

Kennis Check Blok 4

Prompt Engineering 1

1. Wat is prompt engineering?

Prompt engineering is het slim formuleren van je vraag of opdracht aan een AI, zodat je het best mogelijke antwoord krijgt. Je stuurt de AI dus als het ware met jouw woorden.

2. Waarom is het belangrijk om duidelijk te zijn in je prompt?

Als je vaag bent, weet de AI niet goed wat je bedoelt en krijg je soms een verkeerd of vaag antwoord. Hoe duidelijker je bent, hoe beter het antwoord.

3. Wat is het verschil tussen een open en een gesloten vraag aan een AI?

Een open vraag laat ruimte voor uitleg of een mening ("Wat vind jij van...?"). Een gesloten vraag stuurt de AI naar een specifiek antwoord ("Geef 3 voorbeelden van...").

4. Wat betekent "few-shot prompting"?

Je geeft de AI een paar voorbeelden voordat je jouw echte vraag stelt. Zo leert de AI wat voor soort antwoord jij verwacht.

5. Wat doet "chain-of-thought prompting"?

Bij chain-of-thought laat je de AI stap voor stap nadenken. Dit helpt bij moeilijke vragen of rekenvragen, net zoals jij zelf in stappen zou werken.

6. Waarom is het handig om rollen te geven in een prompt (zoals "Jij bent een docent")?

De AI weet dan beter vanuit welk perspectief hij moet antwoorden. Het verandert de toon en inhoud van het antwoord zodat het past bij de rol.

7. Wat is prompt injection?

Dat is als iemand expres een opdracht in de prompt stopt om de AI iets fout of ongewenst te laten doen. Een soort trucje om de AI te misleiden.

8. Wat is een reden om AI je prompt te laten verbeteren?

De AI kan jouw vraag herschrijven zodat je een duidelijker en beter antwoord krijgt. Vooral handig als je zelf niet goed weet hoe je iets moet vragen.

9. Waarom helpt het om de AI een format of structuur te geven in je prompt?

Dan weet de AI precies hoe jij het antwoord wilt hebben, zoals in een tabel, stappenplan of tekst. Dat maakt het resultaat veel bruikbaar.

10. Waarom is oefenen met prompt engineering belangrijk als programmeur of ontwerper?

Omdat je dan beter leert samenwerken met AI. Je krijgt sneller goede antwoorden en kunt het gebruiken bij ontwerpen, schrijven of coderen.

PHP Intro

1. Wat is het verschil tussen front-end en back-end?

Front-end is wat je op het scherm ziet (zoals knoppen en kleuren), back-end is wat er op de server gebeurt (zoals berekeningen of gegevens opslaan).

2. Waarom is het handig om een duidelijk verschil te maken tussen front-end en back-end?

Omdat ze allebei iets anders doen.

Front-end wordt door de browser uitgevoerd en heeft vooral te maken met hoe een **website er uit ziet**.

Back-end wordt door de **server** (of XAMPP) uitgevoerd en heeft vooral te maken met **gegevens** en het **verwerken** van gegevens.

Later zullen we zien dat een database en alles wat daar mee te maken heeft op de backend draait.

3. Wat gebeurt er als je een formulier invult en op verzenden klikt?

De front-end stuurt je gegevens naar de back-end. Daar verwerkt de **server** die gegevens, bijvoorbeeld om ze op te slaan of een antwoord terug te geven.

4. Waarom kun je zeggen dat een webwinkel zowel front-end als back-end gebruikt?

De front-end laat producten en knoppen zien. De back-end checkt of iets op voorraad is en verwerkt je bestelling.

5. Wat doet een server eigenlijk?

Een server voert taken uit die je vanaf een browser vraagt, zoals gegevens opslaan, iets berekenen of een pagina terugsturen.

6. Waarom gebruiken we PHP aan de back-end?

Omdat PHP op de server draait en dingen kan doen zoals inloggen controleren, iets berekenen of data uit een database halen.

7. Hoe kun je weten of iets front-end is?

Als je het kunt zien of aanklikken in de browser (zoals tekst, kleuren of knoppen), dan is het front-end.

8. Wat is de functie van HTML in een website?

HTML bepaalt de structuur van een pagina: wat er op staat, zoals koppen, tekst en plaatjes.

9. Wat is de rol van CSS in de front-end?

CSS zorgt dat de HTML van een website er mooi uit ziet. Het bepaalt kleuren, lettertypes en waar iets op de pagina staat.

10. Waarom heb je een programma zoals XAMPP nodig voor PHP?

Omdat PHP op een server draait en je die server op je eigen computer moet nabootsen. XAMPP doet dat voor je.

PHP 1

1. Wat is het verschil tussen GET en POST in een formulier?

GET laat de ingevulde gegevens in de URL zien, POST doet dat niet. POST is dus beter voor privacy, bijvoorbeeld bij wachtwoorden. GET is weer handiger als je ontwikkelen bent omdat je (op de URL) precies kan zien welke variabelen en waarden er worden verstuurd.

2. Waarom gebruik je een formulier met `method="post"` als je een wachtwoord verstuurt?

Omdat het wachtwoord dan niet zichtbaar is in de URL. Dat is veiliger.

3. Wat doet het stukje PHP-code `$_POST["adres"]`?

Het haalt de waarde op van het invoerveld met de naam "adres" uit het formulier dat je hebt verzonden met POST.

```
<input name="adres">
```

4. Wat is het verschil tussen `include` en `require` in PHP?

Ze voegen allebei een PHP-bestand toe.

Bij include gaat je script verder als het bestand er niet is. Bij require stopt je script met een foutmelding.

5. Wat is een array in PHP?

Een array is een lijstje van dingen. Bijvoorbeeld een lijst van namen of cijfers.

6. Wat is het verschil tussen een gewone array en een associatieve array?

Een gewone array gebruikt nummers als sleutels (0, 1, 2...). Een associatieve array gebruikt woorden als sleutels, zoals "naam" of "leeftijd".

7. Wat doet een foreach-loop in PHP?

Een foreach-loop herhaalt een stukje code voor elk onderdeel van een array. Zo kun je makkelijk alle items uit een lijst laten zien.

8. Waarom is het handig om een functie te maken in PHP?

Een functie is een herbruikbaar stukje code. Je kunt het vaker gebruiken zonder alles opnieuw te typen.

9. Wat is het voordeel van een functie met een parameter?

Dan kun je er iets aan meegeven, zoals een naam, zodat de functie elke keer iets anders kan doen.

```
function mijnFunctie(naam) { .....
```

10. Wat doet session_start() in PHP?

Het start of opent een sessie. Daardoor kun je informatie (zoals een inlogstatus) bewaren terwijl je naar andere pagina's gaat.

