerpen behandeld:

- Clones
- History
- Collaborators
- Rechten
- Naming

## Vooruitblik op jullie eindopdracht voor versiebeheer

Jullie gaan een repo opzetten voor jullie Pygame eindproject. Hier gaan jullie gebruik maken van diverse principes zoals Collaborators, Branches, pulls en cloning. Over de weken gaan jullie deze kennis uitbreiden en toepassen op dit project.

TODO: criteria

## Bonus opdracht

Maak een tweede Repo met daarin het huiswerk van Blok 1 python huiswerk of blok 1 HTML huiswerk. Deel je pushes op in de weken waarin je huiswerk hebt gemaakt, dus: Week 1 Input en print, Week 2 if statements, ...