

# Blok 8 - OOP

*Under construction*

- [Prompt Engineering 4](#)
- [Voorbeeld advanced Prompting](#)
- [Cyber Security 1](#)
- [OOP](#)
- [OOP Challenge](#)
- [OOP Inheritance](#)
- [Kennis Check Blok 8](#)

# Prompt Engineering 4

Deze module is express leeggelaten omdat de ontwikkelen zo snel gaan wordt deze modile later opgeleverd en aangepast aand estand van de techniek op dat moment. Hieronder staan al wel vast losse ideeën.

- ToDo advanced oefeningen
- Dynamic Prompting
- Let AI create prompts
- Let AI ask you questions

Q&A Strategy prompt

..... and ask me some yes/no questions in order to help provide a better recommendation.

what are the different ways to, ..... and give all pro's and con's each suggestion.

Do .... one step at a time and ask me to continue .....

Play the role of a teacher, make complex topics simple.

# Voorbeeld advanced Prompting

## De Tree of Thought (ToT)-methode

(eenvoudige versie)

Ik wil eenvoudig MVC PHP framework maken. Als voorbeeld maak je een controller, een view en een model. De database die je kan gebruiken is bijgevoegd.

**Stap 1: Genereer ten minste drie logische denkstappen die mogelijke richtingen verkennen.**

**stap 2: Evalueer die denkstappen.**

Er worden drie manieren van een aanpak van dit probleem bedacht en elke manier wordt beoordeeld op sterke- en zwakke punten.

<deel van het antwoord weggehaald>

Richting	Sterke punten	Zwakke punten	Beoordeling
A: Kernstructuur	Zorgt voor een goed georganiseerde basis en scheidt verantwoordelijkheden netjes.	Mogelijk over-engineered voor een eerste voorbeeld; vertraagt de “werkende demo”.	<b>Hoog</b> – goed als je het framework later wilt uitbreiden.
B: Voorbeeldgericht	Levert snel zichtbare resultaten; handig voor leren of demonstratie.	Kan leiden tot spaghetti-architectuur als je niet oplet; framework groeit ad hoc.	<b>Midden</b> – goed voor snelle start, minder voor schaalbaarheid.
C: Databasegericht	Helpt bij correcte datamodellering; logisch als data centraal staat.	Makkelijk om te verzanden in datadetails voordat de flow duidelijk is.	<b>Laag-Midden</b> – bruikbaar als datamodellering complex is, anders omslachtig.

# Cyber Security 1

## *0 inhoud*

### **Bescherm je digitale wereld - word een cybersecurity-held!**

#### **Inhoud**

1. **Wat is Cyber Security?**

Cyber Security richt zich op het beschermen van computersystemen tegen misbruik en leert over diverse cyberaanvallen zoals phishing en malware.

2. **HTTPS en netwerkveiligheid**

HTTPS beveiligt de communicatie tussen browser en website door gegevens met een SSL-certificaat te versleutelen, wat meelesen en manipulatie voorkomt.

3. **Encryptie**

Encryptie maakt gegevens onleesbaar voor derden, gebruikmakend van ofwel dezelfde sleutel (symmetrisch) of twee verschillende sleutels (asymmetrisch) voor versleuteling en ontsleuteling.

4. **Hashing**

Hashing is een eenrichtingsversleuteling die gebruikt wordt om wachtwoorden veilig op te slaan, zodat deze niet terug te rekenen zijn naar het origineel.

5. **Brute Force-aanvallen en Loginbeveiliging**

Brute force-aanvallen, waarbij veel wachtwoorden worden geprobeerd, kunnen worden voorkomen door inlogpogingen te limiteren, vertragingen in te bouwen of 2FA toe te passen.

6. **Rainbow tables**

Rainbow tables zijn lijsten van wachtwoorden en hun hashes, gebruikt door aanvallers om snel originele wachtwoorden te achterhalen uit gehashte data.

7. **Salting en encryptie**

Salting voegt een unieke willekeurige tekst ("salt") toe aan een wachtwoord vóór het hashen, wat de veiligheid verhoogt door rainbow tables minder effectief te maken

## 8. [Test je kennis](#)

Heb je alles goed begrepen? Test je kennis en bereid je voor op de kennis-check!

# 1 Wat is Cyber Security?

## ▣▣ Leerdoelen

- Je weet wat Cyber Security is en waarom het belangrijk is.
- Je kent verschillende soorten cyberaanvallen.
- Je kunt voorbeelden noemen van echte incidenten en uitleggen wat er fout ging.

## ▣▣ Uitleg

Cyber Security betekent simpel gezegd: **het beschermen van computersystemen tegen aanvallen of misbruik**. Denk hierbij aan het veilig houden van je website, je wachtwoorden, je e-mails en alle andere digitale gegevens.

Als je een website of app bouwt, ben je automatisch verantwoordelijk voor de veiligheid van de gegevens van je gebruikers. Hackers proberen vaak in te breken via bekende zwakke plekken, zoals slechte wachtwoorden of fouten in je code.

## Voorbeelden van cyberaanvallen:

1. **Phishing:** iemand probeert jou te misleiden om je wachtwoord af te geven (bv. via een nep-mail van je bank)
2. **Adware:** software dat ongevraagd advertenties toont.
3. **Virus:** speciaal soort malware dat zichzelf kan vermenigvuldigen (net als het Corona virus).
4. **DDoS-aanval:** een server wordt overspoeld met aanvragen en raakt onbereikbaar
5. **SQL-injection:** via een formulier wordt je database gehackt
6. **Man-in-the-middle:** iemand onderschept je gegevens tussen jou en een website

▣▣ Je gaat in deze module zien hoe aanvallen werken én hoe je jezelf (en jouw code) daartegen kunt beschermen.

## Opdracht 1, malware

Hierboven staat een lijst van 6 soorten cyberaanvallen. Welke van deze maakt niet per sé gebruik van **kwaadaardige** software?

Zoek informatie op internet en leg uit!

Voeg je **bronvermelding** toe.

## Opdracht 2, risico

Hierboven zijn 6 voorbeelden van cyberaanvallen beschreven. Zoek zelf op wat de volgende soorten cyberaanvallen betekenen:

1. **Ransomware**
2. **Trojan**
3. **Spyware**
4. **Adware**
5. **Keylogger**

Jij bent *IT Security Officer* en jij moet aangeven welke van deze bedreigingen het gevaarlijkst zijn.

Zet deze 6 soorten op volgorde van gevaarlijkheid (volgens jou) en leg bij elk type uit waarom je het op die plek zet.

Voeg je **bronvermelding** toe.

## Opdracht 3, Incident analyseren

1. Zoek online een **bekend cybersecurity-incident** (bijv. een datalek, ransomware-aanval of hack bij een groot bedrijf).
2. Beschrijf in je eigen woorden:
  - Wat er is gebeurd
  - Wat de gevolgen waren

- Wat er beter gedaan had kunnen worden

3. Voeg je *bronvermelding* toe.

## Opdracht 4, Reflectie

- Waarom denk jij dat veel mensen niet nadenken over digitale veiligheid?
- Wat is iets waar jij op gaat letten sinds je deze les hebt gevolgd?

## Inleveren

- Beschrijf alle het antwoord op alle 4 de opdrachten. Neem de vraag over en weer de vraag daarna uit.  
Werk netjes en lever een PDF in.

📄 Om jezelf goed te beschermen tegen 'hacks' kun je heel dingen doen. Eén van de basis zaken is het **versleutelen** van **gegevens**.

📄 In de rest van deze module gaan we vooral hier op in zoomen.

# 2 HTTPS en netwerkveiligheid

## Leerdoelen

- Je begrijpt het verschil tussen HTTP en HTTPS.
- Je weet waarom SSL-certificaten belangrijk zijn voor beveiligde communicatie.
- Je kunt een eigen website beveiligen met HTTPS.

## Uitleg

### HTTP en HTTPS....?

HTTP en HTTPS zijn de 'talen' waarmee je browser met een website praat.

HTTP staat voor **HyperText Transfer Protocol**. Het zorgt ervoor dat je webpagina's kunt opvragen van een server.

HTTPS is hetzelfde, maar dan met een extra beveiligingslaag: de **S** staat voor **Secure**. Bij HTTPS worden de gegevens versleuteld verstuurd, zodat niemand onderweg kan meelezen.

☐ Daarom zie je `http://` of `https://` voor een URL – het geeft aan hoe je browser de website moet benaderen.

## Wat is het verschil tussen HTTP en HTTPS?

HTTP betekent dat gegevens **onversleuteld** worden verzonden. Iedereen die tussen jou en de server in zit (zoals hackers op een openbaar netwerk), kan meekijken.

HTTPS (de S staat voor **Secure**) gebruikt een **SSL-certificaat** om alle communicatie tussen jouw browser en de server te **versleutelen**.

### Wat is versleutelen eigenlijk?

Versleutelen is het omzetten van gegevens zodat ze niet meer zomaar leesbaar zijn.

Vaak heb je een digitale sleutel nodig om deze gegevens weer om te zetten in leesbare informatie.

In de volgende hoofdstukken over encryptie wordt dit uitgelegd.

## Waarom is HTTPS belangrijk?

- Het voorkomt dat anderen je gegevens kunnen "onderscheppen".
- Het zorgt voor vertrouwen bij je bezoekers (slotje in adresbalk).
- Google straft HTTP-sites af in zoekresultaten (door niet op een goede positie te zetten)

### Onterscheppen van gegevens, hoe dan?





Computers op het internet zijn verbonden via netwerken. Deze netwerken kan je vrij eenvoudig aftappen. Je kan dus meelesen met de berichtjes die verstuurd worden over het netwerk. Soms kan je zelfs de berichtjes opvangen, veranderen en doorsturen.

Dit kan op verschillende manieren maar als je toegang hebt tot bepaalde netwerk apparatuur is dat vrij eenvoudig. Dus jouw internet provider zou bijvoorbeeld alles wat jij op het op internet doet kunnen volgen. Als het netwerkverkeer is versleuteld is dat een stuk lastiger.

## Wat is een SSL-certificaat?

Een SSL-certificaat is een soort digitaal paspoort voor je website. Het zorgt ervoor dat bezoekers zeker weten dat ze verbinding hebben met **jouw** site, en dat de data versleuteld is.

### ☐ Waarvoor biedt HTTPS wél bescherming?

HTTPS zorgt ervoor dat de **verbinding tussen jouw computer en een website veilig is**. Dat betekent:

1. **Niemand kan meekijken** met wat jij invult of leest (zoals je wachtwoord of bankgegevens).
2. **Niemand kan de informatie veranderen** terwijl die onderweg is.
3. Je weet zeker dat je met de **echte website** praat (en niet met een nepserver), als het certificaat klopt.

Denkbij https aan een **afgesloten envelop** in plaats van een open briefkaart: anderen kunnen het bericht niet lezen of aanpassen.

### ☐ Waarvoor biedt HTTPS géén bescherming?

HTTPS voorkomt niet alles. Het beschermt je **niet** tegen:

1. **Nepwebsites met een slotje** – criminelen kunnen ook een HTTPS-site maken die er echt uitziet.

2. **Phishing** – als je op een valse link klikt en je wachtwoord daar invult, ben je nog steeds de klos.
3. **Virussen of malware** – als je iets downloadt, controleert HTTPS niet of het veilig is.
4. **Slechte wachtwoorden** – HTTPS helpt niet als jij zelf een zwak of gestolen wachtwoord gebruikt.

## ☐☐ Reflectie

1. Waarom is HTTPS extra belangrijk op loginpagina's? Leg uit wat het risico is.
2. Wat zou er kunnen gebeuren als je een onbeveiligde (HTTP) verbinding gebruikt in een openbaar WiFi netwerk? Leg uit wat het risico is.
3. Je gaat naar [www.nu.nl](http://www.nu.nl) (voor het nieuws) en stel dat je geen slotje ziet en er dus geen SSL verbinding is gemaakt. Wat kan er mis gaan? Wat is het risico?
4. Je meldt je aan voor een sport evenement op een onbeveiligde (HTTP) site. Leg uit wat het risico is.
5. Je hebt een website op je laptop draaien en een medestudent bezoekt deze website via een onbeveiligde (HTTP) verbinding. Leg uit wat het risico is.
6. Zet alle risico's (nummer 1 t/m 5) op volgorde van gevaarlijkheid. Motiveer je antwoord.

De reflectie is een verslag in je **eigen woorden**, AI input wordt niet geaccepteerd!

Deze vragen komen terug in de kennis-check!

## ☐☐ Inleveren

- Lever je reflectie in als een .pdf

# 3 *Encryptie*

## ☐☐ Leerdoelen

- Je weet wat encryptie betekent en waarvoor ze gebruikt worden.
- Je kunt een eenvoudige versleuteling (Caesar cipher) herkennen en kraken.
- Je weet het verschil tussen symmetrische en asymmetrische encryptie.

## 📺 Video

<https://www.youtube.com/embed/r4HQ8Bp-pfw>

## 📖 Uitleg

### Wat is encryptie?

**Encryptie** (versleuteling) betekent dat je gegevens onleesbaar maakt voor anderen. Alleen iemand met de juiste 'sleutel' kan de gegevens weer ontcijferen.

- **Symmetrisch:** dezelfde sleutel voor versleutelen en ontsleutelen (bijv. Caesar cipher).
- **Asymmetrisch:** verschillende sleutels voor versleutelen en ontsleutelen (zoals bij HTTPS).

## 📖 Verdieping over symmetrische en asymmetrische encryptie

### Leg uit, hoe werkt dat asymmetrisch precies?

Stel iedereen heeft een eigen brievenbus met **twee sloten met twee sleutels**: een **public key** waarmee je een brief in de brievenbus kan stoppen en een **private key** waarmee de eigenaar de brievenbus kan openen en de inhoud kan bekijken.

Image.png found or type unknown

"Wil je mij iets sturen?"

Gebruik dan deze **publieke sleutel** (= mijn slotje)."

**Alleen jij** kunt de brievenbus openen,  
want jij hebt de **privé-sleutel** die bij dat slotje hoort.

## Waarom zou asymmetrische encryptie beter zijn dan symmetrische encryptie?

### ☐ Symmetrische encryptie

- **Sleutel is geheim en wordt gedeeld**
- Je gebruikt **dezelfde sleutel** om iets te versleutelen én te ontsleutelen.
- Probleem: je moet die **geheime sleutel veilig delen**, en dat is lastig.

### ☐ Asymmetrische encryptie

- Gebruikt **twee sleutels**:
  - Een **publieke sleutel** (om te versleutelen)
  - Een **privé sleutel** (om te ontsleutelen)
- Je kunt de **publieke sleutel vrij geven aan iedereen**, want alleen jij kunt de boodschap met je **privé sleutel** openen.

### ☐ Waarom is asymmetrische encryptie beter?

- De **privé sleutel** hoef je nooit met iemand te delen en de kans dat deze prive sleutel in verkeerde handen komt is dan dus kleiner.

## Waarom zou je symmetrische encryptie nog gebruiken als asymmetrische encryptie beter is?

### ☐ Kort gezegd:

- **Asymmetrisch** = veilig, wat meer complex en trager.
- **Symmetrisch** = iets minder veilig, maar wel sneller en eenvoudiger

Om dat symmetrische encryptie sneller en eenvoudiger is wordt die vooral gebruikt daar waar de sleutel snel veranderd dan is het niet zo erg als de sleutel in verkeerde handen komt omdat er dan weer een nieuwe sleutel wordt gebruikt.

## ☐ Opdracht - Caesar Encryptie

In deze opdracht wordt jouw gevraagd een wachtwoord te 'hacken'. Het wachtwoord dat je moet ontcijferen is encrypted met het **Caesar algoritme**. Om te kunnen hacken moet je eerst begrijpen hoe het Caesar encryptie algoritme werkt.

## Code Caesar encryptie

```
<?php

$sleutel = 2; // Caesar-sleutel: aantal posities verschuiven

function caesar_encrypt($tekst, $verschuiving) {
    $resultaat = "";

    for ($i = 0; $i < strlen($tekst); $i++) {
        $char = $tekst[$i];

        if (ctype_alpha($char)) {
            $offset = ctype_upper($char) ? ord('A') : ord('a');
            $resultaat .= chr(((ord($char) - $offset + $verschuiving) % 26) + $offset);
        } else {
            $resultaat .= $char; // Laat leestekens, spaties en cijfers ongemoeid
        }
    }

    return $resultaat;
}

function caesar_decrypt($tekst, $verschuiving) {
    return caesar_encrypt($tekst, 26 - $verschuiving); // Omgekeerde verschuiving
}

// Voorbeeld
$origineel = "Hallo Wereld!";
$gecodeerd = caesar_encrypt($origineel, $sleutel);
$ontsleuteld = caesar_decrypt($gecodeerd, $sleutel);

echo "Origineel: $origineel\n";
echo "Versleuteld: $gecodeerd\n";
echo "Ontsleuteld: $ontsleuteld\n";
```

Bekijk deze encryptie methode en probeer te begrijpen wat er gebeurt. Je kunt natuurlijk de code ook proberen/testen.

Als je het algoritme begrijpt, kan je verder.

## Hack-opdracht

Het volgende wachtwoord is encrypted met de Caesar methode.

**Chs\_hr\_fdgdhl!**

Kan jij dit wachtwoord kraken?

## Reflectie

- Is het Caesar algoritme een vorm van symmetrische- of asymetrische encryptie? Leg uit waarom.
- Hoe zou jij aan een niet programmeur uitleggen wat Caesar encryptie is?
- Hoe heb je het wachtwoord gehacked? Welke stappen heb je doorlopen?
- Als je een andere en misschien een iets lastigere wachtwoord moest kraken dat met het Caesar algoritme is ge-encrypt, hoe zou je dat dan aanpakken?

## Inleveren

1. Antwoorden op de reflectie PDF

# 4 Hashing

## Leerdoelen

- Je weet hashing betekenen en waarvoor ze gebruikt worden.
- Je kunt wachtwoorden op een veilige manier hashen in PHP.
- Je weet hoe je hashed wachtwoorden zou kunnen hacken.

## ❑ Wist je dat ....

**Blockchain** en **crypto** gebouwd is op de principes van hashing?

Wil je weten hoe, kijk dan naar deze video; het is niet erg als je het gedeelte over block-chain niet helemaal begrijpt.

<https://www.youtube.com/embed/2BldESGZKB8>

## ❑ Uitleg, wat is hashing?

Hashing is een soort encryptie die maar één kant op werkt; Je kunt wel encrypten maar niet decrypten!

Hashing is dus **eenrichtingsversleuteling**: je zet een waarde om in een versleutelde vorm die je **niet meer kunt terugrekenen** naar het origineel. Dit wordt vaak gebruikt voor het **veilig opslaan van wachtwoorden**.

In PHP kun je een wachtwoord hashen met:

```
$hash = hash("sha256", "geheim123");
```

Het resultaat is bijvoorbeeld:

```
$2y$10$f0849a42909dc18035cb470d239e485acbc1cdd1e48bc41f7a2801e3b08bdbdb
```

Je kunt dit **niet** terug rekenen naar `geheim123`

## ❑ Wat gebeurt er bij het inloggen?

Bij het inloggen controleer je of het ingevoerde wachtwoord overeenkomt met de opgeslagen hash:

```
if hash("sha256", $ingevoerdWachtwoord) == $hash {  
    echo "Ingelogd!";  
}
```

“ △ Je moet het zo zien dat het ingevoerde wachtwoord opnieuw wordt omgezet met de hash-functie. Is het resultaat hetzelfde als de opgeslagen \$hash dan is het ingevoerde wachtwoord goed.

# Opdracht 2 Hashing

Neem om te beginnen de start code, `form.htm` en `check.php` over.

Controleer of het werkt en probeer in te loggen met admin/wachtwoord. Probeer de code te begrijpen.

## form.html

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Loginformulier</title>
</head>
<body>
  <h2>Login</h2>
  <form method="get" action="check.php">
    <label for="gebruikersnaam">Gebruikersnaam:</label><br>
    <input type="text" name="gebruikersnaam" id="gebruikersnaam" required><br><br>

    <label for="wachtwoord">Wachtwoord:</label><br>
    <input type="password" name="wachtwoord" id="wachtwoord" required><br><br>

    <button type="submit">Inloggen</button>
  </form>

  <p><?= $melding ?></p>
</body>
</html>
```

## check.php

```
<?php
$gebruikers = [
  'max' => '',
  'lisa' => '',
  'admin' => 'e0NRta6G8WpOMOMK9qOC6O3z6F7cJcmA0r8GRPt4NeWcAUO4ED8Di' // Vaste hash van
"wachtwoord"
];
```



```

$melding = '';

$gebruikersnaam = $_GET['gebruikersnaam'];
$wachtwoord = $_GET['wachtwoord'];

if (isset($gebruikers[$gebruikersnaam])) {
    if (password_verify($wachtwoord, $gebruikers[$gebruikersnaam])) {
        $melding = "✓ Ingelogd als <strong>$gebruikersnaam</strong>!";
    } else {
        $melding = "✗ Fout wachtwoord.";
    }
} else {
    $melding = "✗ Gebruiker bestaat niet.";
}

?>

```

Als je de code begrijpt dan is de volgende opdracht niet moeilijk.

⚠ Zorg ervoor dat de gebruikers **max** en **lisa** kunnen inloggen met de volgende wachtwoorden.

Gebruikersnaam	Wachtwoord
max	top-secret123!
lisa	sinclair1974

Probeer te ontdekken hoe je dit moet doen en hoe je de juiste wachtwoorden toe kan voegen.

## Hack-opdracht

Ste je hebt de volgende code onderschept;

```

$gebruikers = [
    'max' => '$2y$10$AF4UcvDki/VEQKCUzHQxludD9cYLRaF9v4GIkhTyTzWzxCN/Yo0q6',
    'lisa' => '$2y$10$nWGSzTcP7TShCVwz78eNGO3LjoxR/FPR3WjZqxbodWHQUe/XEzZC.',
    'admin' => '$2y$10$e0NRta6G8WpOMOMK9qOC6O3z6F7cjcmA0r8GRPt4NeWcAUO4ED8Di'
];

```

Aan jouw de taak om de wachtwoorden te 'kraken'.

### Tip 1

Tip Het is niet mogelijk om de hash terug te rekenen vanuit de hash naar een wachtwoord.

### Tip 2

Tip 2 Je kunt natuurlijk wel een wachtwoorden in een hash omzetten en kijken of de hash hetzelfde is.

### Tip 3

Lisa en Max hebben wachtwoorden die veel gebruikt worden en niet erg veilig zijn.

Had je **geen tips** nodig, dan ben je een **meester hacker**!

## ☐☐ Reflectie

- Waarom is het **geen** goed idee om wachtwoorden encrypted (versleuteld) op te slaan en waarom kun je beter hashing gebruiken?  
Leg uit wat er mis zou kunnen gaan als je encryptie gebruikt voor het opslaan van wachtwoorden.
- Stel je hebt een bestand gekregen met allemaal userid's en wachtwoorden. De wachtwoorden zijn hashed. Leg uit hoe je mogelijk toch achter de wachtwoorden kan komen.

## ☐☐ Inleveren

1. aangepaste `check.php`
2. de antwoorden op de vragen uit de reflectie in pdf

# 5 Brute Force-aanvallen en Loginbeveiliging

# ☐☐ Leerdoelen

- Je weet wat een brute force-aanval is.
- Je kunt een eenvoudige loginpagina maken in PHP.
- Je begrijpt hoe je zo'n loginpagina kunt beveiligen tegen brute force-aanvallen.

# ☐☐ Uitleg

## Wat is een brute force-aanval?

Bij een brute force-aanval probeert een aanvaller heel veel verschillende wachtwoorden achter elkaar uit om zo toegang te krijgen tot een account. Als er geen beperking zit op het aantal pogingen, kan dit op den duur succes hebben.

## Hoe bescherm je hiertegen?

Je kunt brute force-aanvallen voorkomen door bijvoorbeeld:

- Een limiet te stellen op het aantal pogingen
- Een vertraging inbouwen bij het inloggen en deze laten groeien als je vaker een fout wachtwoord hebt ingevuld.
- Tijdelijk een gebruiker of IP-adres te blokkeren (=black listing)
- Van te voren alleen bepaalde ip adressen toelaten (=white listing)
- Captcha toe te voegen
- Twee-factor authenticatie toe te passen
- Zet alle mislukte inlogpogingen in een logbestand.

Deze laatste bescherming met logfile voorkomt niet zozeer dat er brute force aanvallen plaatsvinden, maar je kunt wel zien als er wat vreemds gebeurt. Je kan dan op dat moment (extra) maatregelen nemen.

## Wachtwoord lengte

Een wachtwoord bestaat over het algemeen uit hoofdletter, gewone letters, cijfers en een aantal speciale karakters. In totaal zijn dat ongeveer 70 tekens. Dat betekent dat je 70 verschillende wachtwoorden kan maken, bij een wachtwoord lengte van 2 is dit 4 900 verschillende wachtwoorden. Dit aantal loopt snel op:

wachtwoord lengte	aantal mogelijkheden
1	70
2	4 900
3	343 000
4	24 010 000
5	1 680 700 000
6	117 649 000 000

Dit lijkt heel veel maar met één snelle computer met een snelle GPU kan je 10 miljard wachtwoorden per seconden testen. Een wachtwoord van 6 posities heb je dan dus binnen ongeveer 10 seconden gekraakt.

### Hoe lang moet mijn wachtwoord dan zijn, is 10 posities voldoende?

#### Jouw wachtwoord veiligheid hangt af van:

- de gebruikte encypty of hashing (bijvoorbeeld SHA256)
- welke hardware kan je gebruiken om te hacken (via cloudcomputing kan je 1000-den snelle GPU's inzetten).

#### Voorbeeld:

- Je gebruikt de wat verouderde **SHA256** hashing.
- Je gebruikt **10** hele snelle GPU's (NVidia **RTX4080**)

Rekentijd ongeveer **16 dagen**

☐ Bedrijven als **OpenAI, Microsoft, Amazon** of **Google** hebben zoveel rekenkracht dat ze jouw wachtwoord van 10 tekens binnen 20 seconden met brute force *zouden kunnen* kraken.

Ga je naar een wachtwoord van **12 tekens**, dan loopt dit zelfs voor deze bedrijven al snel op naar een **2 tot 3 jaar**.

## ☐ Opdracht

# Maak een eenvoudige loginpagina

Maak een bestand `login.php` met het volgende formulier:

```
<form method="get" action="inlogcontrole.php">
  Gebruikersnaam: <input name="username"><br>
  Wachtwoord: <input type="password" name="password"><br>
  <input type="submit" value="Login">
</form>
```

Maak daarna een eenvoudige `inlogcontrole.php` in PHP:

```
<?php
session_start();

$gebruikersnaam = "admin";
$wachtwoord = "geheim";

if (!isset($_SESSION['pogingen'])) {
    $_SESSION['pogingen'] = 0;
}

if ($_SESSION['pogingen'] >= 3) {
    die("Te veel pogingen. Probeer het later opnieuw.");
}

if ($_GET) {
    if ($_GET['username'] === $gebruikersnaam && $_GET['password'] === $wachtwoord) {
        echo "Welkom!";
        $_SESSION['pogingen'] = 0;
    } else {
        echo "Foutieve inlog.";
        $_SESSION['pogingen']++;
    }
}

?>
```

In het form wordt `$_GET` gebruikt, is dat handig?

Vanuit het oogpunt van cyber security is het beter om `$_POST` te gebruiken. Weet je nog waarom?

## Aanpassingen

- Verander het formulier en de vervolgpagina zodat je geen post meer gebruikt en zodat het user id en wachtwoord niet meer in de URL staan.
- Voeg een vertraging toe met `sleep(1)` bij een mislukte poging.
- Voeg logging toe aan een tekstbestand zodat je pogingen kunt terugzien.

## Extra uitdaging

- Als je bijhoud hoe vaak een gebruiker probeert in te loggen (met een sessie variabele), dan kan je de tijd tussen twee pogingen laten groeien: de eerste keer moet je 1 seconden wachten, dan 2 dan 4 dan 8 dan 16, etc. etc.

Kan jij dat implementeren?

## ☐☐ Reflectie

- Waarom is het belangrijk om een limiet te stellen op het aantal inlogpogingen?
- Wat gebeurt er als je de sessiegegevens wist of in een andere browser werkt?
- Welke nadelen heeft deze eenvoudige beveiliging?

## ☐☐ Inleveren

- De aangepaste code `inlogcontrole.php`
- Reflectie met antwoorden op bovenstaande vragen in pdf.

# 6 *Rainbow tables*

## ☐☐ Leerdoelen

- Je begrijpt wat een rainbow table is en waarom het gevaarlijk is bij wachtwoordbeveiliging.
- Je kunt handmatig een wachtwoord opzoeken in een rainbow table.

- Je snapt waarom salting rainbow tables onbruikbaar maakt.

Menses gebruiken vaak 'standaard' wachtwoorden.

### Wat denk jij dat het meest gebruikte wachtwoord is?

Er zijn [sites](#) waarom de meest gebruikte wachtwoorden staan.

☐ Ga naar de site van [Nordpass](#) en controleer of jouw antwoord klopt.

## ☐ Uitleg: Wat is een rainbow table?

Een **rainbow table** is een lijst van veelgebruikte wachtwoorden met hun bijbehorende hashes. Aanvallers gebruiken dit om snel te achterhalen welk wachtwoord hoort bij een bepaalde hash.

Stel: iemand vindt een lijst met gebruikersnamen en gehashte wachtwoorden. Als jouw wachtwoord in zo'n rainbow table staat, kan die hash direct worden terugvertaald naar het originele wachtwoord – zonder te hoeven raden.

## Waarom werken rainbow tables?

- Hash-algoritmes zoals SHA256 geven altijd dezelfde output voor dezelfde input.
- Veel mensen gebruiken zwakke, veelvoorkomende wachtwoorden.

## Starters code

Deze code vraagt om een wachtwoord te resetten. Het wachtwoord wordt in JSON bestand opgeslagen. Controleer of de code werkt en controleer of het wachtwoord wordt opgeslagen in `gebruikers.json`.

```
<?php
// Als het formulier is verzonden
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $userid = $_POST['userid'] ?? '';
    $wachtwoord = $_POST['wachtwoord'] ?? '';

    // Hash het wachtwoord met SHA-256
    $hashed = hash('sha256', $wachtwoord);
```

```

// Bestandsnaam
$bestand = 'gebruikers.json';

// Bestaat het JSON-bestand al?
if (file_exists($bestand)) {
    $data = json_decode(file_get_contents($bestand), true);
} else {
    $data = [];
}

// Sla of update de gebruiker op
$data[$userid] = $hashed;

// Schrijf naar bestand (mooi opgemaakt voor leesbaarheid)
file_put_contents($bestand, json_encode($data, JSON_PRETTY_PRINT));

echo "<p>Wachtwoord succesvol opgeslagen voor gebruiker <strong>$userid</strong>.</p>";
echo "<p>SHA256 hash: <code>$hashed</code></p>";
exit;
}
?>

```

```

<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <title>Wachtwoord resetten</title>
</head>
<body>
    <h2>Reset je wachtwoord</h2>
    <form method="post">
        <label for="userid">Gebruikers-ID:</label><br>
        <input type="text" id="userid" name="userid" required><br><br>

        <label for="wachtwoord">Nieuw wachtwoord:</label><br>
        <input type="password" id="wachtwoord" name="wachtwoord" required><br><br>

        <button type="submit">Opslaan in JSON</button>
    </form>

```



</body>

</html>

# Hack-opdracht

## 200 meest voorkomende wachtwoorden (internationaal)

123456  
123456789  
12345  
qwerty  
password  
12345678  
111111  
123123  
1234567890  
1234567  
qwerty123  
000000  
1q2w3e  
aa12345678  
abc123  
password1  
1234  
qwertyuiop  
123321  
password123  
1qaz2wsx  
iloveyou  
admin  
qazwsx  
123qwe  
123abc  
654321  
666666  
superman  
112233  
1q2w3e4r  
asdfghjkl  
zxcvbnm  
121212  
1qazxsw2  
letmein

trustno1  
hello  
888888  
football  
monkey  
!@#\$\$%^&\*  
charlie  
batman  
696969  
hottie  
flower  
loveme  
donald  
login  
pokemon  
starwars  
jordan  
dragon  
michael  
shadow  
master  
killer  
maggie  
bite me  
qwerty1  
freedom  
whatever  
cheese  
pepper  
princess  
jennifer  
michelle  
tigger  
hunter  
sunshine  
ashley  
michael1  
lovely  
qwe123  
777777  
ginger  
cookie  
welcome  
taylor

summer  
soccer  
test  
asdf  
internet  
google  
qweasd  
merlin  
mustang  
baseball  
hannah  
snoopy  
thomas  
mypass  
computer  
daniel  
jessica  
pepper123  
pass  
6969  
1111  
999999  
88888888  
444444  
222222  
555555  
333333  
7777777  
0000  
1212  
1004  
2000  
abcd1234  
1989  
1987  
1979  
1978  
q1w2e3r4  
zxcvbn  
a1b2c3  
azerty  
passw0rd  
123123123  
147258

1g2w3e4r  
1a2b3c4d  
2580  
qazxsw  
987654321  
password!  
hunter2  
11111111  
131313  
159753  
a123456  
abc123456  
myspace1  
9999  
147147  
aaaaaa  
zaq12wsx  
q1w2e3  
8888  
iloveyou1  
killer123  
qwe123456  
123456a  
abcdef  
asdasd  
poop  
zxcvbn123  
dragon123  
trustno1!  
abcd  
12344321  
password01  
987654  
test123  
cool  
123123a  
internet123  
letmein123  
master123  
welcome1  
football1  
qwerty  
batman123  
super123

```
77777777
5555
ninja
tinkle
red123
star123
hello123
pass123
admin123
1password
love123
66666666
mypass123
superman123
samsung
qwerty12
asdfgh
admin1
loveyou
pokemon123
iloveu
justin
asdf1234
hottie123
shadow123
hannah123
sunshine1
admin@123
```

1. Vraag een mede-student een wachtwoord uit de lijst van 200 meest voorkomende wachtwoorden te kiezen (lijst staat hier boven).
2. Vraag de student dit wachtwoord in te vullen op jouw Lappie met de "starters code" die je net hebt getest.
3. Laat de mede-student zien dat je het JSON bestand kan lezen en dat je dus zijn hashed wachtwoord kan leven.
4. Hoe kom je er nu achter welke wachtwoord jouw mede-student heeft gebruikt?
5. Precies: je maakt code die alle 200 wachtwoorden hashed en je vergelijkt de gevonden hashes met de hash van je mede student.

## Opdracht: Vind het wachtwoord

Maak een "mini-rainbow table" van de 200 meest voorkomende wachtwoorden en bepaal welk wachtwoord jouw mede-student had ingevoerd.

## Have I been Powned

Op <https://haveibeenpwned.com/Passwords> staat een hele groot bestand met gehashte wachtwoorden. Deze lijst is ook bekend bij hackers en als die een hashed wachtwoord hebben kunnen ze deze opzoeken in deze database en jouw wachtwoord achterhalen.

De lijst bevat alle hashed wachtwoorden die ooit een keer zijn 'gevonden', bijvoorbeeld in een database die is gehacked.

## ☐☐ Reflectie

- Wat maakt rainbow tables zo gevaarlijk?
- Wat zou er gebeuren als we bij elk wachtwoord een andere salt toevoegen?
- Hoe helpt salting om rainbow tables tegen te gaan?

## ☐☐ Inleveren

- Een kort verslag met de gevonden wachtwoorden.
- Een antwoord op de reflectievragen in PDF.

# 7 *Salting en encryptie*

## ☐☐ Leerdoelen

- Je begrijpt wat een **salt** is en waarom het wordt gebruikt bij hashing van wachtwoorden.
- Je kunt zelf code schrijven om te demonstreren hoe salting werkt.
- Je kunt uitleggen hoe je een wachtwoord controleert dat is gehasht met een salt.

## !?!? Waarom Salt

Salting is een beveiligingstechniek die je kan toepassen bij het opslaan van wachtwoorden, omdat het je systeem **veel** beter beschermt tegen aanvallen zoals **rainbow table-aanvallen**.

## Uitleg: Wat is een salt?

Een **salt** is een extra stukje tekst dat je aan een wachtwoord toevoegt voordat je het versleutelt (hasht). Daardoor ziet het wachtwoord er elke keer anders uit, zelfs als iemand hetzelfde wachtwoord gebruikt.

Stel: zonder salt geeft het wachtwoord `geheim123` altijd dezelfde hash. Met een salt (bijv. `!x8Zp#`) wordt het `!x8Zp#geheim123` en krijg je een andere hash.

In dit voorbeeld gebruiken we één vaste salt voor alle wachtwoorden, om het idee simpel te houden.

## Waarom is salting belangrijk?

- Het maakt het moeilijker om gehashte wachtwoorden terug te vinden met voorgeprogrammeerde lijsten (rainbow tables).
- Het maakt brute force-aanvallen minder effectief.

### Waarom heet het “salt”?

#### Stel je voor

Je hebt een bord rijst (= het wachtwoord).

Als iedereen precies dezelfde rijst krijgt, smaakt het bij iedereen hetzelfde.

Maar als jij er wat zout aan toevoegt, wordt het uniek.

Hetzelfde gebeurt met wachtwoorden:

- Zonder “salt” = iedereen met hetzelfde wachtwoord krijgt **dezelfde hash**.
- Met “salt” = elk wachtwoord krijgt **een eigen smaak** (een unieke hash), zelfs als het wachtwoord hetzelfde is.

#### In de techniek

Het “**zout**” is een willekeurig extra stukje tekst dat je toevoegt aan het wachtwoord voordat je het versleutelt of “hasht”.

```
$salt = "x!9f3L";  
$wachtwoord = "123456";  
$hash = hash('sha256', $salt . $wachtwoord);
```

## Opdracht: Hashen met één vaste salt

Schrijf een klein PHP-script dat laat zien hoe salting werkt.

```
$wachtwoord = "geheim123";  
$salt = "ROCAmstelland"; // vaste salt voor alle wachtwoorden  
  
$hash = hash("sha256", $salt . $wachtwoord);  
  
echo "Wachtwoord: $wachtwoord\n";  
echo "Hash: $hash\n";
```

## Controle bij inloggen

Als iemand probeert in te loggen, moet je het ingevoerde wachtwoord opnieuw hashen met dezelfde salt en vergelijken met de opgeslagen hash.

```
$ingevoerd = "geheim123";  
$bekende_hash = .....; // aanvullen  
$salt = "ROCAmstelland";  
  
// vul deze regel code aan zodat de hash wordt gecontroleerd.  
$controle_hash = .....  
  
if ($controle_hash === $bekende_hash) {  
    echo "Ingelogd";  
} else {  
    echo "Fout wachtwoord";  
}
```

1. Bereken met de eerste code de hash van het wachtwoord geheim123
2. Plaats de hash dan in de 2de code (\$bekende\_hash) en vul regel 6 aan zodat de hash wordt gecontroleerd.



## ☐☐ Reflectie

- Waarom is het veiliger om een salt toe te voegen?
- Wat zou nog veiliger zijn dan één vaste salt voor iedereen?
- Stel je wil hacken met behulp van Rainbow tables en je weet welke salt er is gebruikt, wat moet je dan doen?
- In de (laatste) opgave maken we gebruik van een vaste salt waarde. Denk je dat je de beveiliging kan verbeteren door een ander salt waarde te kiezen? Zo ja welke en waarom, leg uit in eigen woorden.

## ☐☐ Inleveren

1. Je PHP-script waarin je laat zien hoe je een wachtwoord hasht met een salt en controleert bij inloggen.
2. Een PDF met de antwoorden op de reflectievragen.

## ☐☐ *Test je kennis*

### Hoofdstuk 1: Wat is Cyber Security?

1. Wat is de primaire focus van Cyber Security?
  - a) Het beschermen van websites tegen computer-bugs
  - b) Het beschermen van computersystemen tegen aanvallen en misbruik.
  - c) Het ontwikkelen van nieuwe softwareprogramma's.
  - d) Het beheren van netwerkinfrastructuur.

#### ☐ Antwoord

b) Het beschermen van computersystemen tegen aanvallen en misbruik.

2. Welke van de volgende cyberaanvallen omvat het misleiden van iemand om wachtwoorden af te geven, vaak via nep-e-mails?

- a) Malware
- b) DDoS-aanval
- c) Phishing
- d) SQL-injection

**Antwoord**

c) Phishing

Welke term wordt gebruikt voor software die ontworpen is om ongevraagd advertenties te tonen en je naar bepaalde webwinkels stuurt?

- a) Adware
- b) Spam
- c) Malware
- d) Commerceware

**Antwoord**

c) Adware

Welke vorm van malware kan zichzelf verspreiden naar andere bestanden of programma's zodra het op een computer is geïnstalleerd?

- a) Worm
- b) Spyware
- c) Virus
- d) Adware

**Antwoord**

c) Virus

## Hoofdstuk 2: HTTPS en netwerkveiligheid

3. Wat is het belangrijkste verschil tussen HTTP en HTTPS?

- a) HTTPS is alleen voor professionele websites, HTTP voor persoonlijke.
- b) HTTPS staat voor HyperText Transfer Protocol Secure en versleutelt de communicatie tussen browser en server.
- c) HTTP is sneller dan HTTPS.
- d) HTTP gebruikt een SSL-certificaat, HTTPS niet.

**Antwoord**

b) HTTPS staat voor HyperText Transfer Protocol Secure en versleutelt de communicatie tussen browser en server.

4. Waarvoor biedt HTTPS geen bescherming?

- a) Het onderscheppen van ingevulde wachtwoorden door derden.
- b) Het veranderen van informatie terwijl deze onderweg is.
- c) Het downloaden van virussen of malware.
- d) Verbinding maken met de echte website in plaats van een nepserver.

**Antwoord**

c) Het downloaden van virussen of malware.

## Hoofdstuk 3: Encryptie

5. Wat is de definitie van encryptie?

- a) Het proces van het verbergen van bestanden op een computer.
- b) Het omzetten van gegevens zodat ze onleesbaar zijn voor onbevoegden, tenzij men de juiste 'sleutel' heeft.
- c) Het back-uppen van gegevens naar een externe schijf.
- d) Het controleren van de integriteit van gegevens.

**Antwoord**

b) Het omzetten van gegevens zodat ze onleesbaar zijn voor onbevoegden, tenzij men de juiste 'sleutel' heeft.

6. Welk type encryptie gebruikt dezelfde sleutel voor zowel versleuteling als ontsleuteling?

- a) Asymmetrische encryptie
- b) Hash-encryptie
- c) Symmetrische encryptie
- d) Kwantumencryptie

#### □ Antwoord

c) Symmetrische encryptie

## Hoofdstuk 4: Hashing

7. Waarom wordt hashing vaak gebruikt voor het opslaan van wachtwoorden?

- a) Omdat de wachtwoorden dan eenvoudig terug te rekenen zijn voor de gebruiker.
- b) Omdat het een eenrichtingsversleuteling is die niet terug te rekenen is naar het origineel.
- c) Omdat het wachtwoorden comprimeert om opslagruimte te besparen.
- d) Omdat het helpt bij het snel ophalen van verloren wachtwoorden.

#### □ Antwoord

b) Omdat het een eenrichtingsversleuteling is die niet terug te rekenen is naar het origineel.

8. Hoe controleert een systeem een ingevoerd wachtwoord als het opgeslagen wachtwoord gehasht is?

- a) Het systeem probeert de opgeslagen hash terug te rekenen naar het originele wachtwoord.
- b) Het systeem stuurt een resetlink naar het e-mailadres van de gebruiker.
- c) Het systeem zet het ingevoerde wachtwoord om met de hash-functie en vergelijkt het resultaat met de opgeslagen hash.

- d) Het systeem vraagt de gebruiker om een tweede authenticatiefactor.

#### □ Antwoord

c) Het systeem zet het ingevoerde wachtwoord om met de hash-functie en vergelijkt het resultaat met de opgeslagen hash.

## Hoofdstuk 5: Brute Force-aanvallen en Loginbeveiliging

9. Wat is een brute force-aanval?

- a) Een aanval waarbij een server wordt overspoeld met aanvragen.
- b) Een aanval waarbij kwaadaardige software op een systeem wordt geïnstalleerd.
- c) Een aanval waarbij systematisch heel veel verschillende wachtwoorden worden geprobeerd om toegang te krijgen.
- d) Een aanval waarbij via een formulier een database wordt gehackt.

#### □ Antwoord

c) Een aanval waarbij systematisch heel veel verschillende wachtwoorden worden geprobeerd om toegang te krijgen.

10. Welke van de volgende is **geen** methode om brute force-aanvallen te voorkomen?

- a) Een limiet stellen op het aantal pogingen.
- b) Tijdelijk een gebruiker of IP-adres blokkeren.
- c) Twee-factor authenticatie toepassen.
- d) Het gebruik van \$\_GET voor het versturen van inloggegevens.

#### □ Antwoord

d) Het gebruik van \$\_GET voor het versturen van inloggegevens.

## Hoofdstuk 6: Rainbow tables

11. Wat is een rainbow table?

- a) Een lijst van alle mogelijke wachtwoorden.
- b) Een database van gehackte IP-adressen.
- c) Een lijst van veelgebruikte wachtwoorden met hun bijbehorende hashes.
- d) Een hulpmiddel om SSL-certificaten te genereren.

**Antwoord**

c) Een lijst van veelgebruikte wachtwoorden met hun bijbehorende hashes.

12. Waarom zijn rainbow tables gevaarlijk voor wachtwoordbeveiliging?

- a) Ze zorgen ervoor dat servers overbelast raken.
- b) Ze maken het mogelijk om gehashte wachtwoorden snel terug te vertalen naar het origineel als het wachtwoord in de tabel staat.
- c) Ze installeren malware op het systeem van de gebruiker.
- d) Ze versleutelen de communicatie tussen de gebruiker en de website.

**Antwoord**

b) Ze maken het mogelijk om gehashte wachtwoorden snel terug te vertalen naar het origineel als het wachtwoord in de tabel staat.

## Hoofdstuk 7: Salting en encryptie

13. Wat is het hoofddoel van 'salting' bij het hashen van wachtwoorden?

- a) Om het hash-algoritme complexer te maken.
- b) Om ervoor te zorgen dat hetzelfde wachtwoord elke keer een unieke hash krijgt, wat rainbow tables minder effectief maakt.
- c) Om de snelheid van het hashen te verhogen.
- d) Om te controleren of een wachtwoord sterk genoeg is.

**Antwoord**

b) Om ervoor te zorgen dat hetzelfde wachtwoord elke keer een unieke hash krijgt, wat rainbow tables minder effectief maakt.

14. Hoe wordt een gehasht wachtwoord met een 'salt' gecontroleerd bij het inloggen?

- a) Het systeem probeert de opgeslagen hash te ontsleutelen met de salt.
- b) Het ingevoerde wachtwoord wordt gehasht zonder de salt en vergeleken met de opgeslagen hash.
- c) Het ingevoerde wachtwoord wordt opnieuw gehasht samen met dezelfde opgeslagen salt, en het resultaat wordt vergeleken met de opgeslagen hash.
- d) De gebruiker wordt gevraagd om de salt handmatig in te voeren.

# OOP

## *1 Wat is OOP en waarom gebruiken we het?*

### □□ Leerdoelen

- Je begrijpt het verschil tussen procedureel programmeren en OOP.
- Je weet wat een class, object en method zijn.
- Je kunt herkennen wanneer OOP handig is in grotere projecten.

### □□ Uitleg

#### Wat is OOP?

OOP staat voor **Objectgeoriënteerd programmeren** (object geörienteerd programmeren). In plaats van losse functies en variabelen, werk je met objecten die gedrag (via classes) en gegevens (properties) combineren.

#### Waarom OOP gebruiken?

- Je kunt code beter organiseren en hergebruiken.
- Je maakt je programma makkelijker uitbreidbaar.
- Je kunt complexe systemen opdelen in logische blokken (objecten).

#### Voorbeeld: procedureel vs OOP

##### Procedureel:

```
<?php  
$naam = "Tessa";
```



```
$leeftijd = 19;

function begroet($naam) {
    echo "Hallo, ik ben " . $naam;
}

begroet($naam);
?>
```

## Objectgeoriënteerd:

```
<?php
class Persoon {
    public $naam;
    public $leeftijd;

    public function begroet() {
        echo "Hallo, ik ben " . $this->naam;
    }
}

$tessa = new Persoon();
$tessa->naam = "Tessa";
$tessa->leeftijd = 19;
$tessa->begroet();
?>
```

☐ Je ziet hier dat de functie en de variabelen in één blok (class) staan.

We gebruiken bij OOP soms vreemde termen:

Procedureel	OOP
variabele	<b>property</b> (van een class)
functie	<b>method</b>

## ☐ Opdracht – Van procedureel naar objectgeoriënteerd

Je gaat een simpel PHP-script omzetten naar OOP.

1. Maak een bestand `mijn-procedureel.php` en schrijf de volgende code:

```
<?php
$autoMerk = "Toyota";
$bouwjaar = 2018;

function toonAuto($merk, $jaar) {
    echo "Dit is een $merk uit $jaar";
}

toonAuto($autoMerk, $bouwjaar);
?>
```

2. Maak nu een tweede bestand: `mijn-oop.php` en herschrijf dit script in OOP-stijl:

```
<?php
class Auto {
    ...
    ...
    ...
    ...
}

$auto = new Auto();
$auto->merk = "Toyota";
$auto->bouwjaar = 2018;
$auto->toon();
?>
```

## ☐☐ Reflectie

- Wat vond je makkelijker: de procedurele of de OOP-versie?
- Waarom denk je dat OOP belangrijk is bij grotere projecten?

## ☐☐ Inleveren

- Lever beide bestanden in: `mijn-procedureel.php` en `mijn-oop.php`
- Maak een kort reflectiebestand: `reflectie-les1-<jouw-naam>.txt`
- Let op: je code moet uitvoerbaar zijn in de browser (via XAMPP of localhost).

## 2 Classes en objects

### □□ Leerdoelen

- Je weet wat een class en een object zijn.
- Je kunt een eenvoudige class maken in PHP.
- Je kunt een object aanmaken op basis van die class en ermee werken.

### □□ Uitleg

#### Wat is een class?

Een class is een soort blauwdruk of bouwtekening. Je beschrijft hiermee hoe een bepaald type object eruitziet en wat het kan doen.

#### Wat is een object?

Een object is een concreet exemplaar van een class. Je maakt een object aan met `new`. Een class kun je meerdere keren gebruiken om verschillende objecten te maken.

Class, Object, Properties en Methods worden nog een keer duidelijk gemaakt in:

<https://slides.com/jamescandan/oop-php-for-beginners-1-2/fullscreen#/5>

<https://www.youtube.com/embed/tZJcsFp9ttU>

Dus een class is een sjabloon van een object en met het commando `new <class_naam>` maken we een nieuw object van de sjabloon. In een class heten de variabelen **properties** en de functies heten **methods**.

#### Voorbeeld:

```

<?php
class Hond {
    public $naam;
    public $leeftijd;

    public function blaf() {
        echo "Woef! Ik ben " . $this->naam;
    }
}

$mijnHond = new Hond();
$mijnHond->naam = "Rex";
$mijnHond->leeftijd = 3;
$mijnHond->blaf();

$jouwHond = new Hond();
$jouwHond->naam = "Sip";
$jouwHond->leeftijd = 5;
$jouwHond->blaf();
?>

```

☐ Je hebt **één class**, maar je hebt **twee objecten** (honden) van deze class: `$mijnHond` en `$jouwHond`

## ☐ Opdracht – Maak je eigen class

1. Maak een nieuw bestand: `mijn-klas.php`
2. Maak hierin een class `Student` met de eigenschappen `voornaam`, `achternaam` en `opleiding`.
3. Maak een methode `stelVoor()` die zegt: "Hallo, ik ben [voornaam] [achternaam] en ik doe de opleiding [opleiding]."
4. Maak **drie** objecten ( `$student01`, `$student02`, en `$student03` ) van de class en test het resultaat in je browser.

## ☐ Reflectie

- Wat viel je op bij het maken van een object?

- Wat gebeurt er als je een properties (voornaam, achternaam en opleiding) **niet** vult voor je `stelVoor()` gebruikt?

## ☐☐ Inleveren

- Bestandsnaam: `mijn-klas.php`
- Reflectievragen in een apart txt- of PDF-bestand.

# 3 Constructors en eigenschappen

## ☐☐ Leerdoelen

- Je weet wat een constructor is in een class.
- Je kunt automatisch waarden toekennen bij het maken van een object.
- Je begrijpt het verschil tussen eigenschappen (properties) en methoden.

## ☐☐ Uitleg

### Wat is een constructor?

Een constructor is een speciale methode die automatisch wordt uitgevoerd als je een object maakt. In PHP heet deze methode `__construct()`.

### Waarom handig?

Met een constructor hoef je niet handmatig eigenschappen toe te wijzen nadat je een object hebt aangemaakt.

### Voorbeeld zonder constructor:

```
<?php
class Auto {
    public $merk;
    public $bouwjaar;
}
```

```
$auto = new Auto();  
$auto->merk = "Toyota";  
$auto->bouwjaar = 2018;  
?>
```

## Voorbeeld met constructor:

```
<?php  
class Auto {  
    public $merk;  
    public $bouwjaar;  
  
    public function __construct($merk, $bouwjaar) {  
        $this->merk = $merk;  
        $this->bouwjaar = $bouwjaar;  
    }  
}  
  
$auto = new Auto("Toyota", 2018);  
?>
```

## Opdracht – Constructor gebruiken

1. Maak een nieuw bestand: `les3-student.php`
2. Maak opnieuw een class `Student` met eigenschappen: `voornaam`, `achternaam` en `opleiding`.
3. Voeg een constructor toe waarin deze waarden worden ingesteld.
4. Maak een methode `stelVoor()` die de gegevens netjes toont.
5. Maak minstens twee objecten aan en laat de `stelVoor()`-methode zien in de browser.

## Reflectie

- Wat is het voordeel van een constructor ten opzichte van handmatige toekenning?
- Wat gebeurt er als je het aantal parameters niet goed doorgeeft?

- Wat betekent `$this->...` in de constructor?

## ☐☐ Inleveren

- Bestandsnaam: `les3-student.php`
- Reflectievragen in een apart bestand: `reflectie-les3-<jouw-naam>.txt`

# 4 Methoden met parameters en returnwaarden

## ☐☐ Leerdoelen

- Je begrijpt hoe je een methode maakt die parameters accepteert.
- Je kunt een methode schrijven die een waarde teruggeeft met `return`.
- Je ziet hoe objectmethoden kunnen samenwerken met parameters uit de code.

## ☐☐ Uitleg

### Wat is een parameter?

Een parameter is een waarde die je doorgeeft aan een methode, zodat deze methode die waarde kan gebruiken in de berekening of actie.

### Wat is een returnwaarde?

Met `return` geef je een resultaat terug uit een methode, zodat je dat resultaat ergens anders kunt gebruiken.

### Voorbeeld:

```
<?php
class Rekening {
    public $saldo;
```

```

public function __construct() {
    $this->saldo = 0;
}

public function stort($bedrag) {
    // vul hier de code aan
}

public function toonSaldo() {
    // vul hier de code aan
}
}

$mijnRekening = new Rekening();
$mijnRekening->stort(50);
echo "Je saldo is: ". $mijnRekening->toonSaldo();
?>

```

## Opdracht – Werk met parameters en return

1. Maak een nieuw bestand: `les4-rekening.php`
2. Maak een class `Rekening` met een startsaldo van 0.
3. Voeg een methode `stort($bedrag)` toe die het bedrag bij het saldo optelt.  
Tip: vegeet ni `$this->` te gebruiken!
4. Voeg een methode `opname($bedrag)` toe die het bedrag van het saldo afhaalt.
5. Voeg een methode `toonSaldo()` toe die het saldo teruggeeft met `return`.
6. Test de class met een **paar stortingen** en **paar** opnames. Toon aan dat:
  - dat stortingen het saldo verhogen
  - dat opnames het saldo verlagen
  - dat een te hoge opname het saldo niet verandert

Probeer het zonder code, maar kom je er niet uit dan kun je deze code als template gebruiken .



## Template

```
echo "<pre>";

$rekening = new Rekening();

// Test 1: stort 50
// .....
echo "Test 1: verwacht 50 → resultaat: ".....

// Test 2: opname van 20
// .....
echo "Test 2: verwacht 30 → resultaat: " .....

// Test 3: opname van 100 (te veel)
.....
echo "Test 3: verwacht nog steeds 30 → resultaat: " .....

echo "</pre>";
```

## ☐☐ Reflectie

- Wat gebeurt er als je een opname doet die groter is dan je saldo?
- Waarom heeft de constructor in de class Rekening in het voorbeeld geen parameter (om het saldo op 0 te zetten)?
- Wat zijn de voordelen van werken met `return`?

## ☐☐ Inleveren

- Bestandsnaam: `les4-rekening.php`
- Reflectievragen in een apart bestand: `reflectie-les4-<jouw-naam>.txt`

# 5 Checkpoint: Wat heb je geleerd?

## ☐☐ Samenvatting

### 1. Classes & Objects

- Je kent het verschil tussen een class (blauwdruk) en een object (instantie).
- Je weet hoe je met `new` een object maakt.

### 2. Properties & Methods

- Je gebruikt `public` properties om data in een object op te slaan.
- Je roept methods aan met parameters (`method($param)`) en begrijpt wat een returnwaarde is.

### 3. Constructors

- Je kunt een `__construct()` schrijven om bij creatie van een object meteen eigenschappen in te stellen.
- Je weet dat `$this->...` verwijst naar de huidige instantie.

### 4. Encapsulation Basics

- Je begrijpt waarom je data soms `private` maakt en via getters/setters benadert.

“**Tip:** Ga kort na of je in je eigen voorbeelden bovenstaande concepten herkent en toepast voordat je verdergaat naar encapsulation en private properties in Les 5.

## ☐☐ Opdracht

Maak van elk van de vier genoemde onderwerpen een multiple choice vraag, met 3 foute en één goed antwoord. Als de vragen goed zijn dan zullen ze terug komen in de kennis-check.

### Voorbeeld/template

Vraag: Wat is een object?

A: Dat is een variabele in een class.

B: Dat is een instantie van een class.

C: Dat is een functie in een class  
D: Dat is een data-structuur in PHP.  
Juiste antwoord is B

Vraag:

A:

B:

C:

D:

Juiste antwoord is:

Vraag:

A:

B:

C:

D:

Juiste antwoord is:

Vraag:

A:

B:

C:

D:

Juiste antwoord is:

Vraag:

A:

B:

C:

D:

Juiste antwoord is:

## Inleveren

De vier multiple choice vragen in txt bestand.

# 6 Encapsulation en private properties

## Leerdoelen

- Je weet wat encapsulation is.
- Je begrijpt het verschil tussen `public` en `private`.
- Je kunt `getters` en `setters` gebruiken om toegang tot eigenschappen te regelen.

## Uitleg

Encapsulation betekent dat je data (properties van een class) beschermt tegen direct aanpassen van buiten de class. In plaats daarvan gebruik je methods om de data op een gecontroleerde manier te bekijken of aan te passen.

## Waarom gebruiken?

Zo voorkom je fouten doordat andere delen van je code zomaar eigenschappen aanpassen zonder controle.

## Voorbeeld:

```
<?php
class Bankrekening {
    private $saldo = 0;

    public function stort($bedrag) {
        if ($bedrag > 0) {
            $this->saldo += $bedrag;
        }
    }

    public function getSaldo() {
        return $this->saldo;
    }
}
```

```
}

$rekening = new Bankrekening();
$rekening->stort(100);
echo "Saldo: ".$rekening->getSaldo();
?>
```

## Opdracht – Bescherm je data

1. Maak een nieuw bestand: `les5-bankrekening.php`
2. Maak een class `Bankrekening` met een `private` eigenschap `$saldo`.
3. Maak een `stort()`-methode die alleen positief bedrag accepteert.
4. Maak een `getSaldo()`-methode die het saldo teruggeeft.
5. Test de class met verschillende stortingen en probeer ook een negatieve storting (die moet worden geweigerd).
6. Maak nu een `opnemen($bedrag)` waarmee je geld kan opnemen. Zorg ervoor dat je alleen geld kan opnemen als je voldoende saldo hebt. Heb je dat niet, dan vind er geen opname plaats.
7. Test de method `opname($bedrag)` uit!

## Reflectie

- Wat is het verschil tussen `public` en `private`?
- Waarom zou je eigenschappen `private` maken?
- Wat gebeurt er als je probeert `$saldo` direct aan te passen: `$rekening->saldo = 1000;`  
Wat zie je? En hoe verklaar je dat?

## Inleveren

- Bestandsnaam: `les5-bankrekening.php`
- Reflectievragen in pdf.

# 7 Bibliotheekstelsysteem met Twee Classes

## Leerdoelen

- Je maakt en gebruikt meerdere classes in één project.
- Je begrijpt hoe objecten van verschillende classes met elkaar communiceren.
- Je oefent met private properties, getters/setters en method-parameters.
- Je bouwt een klein functioneel systeem met relaties tussen objecten.

## Uitleg

In dit mini-project bouw je een eenvoudig bibliotheekstelsysteem met slechts twee classes:

1. **Book:** een boek met een titel, auteur en beschikbaarheidsstatus.
2. **Library:** beheert een collectie boeken en wie ze geleend heeft.

## Opdracht – Bouw je eigen bibliotheek

### Book Class

Book.php bevat de **class Book** waarmee je het object book kan maken.

Een boek heeft een **title**, een **author** en een **available (beschikbaarheid)**. De class bevat **getters** en **setters** om deze properties te lezen en aan te passen.

#### book.php

```
<?php
class Book {
    private string $title;
    private string $author;
```

```

private bool $available = true;

public function __construct(string $title, string $author) {
    $this->title = $title;
    $this->author = $author;
}

public function getTitle(): string {
    return $this->title;
}

public function getAuthor(): string {
    return $this->author;
}

public function isAvailable(): bool {
    return $this->available;
}

public function setAvailable(bool $avail): void {
    $this->available = $avail;
}
}
?>

```

## Library Class

De **class Library** maakt gebruik van de class Book. Deze class kan boeken **toevoegen** aan de library, kan boeken **uitlenen** en boeken weer **terugnemen**. Als laatst is er een method waarmee je alle boeken uit de library kan tonen.

### library.php

```

<?php
require_once 'book.php';

```

```
class Library {

    private array $books = [];
    private array $loans = [];

    public function addBook(Book $book): void {
        // Voeg het boek toe onder de sleutel van de titel
        $this->books[$book->getTitle()] = $book;
    }

    public function lendBook(string $title, string $member): string {
        // Bestaat het boek?
        if (!isset($this->books[$title])) {
            return "Boek “{$title}” bestaat niet.\n";
        }
        $book = $this->books[$title];

        // Is het al uitgeleend?
        if (!$book->isAvailable()) {
            return "Boek “{$title}” is al uitgeleend.\n";
        }

        // Leen uit
        $book->setAvailable(false);
        $this->loans[$title] = $member;
        return "{$member} leent “{$title}”.\n";
    }

    public function returnBook(string $title): string {
        // Bestaat het boek?
        if (!isset($this->books[$title])) {
            return "Boek “{$title}” bestaat niet.\n";
        }

        // Was het uitgeleend?
        if (!isset($this->loans[$title])) {
            return "Boek “{$title}” was niet uitgeleend.\n";
        }
    }
}
```



```

    // Verwerk retour
    $member = $this->loans[$title];
    unset($this->loans[$title]);
    $this->books[$title]->setAvailable(true);
    return "{$member} brengt “{$title}” terug.\n";
}

public function listBooks(): string {
    $output = "";
    foreach ($this->books as $book) {
        if ($book->isAvailable()) {
            $status = 'beschikbaar';
        } else {
            $status = 'uitgeleend aan ' . $this->loans[$book->getTitle()];
        }
        $output .= $book->getTitle() . ' - ' . $status . "\n";
    }
    return $output;
}
}

```

## Main code

In de main code wordt de class library gebruikt om boeken in de library te zetten en uit te lenen of terug te nemen.

Met deze code wordt de library- en book class getest.

Kijk goed wat er gebeurt en probeer het te begrijpen.

### main.php

```

<?php
require_once 'book.php';
require_once 'library.php';

echo '<pre>';

```

```
$lib = new Library();  
$lib->addBook(new Book('1984', 'George Orwell'));  
$lib->addBook(new Book('De Avonden', 'Gerard Reve'));  
  
echo "Leen boek 1884 uit aan Alice<br>resultaat: ";  
echo $lib->lendBook('1984', 'Alice');  
echo "<br>";  
  
echo "Leen boek 1884 uit aan Bob<br>resultaat: ";  
echo $lib->lendBook('1984', 'Bob'); // Alice heeft het nog  
echo "<br>";  
  
echo "Laat alle Boeken zien<br>resultaat: ";  
echo $lib->listBooks();  
echo "<br>";  
  
echo "Boek 1984 wordt tgeruggebracht<br>resultaat: ";  
echo $lib->returnBook('1984');  
echo "<br>";  
  
echo "Leen boek 1884 uit aan Bob<br>resultaat: ";  
echo $lib->lendBook('1984', 'Bob'); // nu kan Bob lenen  
echo "<br>";  
  
echo "Laat alle Boeken zien<br>resultaat: ";  
echo $lib->listBooks();  
echo "<br>";  
  
echo '</pre>'  
?>
```

## Overzicht

**Opdracht:** draai dit script in je browser en verifieer de uitvoer.

**Hier zie je een eenvoudige UML-class-diagram die laat zien hoe de class Library en de class Book met elkaar communiceren**

image.png  
image background or type unknown

- **Library**

- Bevat een collectie ( `books` ) van `Book`-objecten.
- Methoden zoals `addBook()`, `lendBook()`, `returnBook()` en `listBooks()` gebruiken de `Book`-objecten.

- **Book**

- Heeft private eigenschappen: `title`, `author`, `available` (en eventueel `loansCount`).
- Biedt publieke getters ( `getTitle()`, `isAvailable()` ) en een setter ( `setAvailable()` ) om zijn status te wijzigen.

De pijlen geven aan dat de `Library` klas methoden aanroept op de `Book` objecten, bijvoorbeeld bij het uitlenen (zetten `available` op `false`) en het terugbrengen (zetten `available` op `true`). Zo zie je visueel hoe de objecten samenwerken binnen het systeem.

## Opdracht

Van elk boek wi je bijhouden hoe vaak het is uitgeleend.

1. Voeg een property toe aan de class `Book` en noem die `$aantalKeerUitgeleend`.
2. Verhoog de property als een boek wordt uitgeleend met 1.
3. Voeg een method `getUitleenData()` toe in de class `Book` en laat deze van elk boek, de titel, auteur en het aantal keer dat dit boek is uitgeleend zien.
4. Test dit via de code in `mainn.php`, 1984 is al twee maal uitgeleend, maak nog een derde boek en leen dat ook één maal uit.

Aan het eind hebben we dus drie boeken: één boek is nooit uitgeleend, één boek is 1x uitgeleend en één boek is twee maal uitgeleend.

## Reflectie

1. Wat doet de `foreach loop` in de method `listBooks()` ?
2. Waarom is `$author` private in **Book** en kan je de `$author` van een `Book` wijzigen?

3. In de key van `$books` uit de class `Library` staat de *titel van het boek*, wat staat er in de *value*?
4. Beschrijf wat er in de *key* en de *value* staat van de `$loans` uit de class `Library`.

## ☐☐ Inleveren

1. De aangepast class `Book` (`book.php`).
2. De aangepaste `main.php`
3. Antwoord op de reflectievragen (`pdf`)

## 8 Complete mini-project

(ga niet verder voordat je de vorige opdracht goed hebt afgerond)

## ☐☐ Leerdoelen

- Je past classes, objects en constructors toe.
- Je gebruikt public en private properties met getters/setters.
- Je schrijft methods met parameters en returnwaardes.
- Je beheert data in een klein OOP-systeem.

## ☐☐ Uitleg

Je bouwt een klein “winkelmandje”-systeem in PHP. Je maakt twee classes:

- `Product` met `private $naam`, `private $prijs` en een constructor.  
Voeg getters toe: `getNaam()` en `getPrijs()`.
- `Winkelmandje` met `private $items = []`.  
Voeg methods toe om `voegToe(Product $p, int $aantal)`, `verwijder(Product $p)` en `getTotaal(): float` te implementeren.

## ☐☐ Opdracht – Winkelmandje bouwen

1. Maak bestand `les6-product.php` en definieer class `Product`:

```
<?php
class Product {
    private $naam;
    private $prijs;

    public function __construct(string $naam, float $prijs) {
        // ...
    }

    public function getNaam(): string {
        // ...
    }

    public function getPrijs(): float {
        // ...
    }
}
?>
```

2. Maak bestand `les6-winkelmandje.php` en definieer class `Winkelmandje`:

```
<?php
class Winkelmandje {
    private $items = []; // array van ['product' => Product, 'aantal' => int]

    public function voegToe(Product $p, int $aantal): void {
        // voeg product toe of verhoog aantal
        // Kijk of het product al in het mandje zit

        // loop door het winkelmandje heen
        foreach ($this->items as &$item) {
            if ($item['product']->getNaam() === $p->getNaam()) { // kijk of we het product al hebben
                // Verhoog het bestaande aantal
                $item['aantal'] += $aantal;
                return; // klaar
            }
        }
    }
}
```

```

        // Anders (hebben we geen return gehad dus: nieuw item toevoegen
        $this->items[] = [
            'product' => $p,
            'aantal' => $aantal
        ];
    }

    public function verwijder(Product $p): void {
        // verwijder product volledig uit items
        .....
    }

    public function getTotaal(): float {
        // bereken en return totaalprijs
        .....
    }
}
?>

```

### 3. Maak een testscript `les6-test.php`:

```

<?php
require 'les6-product.php';
require 'les6-winkelmandje.php';

$p1 = new Product("Boek OOP", 25.50);
$p2 = new Product("USB-stick", 8.99);

$mandje = new Winkelmandje();
$mandje->voegToe($p1, 2);
$mandje->voegToe($p2, 1);
$mandje->verwijder($p2);

echo "Totaal: €" . $mandje->getTotaal();
?>

```

## Reflectie

- Hoe zorg je ervoor dat een `Product` niet negatief geprijsd kan zijn?
- Wat gebeurt er als je `voegToe` aanroept met `$aantal = 0`?
- Waarom gebruik je hier `private` properties en geen `public`?
- Hoe kun je de code uitbreiden met een `updateAantal(Product $p, int $nieuwAantal)`?
- In de function `voegToe(Product $p, int $aantal)` moeten we controleren of het product al bestaat. Als dat zo is dan verhogen we het aantal, anders moeten we het product aanmaken. We zouden verwachten dat we met een `if-then-else` controleren of het product al bestaat. Waarom staat er in de code geen `if-then-else`?

## ☐ Inleveren

- Bestanden: `les6-product.php`, `les6-winkelmandje.php`, `les6-test.php`
- Reflectie in apart bestand: `reflectie-les6-<jouw-naam>.txt`

# 9 Test je Kennis

### Wat betekent OOP en hoe verschilt het van procedureel programmeren?

OOP staat voor *Objectgeoriënteerd programmeren*. In plaats van functies en variabelen apart te gebruiken, bundel je bij OOP data en gedrag in objecten. Zo kun je code beter organiseren, hergebruiken en opsplitsen in logische blokken.

### Wat is een klasse in OOP?

Een klasse is een blauwdruk of sjabloon waarin je beschrijft welke gegevens (*properties*) en functies (*methods*) een object moet hebben.

### Wat is een object?

Een object is een concreet exemplaar van een klasse, gemaakt met `new`. Je kunt meerdere objecten maken van dezelfde klasse, elk met eigen waarden.

### Hoe noem je in OOP een variabele en een functie binnen een klasse?

In OOP noem je variabelen **properties**, en functies **methods**

### Wat is encapsulation?

Encapsulation betekent dat je de data (properties) van een object beschermt. Je maakt gegevens vaak **private** en gebruikt **methods** om ze gecontroleerd te lezen of aanpassen .

### Wat is het verschil tussen **public** en **private** properties/methods?

- **public**: toegankelijk en aanpasbaar van buiten de class.
- **private**: alleen toegankelijk binnen de class zelf. Dit beschermt de interne gegevens.

### Waarom is OOP handig bij grote projecten?

Omdat je code makkelijker kunt organiseren in logische blokken (objecten), hergebruiken, uitbreiden en onderhouden. Daardoor is je programma stabiel en schaalbaarder .

### Waarvoor gebruik je '\$this' -> in PHP?

`$this->` verwijst naar een property of een object uit **dit** object. Met dit object wordt bedoeld het object waar `$this->` in staat.

## 7 Inheritance (Overerving)

### 📋 Leerdoelen

- Je weet wat inheritance (overerving) betekent in OOP.
- Je kunt een class maken die eigenschappen en methodes **erft** van een andere class.
- Je ziet waarom inheritance handig is: minder herhaling, meer overzicht.



# Uitleg

## Wat is inheritance?

Inheritance betekent dat je een nieuwe class maakt die eigenschappen en methodes **overneemt** van een andere class. Dit heet overerving. De 'ouderclass' noem je ook wel de *superclass*, en de 'kindclass' de *subclass*.

## Voorbeeld

```
<?php
class Dier {
    public $naam;

    public function adem() {
        echo $this->naam . " ademt.\n";
    }
}

class Hond extends Dier {
    public function blaf() {
        echo $this->naam . " zegt: Woef!\n";
    }
}

$rex = new Hond();
$rex->naam = "Rex";
$rex->adem(); // komt uit Dier
$rex->blaf(); // komt uit Hond
?>
```

De class `Hond` **erft** de methode `adem()` van `Dier`, maar voegt ook zijn eigen gedrag toe: `blaf()`.

## Opdracht – Maak je eigen dier met overerving

### 1. Bestand: `dier.php`

Maak een class `Dier` met:

- een property `$naam`
- een methode `beweeg()` die `"{$this->naam} beweegt."` toont

## 2. Bestand: **vogel.php**

Maak een class `Vogel` die `extends Dier`:

- een methode `vlieg()` die `"{$this->naam} vliegt!"` toont

## 3. Bestand: **test.php**

- Maak een object `$mus` van de class `Vogel`
- Stel zijn naam in op `"Mus"`
- Roep zowel `beweeg()` als `vlieg()` aan

# ☐☐ Reflectie

- Wat gebeurt er als je `vlieg()` oproept op een `Dier`-object?
- Kun je uitleggen waarom `Vogel` de methode `beweeg()` kan gebruiken zonder die zelf te schrijven?

# ☐☐ Inleveren

- `dier.php`, `vogel.php`, `test.php`
- Reflectie in `reflectie-les7-<jouw-naam>.txt`

# OOP Challenge

## *Challenge A: OOP Login Systeem*

### □□ Inleiding, wat ga je doen?

De **OOP Challenge** is een fantastische kans om jouw vaardigheden in **objectgeoriënteerd programmeren** (OOP) naar een hoger niveau te tillen en direct toe te passen in de praktijk. Je combineert je **Cyber Security** vaardigheden met de **OOP** vaardigheden.

Je hebt de keuze uit twee boeiende projecten;

- je maakt een [OOP Login Systeem](#), of
- je maakt een [\(mobiele\) ToDo manager](#) die je kan omzetten naar een mobiele applicatie, dit is lastiger maar ook veel leuker want je maakt een echte mobiele app!

### □□ Leerdoelen

- Je past klassen, objecten, constructors en methodes toe in een realistisch scenario.
- Je gebruikt encapsulation om gebruikersgegevens te beveiligen.
- Je slaat gebruikers op met PDO en hashed wachtwoorden.
- Je maakt een registratie- en loginroutine in OOP-stijl.

### □□ Opdrachtbeschrijving

**Situatie:** Je werkt bij een klein webbureau en een klant wil een eenvoudig maar veilig **login-systeem** laten bouwen in PHP. Je gaat dit project zelfstandig uitvoeren als een mini-challenge. Dit is een uitstekende kans om je **OOP-vaardigheden** te laten zien in je **GitHub portfolio**. Dat kan je helpen om sneller een stage of baan te vinden in de webdevelopment-sector!

**Wat je gaat bouwen:**

- Een registratiepagina waar nieuwe gebruikers zich kunnen aanmelden
- Een loginpagina waar gebruikers zich kunnen aanmelden
- Beveiligde opslag van wachtwoorden via `password_hash()`
- Gebruik van `PDO` voor communicatie met de database (bijv. SQLite of MySQL)

## ☐☐ Structuur en Klassen

- **Class User**
  - Properties: `private $id`, `$username`, `$passwordHash`
  - Constructor stelt naam + wachtwoord in (en hashed het wachtwoord)
  - Methode `verifyPassword($plainText)` vergelijkt met hash
  - Getters voor `getId()` en `getUsername()`
- **Class UserDatabase**
  - Maakt verbinding via `PDO` (gebruik SQLite of MySQL)
  - Methodes: `addUser(User $user)`, `findUserByUsername($name)`
  - Prepared statements verplicht!

## ☐☐ Technische vereisten

- Gebruik **`password_hash()`** en **`password_verify()`** voor wachtwoorden
- PDO met **prepared statements** (geen ruwe SQL!)
- Toon foutmeldingen bij fouten (bestaande user, fout wachtwoord, etc.)

## ☐☐ Bestanden

- `User.php` – beschrijft de user zelf
- `UserDatabase.php` – regelt opslag en ophalen
- `register.php` – laat gebruiker registreren
- `login.php` – laat gebruiker inloggen
- `login-success.php` – pagina die je ziet na succesvol inloggen

## ☐☐ Reflectie

- Wat zijn de voordelen van wachtwoord hashing?
- Waarom gebruiken we een aparte klasse voor opslag en een voor gebruikers?
- Wat zou je uitbreiden als je dit systeem echt online wilde zetten?

## ☐☐ Bonus: Voor je portfolio

Zet dit project op **GitHub** met een korte uitleg in de README over hoe het werkt. Voeg schermafbeeldingen toe van het registratie- en loginproces. Dit is een mooi voorbeeldproject om te laten zien wat je kunt!

## ☐ Gebruik AI

AI gebruik op de juiste manier, zoals geleerd tijdens de lessen, is toegestaan en wordt zelfs aangemoedigd.

Het is wel verplicht om de AI log in te leveren en je moet je code kunnen uitleggen!

## ☐☐ Inleveren

- AI log - alle hulp van AI inleveren!
- SQL export.
- Alle bestanden nodig voor jouw website.
- Reflectie in PDF
- (Optioneel) Link naar je GitHub repository

## *Challenge B: (mobiele) OOP ToDo Manager*

## ☐☐ Inleiding, wat ga je doen?

De **OOP Challenge** is een fantastische kans om jouw vaardigheden in **objectgeoriënteerd programmeren** (OOP) naar een hoger niveau te tillen en direct toe te passen in de praktijk. Je combineert je **Cyber Security** vaardigheden met de **OOP** vaardigheden.

Je hebt de keuze uit twee boeiende projecten;

- je maakt een [OOP Login Systeem](#), of
- je maakt een [\(mobiele\) ToDo manager](#) die je kan omzetten naar een mobiele applicatie, dit is lastiger maar ook veel leuker want je maakt een echte mobiele app!

## ▣▣ Leerdoelen

- Je past OOP toe in een realistisch webproject.
- Je maakt meerdere klassen die met elkaar samenwerken.
- Je gebruikt PDO en password hashing op een veilige manier.
- Je bouwt een werkend mini-systeem dat je op GitHub kunt zetten.

## ▣▣ Opdrachtbeschrijving

**Situatie:** Een start-up wil een prototype van een eenvoudige "ToDo Manager" waarin gebruikers taken kunnen aanmaken, afvinken en verwijderen. Jij bouwt dit systeem in PHP met OOP. Dit is een mooie kans om een compleet mini-project te maken voor je **GitHub portfolio** — ideaal om te laten zien bij een sollicitatie voor stage of werk!

## ▣▣ Functionaliteiten

- Registratiepagina: nieuwe gebruikers kunnen zich aanmelden
- Loginpagina: gebruikers kunnen inloggen met hun wachtwoord
- Taakpagina: ingelogde gebruikers kunnen taken toevoegen, afvinken en verwijderen
- Alle data wordt opgeslagen in een database met **PDO** (SQLite of MySQL).

## ▣▣ Mobiele App (optioneel)

- Je kan de ToDo manager omzetten naar een **Progressive Web App**. Je krijgt de beste gebruikers ervaring als je je ToDo manager zoveel mogelijk omzet naar een **Single Page Web Application** (SPA). Dat kan met een Framework, maar ook met JavaScript.

## Wat is een SPA en een Progressive Web App?

Een **one page applicatie** (of **Single Page Application**, afgekort **SPA**) is een webapplicatie of website die uit **één HTML-pagina** bestaat en **dynamisch inhoud laadt**, zonder de hele pagina opnieuw te laden bij navigatie of interactie.

### ☐ Hoe werkt het?

Bij een SPA:

- Wordt bij het eerste bezoek één HTML-pagina geladen.
- Daarna worden alleen **delen van de pagina aangepast via JavaScript** (meestal met behulp van frameworks zoals React, Vue of Angular).
- Communicatie met de server gebeurt via **AJAX** of **fetch()-aanroepen** om data op te halen of op te slaan, vaak in JSON-formaat.

Je kunt een SPA omzetten in een PWA, Progressieve Web App.

### ☐ Wat is een PWA?

Een **PWA** is een **website die aanvoelt als een app**. Je opent hem in de browser, maar je kunt hem ook **op je telefoon zetten als icoon**, net zoals een echte app uit de App Store of Play Store.

### ☐ Hoe werkt het?

Een PWA:

- Start gewoon via je browser (zoals Safari of Chrome)
- Kan op je **beginscherm gezet worden** als een app-icoon
- **Werkt ook offline** als je het goed instelt
- Ziet eruit en werkt zoals een gewone app

*Wil je een Mobiele app op deze manier maken, laat AI je dan de details verder uitleggen,*

# ☐☐ Klassen

- **Class User**

- Properties: `$id`, `$username`, `$passwordHash`
- Constructor en `verifyPassword($plainText)`

- **Class UserDatabase**

- Maakt verbinding via `PDO`
- Methodes: `addUser()`, `findUserByUsername()`

- **Class Task**

- Properties: `$id`, `$user_id`, `$description`, `$completed`
- Methodes: `toggleCompleted()`, `getDescription()`, `isCompleted()`

- **Class TaskManager**

- Methodes: `addTask()`, `getTasksByUser()`, `deleteTask()`, `toggleTask()`
- Alle database-interacties via `PDO` met prepared statements

# ☐☐ Reflectie

- Waarom werken we met meerdere klassen in plaats van één grote?
- Wat gebeurt er als een niet-ingelogde gebruiker naar de takenpagina navigeert?
- Hoe zorgt jouw systeem ervoor dat taken niet tussen gebruikers verwisseld kunnen worden?

# ☐☐ Bonus voor je portfolio

Zet dit project op **GitHub**. Voeg screenshots toe en schrijf een README-bestand waarin je uitlegt wat het systeem doet. Dit laat aan stagebedrijven of werkgevers zien dat jij zelfstandig een werkend OOP-project kunt bouwen!

# ☐ Gebruik AI

AI gebruik op de juiste manier, zoals geleerd tijdens de lessen, is toegestaan en wordt zelfs aangemoedigd.



Het is wel verplicht om de AI log in te leveren en je moet je code kunnen uitleggen!

## Inleveren

- AI log - alle hulp van AI inleveren!
- SQL export.
- Alle bestanden nodig voor jouw website.
- Reflective in PDF
- (Optioneel) Link naar je GitHub repository

--

# OOP Inheritance

## 1 Inheritance (Overerving)

### Leerdoelen

- Je weet wat inheritance (overerving) betekent in OOP.
- Je kunt een class maken die eigenschappen en methodes **erft** van een andere class.
- Je ziet waarom inheritance handig is: minder herhaling, meer overzicht.

### Uitleg

#### Wat is inheritance?

Inheritance betekent dat je een nieuwe class maakt die eigenschappen en methodes **overneemt** van een andere class. Dit heet overerving. De 'ouderclass' noem je ook wel de *superclass*, en de 'kindclass' de *subclass*.

#### Voorbeeld

```
<?php
class Dier {
    public $naam;

    public function adem() {
        echo $this->naam . " ademt.\n";
    }
}

class Hond extends Dier {
    public function blaf() {
        echo $this->naam . " zegt: Woef!\n";
    }
}
```

```
}

$rex = new Hond();
$rex->naam = "Rex";
$rex->adem(); // komt uit Dier
$rex->blaf(); // komt uit Hond
?>
```

□□ De class `Hond` **erft** de methode `adem()` van `Dier`, maar voegt ook zijn eigen gedrag toe: `blaf()`.

## □□ Opdracht – Maak je eigen dier met overerving

### 1. Bestand: `dier.php`

Maak een class `Dier` met:

- een property `$naam`
- een methode `beweeg()` die `"{$this->naam} beweegt."` toont

### 2. Bestand: `vogel.php`

Maak een class `Vogel` die `extends Dier`:

- een methode `vlieg()` die `"{$this->naam} vliegt!"` toont

### 3. Bestand: `test.php`

- Maak een object `$mus` van de class `Vogel`
- Stel zijn naam in op `"Mus"`
- Roep zowel `beweeg()` als `vlieg()` aan

## □□ Reflectie

- Wat gebeurt er als je `vlieg()` oproept op een `Dier`-object?
- Kun je uitleggen waarom `Vogel` de methode `beweeg()` kan gebruiken zonder die zelf te schrijven?

## □□ Inleveren

- `dier.php` , `vogel.php` , `test.php`
- Reflectie in `reflectie-les7-<jouw-naam>.txt`

# Kennis Check Blok 8

## *Cyber Security*

### Hoofdstuk 1: Wat is Cyber Security?

1. Wat is de primaire focus van Cyber Security?

- a) Het beschermen van websites tegen computer-bugs
- b) Het beschermen van computersystemen tegen aanvallen en misbruik.
- c) Het ontwikkelen van nieuwe softwareprogramma's.
- d) Het beheren van netwerkinfrastructuur.

#### ☐ Antwoord

b) Het beschermen van computersystemen tegen aanvallen en misbruik.

2. Welke van de volgende cyberaanvallen omvat het misleiden van iemand om wachtwoorden af te geven, vaak via nep-e-mails?

- a) Malware
- b) DDoS-aanval
- c) Phishing
- d) SQL-injection

#### ☐ Antwoord

c) Phishing

Welke term wordt gebruikt voor software die ontworpen is om ongevraagd advertenties te tonen en je naar bepaalde webwinkels stuurt?

- a) Adware

- b) Spam
- c) Malware
- d) Commerceware

**Antwoord**

c) Adware

Welke vorm van malware kan zichzelf verspreiden naar andere bestanden of programma's zodra het op een computer is geïnstalleerd?

- a) Worm
- b) Spyware
- c) Virus
- d) Adware

**Antwoord**

c) Virus

## Hoofdstuk 2: HTTPS en netwerkveiligheid

3. Wat is het belangrijkste verschil tussen HTTP en HTTPS?

- a) HTTPS is alleen voor professionele websites, HTTP voor persoonlijke.
- b) HTTPS staat voor HyperText Transfer Protocol Secure en versleutelt de communicatie tussen browser en server.
- c) HTTP is sneller dan HTTPS.
- d) HTTP gebruikt een SSL-certificaat, HTTPS niet.

**Antwoord**

b) HTTPS staat voor HyperText Transfer Protocol Secure en versleutelt de communicatie tussen browser en server.

4. Waarvoor biedt HTTPS geen bescherming?

- a) Het onderscheppen van ingevulde wachtwoorden door derden.
- b) Het veranderen van informatie terwijl deze onderweg is.
- c) Het downloaden van virussen of malware.
- d) Verbinding maken met de echte website in plaats van een nepserver.

**Antwoord**

c) Het downloaden van virussen of malware.

## Hoofdstuk 3: Encryptie

5. Wat is de definitie van encryptie?

- a) Het proces van het verbergen van bestanden op een computer.
- b) Het omzetten van gegevens zodat ze onleesbaar zijn voor onbevoegden, tenzij men de juiste 'sleutel' heeft.
- c) Het back-uppen van gegevens naar een externe schijf.
- d) Het controleren van de integriteit van gegevens.

**Antwoord**

b) Het omzetten van gegevens zodat ze onleesbaar zijn voor onbevoegden, tenzij men de juiste 'sleutel' heeft.

6. Welk type encryptie gebruikt dezelfde sleutel voor zowel versleuteling als ontsleuteling?

- a) Asymmetrische encryptie
- b) Hash-encryptie
- c) Symmetrische encryptie
- d) Kwantumencryptie

**Antwoord**

c) Symmetrische encryptie

## Hoofdstuk 4: Hashing

7. Waarom wordt hashing vaak gebruikt voor het opslaan van wachtwoorden?

- a) Omdat de wachtwoorden dan eenvoudig terug te rekenen zijn voor de gebruiker.
- b) Omdat het een eenrichtingsversleuteling is die niet terug te rekenen is naar het origineel.
- c) Omdat het wachtwoorden comprimeert om opslagruimte te besparen.
- d) Omdat het helpt bij het snel ophalen van verloren wachtwoorden.

### Antwoord

b) Omdat het een eenrichtingsversleuteling is die niet terug te rekenen is naar het origineel.

8. Hoe controleert een systeem een ingevoerd wachtwoord als het opgeslagen wachtwoord gehasht is?

- a) Het systeem probeert de opgeslagen hash terug te rekenen naar het originele wachtwoord.
- b) Het systeem stuurt een resetlink naar het e-mailadres van de gebruiker.
- c) Het systeem zet het ingevoerde wachtwoord om met de hash-functie en vergelijkt het resultaat met de opgeslagen hash.
- d) Het systeem vraagt de gebruiker om een tweede authenticatiefactor.

### Antwoord

c) Het systeem zet het ingevoerde wachtwoord om met de hash-functie en vergelijkt het resultaat met de opgeslagen hash.

## Hoofdstuk 5: Brute Force-aanvallen en Loginbeveiliging

9. Wat is een brute force-aanval?

- a) Een aanval waarbij een server wordt overspoeld met aanvragen.
- b) Een aanval waarbij kwaadaardige software op een systeem wordt geïnstalleerd.



- c) Een aanval waarbij systematisch heel veel verschillende wachtwoorden worden geprobeerd om toegang te krijgen.
- d) Een aanval waarbij via een formulier een database wordt gehackt.

**Antwoord**

c) Een aanval waarbij systematisch heel veel verschillende wachtwoorden worden geprobeerd om toegang te krijgen.

10. Welke van de volgende is **geen** methode om brute force-aanvallen te voorkomen?

- a) Een limiet stellen op het aantal pogingen.
- b) Tijdelijk een gebruiker of IP-adres blokkeren.
- c) Twee-factor authenticatie toepassen.
- d) Het gebruik van \$\_GET voor het versturen van inloggegevens.

**Antwoord**

d) Het gebruik van \$\_GET voor het versturen van inloggegevens.

## Hoofdstuk 6: Rainbow tables

11. Wat is een rainbow table?

- a) Een lijst van alle mogelijke wachtwoorden.
- b) Een database van gehackte IP-adressen.
- c) Een lijst van veelgebruikte wachtwoorden met hun bijbehorende hashes.
- d) Een hulpmiddel om SSL-certificaten te genereren.

**Antwoord**

c) Een lijst van veelgebruikte wachtwoorden met hun bijbehorende hashes.

12. Waarom zijn rainbow tables gevaarlijk voor wachtwoordbeveiliging?

- a) Ze zorgen ervoor dat servers overbelast raken.

- b) Ze maken het mogelijk om gehashte wachtwoorden snel terug te vertalen naar het origineel als het wachtwoord in de tabel staat.
- c) Ze installeren malware op het systeem van de gebruiker.
- d) Ze versleutelen de communicatie tussen de gebruiker en de website.

#### □ Antwoord

b) Ze maken het mogelijk om gehashte wachtwoorden snel terug te vertalen naar het origineel als het wachtwoord in de tabel staat.

## Hoofdstuk 7: Salting en encryptie

13. Wat is het hoofddoel van 'salting' bij het hashen van wachtwoorden?

- a) Om het hash-algoritme complexer te maken.
- b) Om ervoor te zorgen dat hetzelfde wachtwoord elke keer een unieke hash krijgt, wat rainbow tables minder effectief maakt.
- c) Om de snelheid van het hashen te verhogen.
- d) Om te controleren of een wachtwoord sterk genoeg is.

#### □ Antwoord

b) Om ervoor te zorgen dat hetzelfde wachtwoord elke keer een unieke hash krijgt, wat rainbow tables minder effectief maakt.

14. Hoe wordt een gehasht wachtwoord met een 'salt' gecontroleerd bij het inloggen?

- a) Het systeem probeert de opgeslagen hash te ontsleutelen met de salt.
- b) Het ingevoerde wachtwoord wordt gehasht zonder de salt en vergeleken met de opgeslagen hash.
- c) Het ingevoerde wachtwoord wordt opnieuw gehasht samen met dezelfde opgeslagen salt, en het resultaat wordt vergeleken met de opgeslagen hash.
- d) De gebruiker wordt gevraagd om de salt handmatig in te voeren.

**OOP**

### Wat betekent OOP en hoe verschilt het van procedureel programmeren?

OOP staat voor *Objectgeoriënteerd programmeren*. In plaats van functies en variabelen apart te gebruiken, bundel je bij OOP data en gedrag in objecten. Zo kun je code beter organiseren, hergebruiken en opsplitsen in logische blokken .

### Wat is een klasse in OOP?

Een klasse is een blauwdruk of sjabloon waarin je beschrijft welke gegevens (*properties*) en functies (*methods*) een object moet hebben .

### Wat is een object?

Een object is een concreet exemplaar van een klasse, gemaakt met `new`. Je kunt meerdere objecten maken van dezelfde klasse, elk met eigen waarden .

### Hoe noem je in OOP een variabele en een functie binnen een klasse?

In OOP noem je variabelen **properties**, en functies **methods**

### Wat is encapsulation?

Encapsulation betekent dat je de data (properties) van een object beschermt. Je maakt gegevens vaak `private` en gebruikt methods om ze gecontroleerd te lezen of aanpassen .

### Wat is het verschil tussen public en private properties/methods?

- **public**: toegankelijk en aanpasbaar van buiten de class.
- **private**: alleen toegankelijk binnen de class zelf. Dit beschermt de interne gegevens.

### Waarom is OOP handig bij grote projecten?

Omdat je code makkelijker kunt organiseren in logische blokken (objecten), hergebruiken, uitbreiden en onderhouden. Daardoor is je programma stabiel en schaalbaarder .

## Waarvoor gebruik je '\$this' -> in PHP?

`$this->` verwijst naar een property of een object uit **dit** object. Met dit object wordt bedoeld het object waar `$this->` in staat.

## Opdracht

Maak de kennis-check.

## Inleveren

Aan het einde van de kennis-check ontvang je een certificaat. Maak een schermafdruck en lever deze in.