Programmeerbasis

Software developer

ICT Lyceum, Deltion College, Zwolle

31 augustus 2022

Auteurs: Docenten ICT-Lyceum, Deltion College.

Deze opdracht is geschreven met de volgende tools: LaTeX, Sublime text, Visual Studio Code, TeXstudio, IntelliJ IDEA Ultimate, sourcetree, gitkraken, vi en Git. Allemaal op Fedora Linux, Apple MacOS en Microsoft Windows.



Dit werk valt onder een Creative Commons Naamsvermelding-NietCommercieel 4.0 internationaal-licentie. http://creativecommons.org/licenses/by-nc/4.0/

Inhoudsopgave

1	Introductie	4
2	Backlog	6
3	Basiselementen	7
4	Echo	8
5	Variabelen	11
6	Rekenen met variabelen	14
7	Userinvoer	16
8	While lus	18
9	For lus	21
10	If statement	24
11	Functie	26

12 Arrays 29

13 Verdiepen in programmeren 31

14 Eindtaak : Een Quiz 32

15 Afronding 34

16 Rubric Programmeerbasis 35

Hoofdstuk 1

Introductie

Programmeren is het schrijven van code zodat computers bepaalde taken uitvoeren. Deze taken zijn bijvoorbeeld:

- het opslaan van een document;
- het sorteren van een lijstje met namen;
- het uitvoeren van een berekening;
- het printen van een document;
- het afspelen van muziek;
- het streamen van een film;
- het uitvoeren van een pinbetaling;
- het bijhouden van een uitslag in een game;
- het opbouwen van een scenery in een game;
- etc.

Om deze taken uit te voeren is een programma nodig. [1] Het streamen van een film doe je bijvoorbeeld met de YouTube of NetFlix app. Deze bedrijven hebben daarvoor een app ontwikkeld waarmee je de gewenste functionaliteit daadwerkelijk kunt gebruiken.

[1] Andere name voor programma zijn app, applicatie, tool

Voor het schrijven van een app gebruik je een programmeertaal. Door de jaren heen zijn er veel programmeertalen ontwikkeld. Sommige talen zijn generiek en kun je voor veel situaties gebruiken, andere talen zijn heel specifiek en gebruik je alleen in hele specifieke situaties. De taal C is bijvoorbeeld een generieke taal waarmee je alle gewenste apps kunt schrijven. De taal APL is daarentegen een hele specifieke taal waarmee je alleen wiskundige apps schrijft.

Een mooi overzicht van programmeertalen vind je op https://en.wikipedia.org/wiki/List_of_programming_languages. Schrik niet van de lengte van de lijst. Sommige talen worden tegenwoordig niet meer gebruikt, terwijl andere programmeertalen nog steeds populair zijn. De populariteit van programeertalen verandert per jaar.

Talen die nu populair zijn, waren dat een paar jaar geleden niet. "Python is met 28,38% verreweg de meest populaire taal wereldwijd, zo blijkt uit de PYPL index. Java (17,5%) en JavaScript (9,29%) volgen op gepaste afstand."(https://pypl.github.io/PYPL.html)

Als software developer zul jij je moeten instellen op deze veranderende populariteit. Als software developer ben jij nooit uitgeleerd. Jij zult je elke dag moeten verdiepen in nieuwe technieken en nieuwe talen.

Let Op: De programma's in deze opdracht worden op de commando regel (cli[2]) uitgevoerd. Je hoeft deze programma's niet via de browser op te starten.

[2]Command line interface

Hoofdstuk 2

Backlog

- Als software developer wil ik de basisstructuren van programmeertalen kennen zodat ik code kan schrijven.
- Als software developer wil ik de basisstructuren van programmeertalen kennen zodat ik code van andere developers kan lezen.
- Als software developer wil ik de basisstructuren van programmeertalen kennen zodat ik code op het internet kan beoordelen op efficiëntie en veiligheid.

Hoofdstuk 3

Basiselementen

De meeste talen hebben een aantal overeenkomsten. Dit zijn bijvoorbeelde de basiselementen. Het goed leren van deze basiselementen is noodzakelijk om een goede software developer te worden.

Om de basiselementen goed te leren gebruik je het liefste een echte programmeertaal. Wij gebruiken in deze opdracht de PHP taal om de basiselementen te leren. Je kunt PHP aparte installeren maar ook als onderdeel van XAMPP . Als het goed is heb je zowel PHP als XAMPP geïnstalleerd in de opdracht-Ontwikkelomgeving. Mocht je PHP nog niet geïnstalleerd hebben, installeer deze dan voordat je verder gaat met deze opdracht.

De volgende basiselementen worden in de komende pagina's behandeld.

- `echo` of `print`
- variabelen
- user invoer
- `while`
- `if`
- `for`
- `function`

Hoofdstuk 4

Echo

Het eerste statement dat jij leert is het `echo` statement. Met dit statement zet je tekst op het scherm van de gebruiker. Sommige talen kennen geen `echo` maar hebben in plaats daarvan het `print` statement. Er zijn zelfs talen die beide statements ondersteunen.

Let Op: De programma's in deze opdracht worden op de commando regel (cli[1]) uitgevoerd. Je hoeft deze programma's niet via de browser op te starten.

In listing 4.1 zie je het echo statement van PHP. Dit simpele programmaatje zet de tekst Hello World op jouw beeldscherm.

Listing 4.1: HelloWorld.php

```
1 <?php
2
3 echo "Hello World\n";
4
5 ?>
```

In de volgende taak ga je het programma van listing 4.1 uitvoeren.

Taak : Schrijf jouw eigen HelloWorld programma

[1]Command line interface

1. Maak in jouw user folder een folder met de naam `phpapps`.
2. Plaats deze folder onder git controle met `git init`.
3. Plaats met een code editor de code van listing 4.1 in het bestand `HelloWorld.php`.
4. Plaats `HelloWorld.php` onder `git` controle dmv `git add` en `git commit`. Controleer met `git log` of dit goed is gegaan.
5. Open een command prompt aka. of cmd of powershell en navigeer naar de `phpapps` folder;
6. Voer `HelloWorld.php` uit met het volgende commando:
   `c:\...\> php HelloWorld.php`
   Let op: Start HelloWorld.php niet via de browser maar via de cli.
7. Werkt het programma? Pas het programma aan zodat het wel werkt en commit `HelloWorld.php` in `Git`.
8. Probeer voor je zelf te verklaren wat er in `HelloWorld.php` gebeurt.

In listing 4.2 voeren we meerdere echo statements uit. Wat denk je dat het resultaat is?

Listing 4.2: HelloWorld.php, versie 2

```
1 <?php
2
3 echo "Hello World\n";
4 echo "Hello World\n";
5 echo "Hello World\n";
6 echo "Hello World\n";
7
8 ?>
```

In de volgende taak voer je de code van listing 4.2 uit.

Taak : Pas HelloWorld aan

1. Pas met een code editor `HelloWorld.php` aan zodat het overeenkomt met de code in 4.2.

2. Voer het bestand `HelloWorld.php` uit.
3. Wat is het verschil met de vorige keer dat je `HelloWorld.php` uitvoerde?
4. Commit het `HelloWorld.php` in `Git`.

Hoofdstuk 5

Variabelen

Een variabele wordt gebruikt om dingen op te slaan. Een goed voorbeeld is een score in een spel. In je spel wil je uiteindelijk zien wat je score is. Die score moet je bijhouden. De score slaan we op in een 'variabele'. We noemen het een variabele omdat de waarde variabel is, de waarde kan veranderen. Als je een punt scoort verandert je score met 1 punt. **lerenprogrammeren-variabele**

Taak : Pas HelloWorld opnieuw aan

1. pas met een code editor het `HelloWorld.php` aan zodat elke regel `Hello World!` laat zien; dus met een uitroepteken achter World;
2. voer `HelloWorld.php` opnieuw uit en controleer de uitvoer;
3. commit `HelloWorld.php` in `Git`;

Omdat de tekst vijf keer in het programma stond, moest de tekst vijf keer aangepast worden. Het was waarschijnlijk vervelend om vijf keer een uitroepteken in de tekst te moeten zetten.

Door een variabele te gebruiken wordt het makkelijker om de tekst in dit programma aan te passen.

In listing 5.1 gebruiken we een variabele om de tekst maar één keer in de code te schrijven.

Listing 5.1: HelloWorld.php, versie 3

```
1  <?php
2
3  $tekst="Hello World!\n";
4
5  echo $tekst;
6  echo $tekst;
7  echo $tekst;
8  echo $tekst;
9  echo $tekst;
10
11 ?>
```

Taak : Pas HelloWorld aan met variabelen

In deze taak voer je de code van listing 5.1 uit.

1. pas `HelloWorld.php` aan volgens listing 5.1;
2. run `HelloWorld.php`;
3. `commit` `HelloWorld.php` in `Git`;
4. pas de inhoud van de `tekst`-variabele aan door een komma te plaatsen tussen Hello en World zoals hieronder weergegeven:
   `$tekst="Hello,World!";`
5. voer `HelloWorld.php` uit;
6. hebben alle regels in de uitvoer een komma tussen Hello en World?
7. `commit` `HelloWorld.php` in `Git`;
8. probeer voor jezelf te verwoorden wat er in `HelloWorld.php` gebeurt;

Taak : Wijzig HelloWorld met eigen ideeën

Nu je weet hoe `HelloWorld.php` werkt, kun je zelf wijzigigen aanbrengen in het programma en testen wat de resultaten zijn van jouw wijzigingen.

1. pas `HelloWorld.php` aan door

- de variabele-inhoud naar eigen inzicht te wijzigen;
- variabelen en `echo`-regels naar eigen inzicht toe te voegen;
- commit je wijzigingen in `Git`;

Hoofdstuk 6

Rekenen met variabelen

Getallen en waarden worden vaak in variabelen opgeslagen. Meestal moet je met deze waarden berekeningen uitvoeren, zoals een totaal of het gemiddelde bepalen.

Listing 6.1: reken.php

```
1  <?php
2  $getal1=15;
3  $getal2=3;
4
5  $som=$getal1 + $getal2;
6  echo "$getal1 + $getal2 = ".$som;
7  echo "\n";
8
9  $verschil=$getal1 - $getal2;
10 echo "$getal1 - $getal2 = ".$verschil;
11 echo "\n";
12
13 $product=$getal1 * $getal2;
14 echo "$getal1 * $getal2 = ".$product;
15 echo "\n";
16
17 $quotient=$getal1 / $getal2;
18 echo "$getal1 / $getal2 = ".$quotient;
19 echo "\n";
20
21 $rest=$getal1 % $getal2;
```

```
22 echo "$getal1 % $getal2 = ".$rest;
23 echo "\n";
24 ?>
```

Taak : Rekenen in PHP

In deze taak maak je kennis met het rekenen in PHP.

1. Maak het bestand `Reken.php` met de inhoud van listing 6.1;
2. Voer `Reken.php` uit en controleer de uitvoer;
3. Commit `Reken.php` in `Git`;
4. Pas de getallen in `$getal1` en `$getal2` aan en voer `Reken.php` opnieuw uit;
5. Klopt de uitvoer van `Reken.php`?
6. `Commit`!!

Hoofdstuk 7

Userinvoer

Op dit moment werken onze programma's met vaste gegevens. Deze gegevens staan in de source code van het programma en daar worden dan bewerkingen op gedaan. Tot nu toe kunnen onze programma's nog niet werken met gegevens van verschillende gebruikers.

Om een programma met andere gegevens te laten werken moet de source code van het programma aangepast worden. Dit is natuurlijk onhandig voor gebruikers die niet kunnen programmeren. Ook wordt de kans op fouten groter als een gebruiker steeds de source code moet aanpassen.

Het programma `leeftijd.php` is bijvoorbeeld niet bruikbaar voor iemand die niet jouw leeftijd heeft. Als ik bijvoorbeeld 30 jaar ouder ben dan jij, dan kloppen de berekende jaren. We hebben een manier nodig om gegevens van een gebruiker te vragen en deze in het programma te gebruiken. Voor PHP is dit het `readline` commando.

Door middel van het `readline` commando kun jij informatie aan de gebruiker gevraagd via het toetsenbord.

In listing 7.1 vragen we de gebruiker om een woord. Dit woord slaan we op in de variabele invoer. We gebruiker deze variabele om het woord later op het beeldscherm te tonen.

Listing 7.1: invoer.php

```
1 <?php
2
3 $invoer = readline("Geef een woord : ");
```

```
4
5 echo "Jij gaf het volgende woord: $invoer";
6 echo "\n";
7 ?>
```

Taak : Vraag invoer van een gebruiker

1. Maak het bestand `invoer.php` en plaats daarin de code van listing 7.1;
2. Voer `invoer.php` meerdere keren uit en kijk wat er gebeurt;

Hoofdstuk 8

While lus

'A While Loop is used to repeat a specific block of code an unknown number of times, until a condition is met.' **utahedu-while**

In listing 8.1 gebruiken we een while lus om 10 keer een regel met tekst te tonen.

Listing 8.1: whilelus.php

```
1  <?php
2
3    $teller=1;
4    $maximum=10;
5
6    echo "Teller is voor de while lus ".$teller."\n";
7
8    while ( $teller <= $maximum ) {
9       echo "Teller is nu ".$teller."\n";
10      $teller++;
11   }
12
13   echo "Teller is na de while lus ".$teller."\n";
14
15 ?>
```

Taak : Je eerste while lus

1. Plaats de code van listing 8.1 in het bestand `WhileLus.php`.
2. Run `WhileLus.php`.
3. `commit` in git.
4. Pas `WhileLus.php` aan zodat hij stopt zodra teller 15 is.
5. Run `WhileLus.php`.
6. `commit` in git.
7. Pas `WhileLus.php` aan zodat hij begint bij 8.
8. Run `WhileLus.php`.
9. `commit` in git.
10. Wat gebeurt er als je $teller groter maakt dan $maximum? test dit.

Kijk nog eens naar de code in listing 5.1 op pagina 12. In die code hebben we 5 maal dezelfde regel staan. Door middel van een `while`-lus kunnen we deze code ook met een lus schrijven.

Listing 8.2: whilelus.php, versie2

```
1  <?php
2
3  $teller=1;
4  $maximum=10;
5  $tekst="Hello, World!\n";
6
7  while ( $teller <= $maximum ) {
8      echo $tekst;
9
10     $teller++;
11 }
12 ?>
```

Taak : HelloWorld in een lus

1. Pas `HelloWorld.php` aan met de code van 8.2.

2. Voer `HelloWorld.php` uit en verklaar de uitvoer.

3. `commit` in git.

4. Pas de waarden voor `$teller` en `$maximum` aan en controleer wat jouw wijzigingen met de uitvoer van het programma doen.

5. `Commit` in git na iedere wijziging.

We hebben het `readline` commando al gezien. Gecombineerd met een while lus kunnen we nu een programma schrijven dat pas stopt als een gebruiker dat wil.

Listing 8.3: userstop.php

```
1  <?php
2
3  $stop = "nee";
4  $teller = 1;
5
6  while ( $stop == "nee" ) {
7
8          echo "Dit is run $teller\n";
9
10         $stop = readline("Wil je stoppen [ja/nee] : ");
11         $teller++;
12 }
13
14 ?>
```

Taak : Lus met userinvoer

1. maak het bestand `UserStop.php` en plaats de code van listing 8.3 in dit bestand;

2. voer het programma `UserStop.php` uit;

3. `commit`!!

Hoofdstuk 9

For lus

'A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.' **tutorialspoint-for**

Kijk nog eens naar de code van listing van 8.1 op pagina 18. We kunnen deze `while`-lus ook herschrijven in een `for`-lus.

De herschreven code ziet er alsvolgt uit:

Listing 9.1: forlus.php

```
1  <?php
2
3  $begin=1;
4  $maximum=10;
5  $tekst="Hello, World!\n";
6
7  for ($teller=$begin;$teller<=$maximum;$teller++) {
8      echo $tekst;
9  }
10 ?>
```

Taak : Je eerste for lus

1. Plaats de code van listing 9.1 in het bestand `ForLus.php`;

2. voer `ForLus.php` uit;
3. verklaar de uitvoer van `ForLus.php`;
4. pas de waarden in `ForLus.php` aan en kijk wat het effect is;
5. `commit` in git;

Verschil tussen For en While

Het belangrijke verschil tussen een `while` en `for` lus is dat een `for` lus een zogenaamde tellende lus is. Je gebruikt een `for` lus voor code blokken die je een aantal keren wilt herhalen waarbij aantal een getal is. Dit aantal is bekend of kan berekend worden aan het begin van de for-lus.

Een `while` lus is een lus die uitgevoerd wordt zolang een bepaalde conditie waar is. Deze conditie kan een getal zijn maar dat hoeft niet. Het is meestal van te voren niet bekend hoe vaak een `while`-lus uitgevoerd zal worden. While condities kunnen ook zijn:

- Bevat een variabele een bepaald woord.
- Bevat variabeleA een waarde groter dan variabeleB.
- Bevat variabeleC dezelfde waarde dan variabaleD.
- Voor elke regel in een bestand.
- Voor elke packet ontvangen van het netwerk.
- Zolang de ingelezen getallen opgeteld niet groter zijn dan 1024.
- enzovoort.

Door `readline` te gebruiken kun je van een gebruiker invoer vragen en daarmee rekenen.

In listing 9.2 vragen we de gebruiker om een gewenste tafel. Daarna laten we de gewenste tafel zien.

Listing 9.2: tafel.php

```
1  <?php
2
3  $tafel=readline("Welke tafel wil je zien? ");
4  $begin=1;
5  $eind=10;
6
7  echo "\n\nTafel van $tafel:\n";
8
9  for ($teller=$begin;$teller<=$eind;$teller++) {
10        product=$teller*$tafel;
11        echo $teller." x ".$tafel." = ".$product."\n";
12 }
13
14 ?>
```

Taak : Tafel

1. plaats de code van listing 9.2 in het bestand `Tafel.php`;
2. run `Tafel.php` enkele keren met verschillende invoer;
3. `commit` in git;

Hoofdstuk 10

If statement

”The ability to control the flow of your program, letting it make decisions on what code to execute, is valuable to the programmer. The if statement allows you to control if a program enters a section of code or not based on whether a given condition is true or false. One of the important functions of the if statement is that it allows the program to select an action based upon the user’s input. For example, by using an if statement to check a user-entered password, your program can decide whether a user is allowed access to the program.”**cprogramming-if**

Met het `if` statement worden beslissingen in programma’s genomen. Afhankelijk van een situatie wordt er wel of geen code uitgevoerd.

In listing 10.1 vragen we de gebruiker om zijn leeftijd. Afhankelijk van zijn leeftijd geven we verschillende uitvoer.

Listing 10.1: ifkeuze.php

```
1  <?php
2
3  $naam = readline("Geef jouw naam : ");
4  $leeftijd = readline("Geef jouw leeftijd : ");
5
6  echo "Hallo $naam \n";
7
8  if ( $leeftijd < 22 ) {
9         echo "Jij bent jonger dan 22 jaar.\n";
10 }
11 if ( $leeftijd > 22 ) {
```

```
12          echo "Jij bent ouder dan 22 jaar.\n";
13  }
14  if ( $leeftijd == 22 ) {
15          echo "Jij bent precies 22 jaar.\n";
16  }
```

Taak

1. maak het bestand `IfKeuze.php` en plaats daarin de code uit listing 10.1;
2. voer het programma meerdere keren uit en voer steeds andere waarden in;
3. pas het programma aan zodat het controleert of jij jonger dan, ouder dan of precies 25 jaar oud bent;
4. pas het programma aan zodat het controleert of jij jonger dan, ouder dan of precies 21 jaar oud bent;
5. heb jij bij de vorige stap drie maal het getal 21 in de code geplaatst? Zorg ervoor dmv. variabelen voor dat het getal 21 maar éénmaal in de code staat;
6. `commit` jouw wijzigingen in git;

Hoofdstuk 11

Functie

'A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.' **tutorialspoint-function**

Een functie in een programma geeft ons de mogelijkheid om dezelfde code op meerdere plaatsen uit te voeren zonder die code steeds te herhalen. Een functie is een gedefinieerde procedure die je vanaf verschillende locaties in jouw programma aanroept.

Hoewel de functionaliteit van een functie in veel talen voorkomt, hebben verschillende talen andere namen bedacht voor een functie. Zo betekenen method, sub-routine, procedure hetzelfde als functie.

In listing 11.1 maken we een functie `func`. Deze functie roepen we op verschillende plaatsen aan.

Listing 11.1: functie.php

```
1 <?php
2
3 function func() {
4    echo "+ Deze regel is in func() uitgevoerd.\n";
5 }
6
7 echo "-- Deze code is voor func() uitgevoerd.\n";
8
9 func();
```

```
10
11 echo "== Deze regel is tussen twee func() uitgevoerd.\n";
12
13 func();
14
15 echo "-- Deze regel is na func() uitgevoerd.\n";
16
17 ?>
```

Taak : Je eerste functie

1. Maak het bestand `Functie.php` en neem daarin de code van listing 11.1 over.
2. Voer het programma `Functie.php` uit.
3. `Commit` in git.
4. Onderzoek hoe het programma werkt.

Een functie kan ook aangeroepen worden vanuit één van de basiselementen, `while`, `for` en `if`.

Listing 11.2: functiefor.php

```
1  <?php
2
3  function func() {
4          echo "+ Deze code is in func() uitgevoerd.\n";
5  }
6
7  echo "- Deze code is voor de for-lus uitgevoerd.\n";
8
9  for ($counter=1;$counter<=3;$counter++) {
10         echo "= Deze code is in de for-lus\n";
11         echo " maar voor func() uitgevoerd.\n";
12         func();
13         echo "= Deze code is in de for-lus\n";
14         echo " maar na func() uitgevoerd.\n";
15 }
```

```
16
17 echo "- Deze code is na de for-lus uitgevoerd.\n";
18
19 ?>
```

Taak : Een functieaanroep in een basiselement

1. Sla de listing van 11.2 op in het bestand `FunctieFor.php`.
2. Voer `FunctieFor.php` uit.
3. `commit`.
4. Probeer voor jezelf te verklaren hoe dit programma werkt.

Hoofdstuk 12

Arrays

Bij de meeste programmeertalen is het mogelijk om array's gebruikt. Om goed te kunnen programmeren is het dus belangrijk om te weten wat een array is en hoe je deze kunt gebruiken. Een array kun je eigenlijk zien als een variabele die meerdere waardes kan bevatten, een lijst van waardes. Deze variabele bestaat dan uit meerdere rijen (entries) die allemaal een waarde hebben. Het declareren/aanmaken van een array gaat bijna hetzelfde als het declareren van een variabele. Bijvoorbeeld:

Listing 12.1: PHP Array declaratie

```
1 $weekdagen = array('Zondag','Maandag','Dinsdag','Woensdag','
    ↪ Donderdag','Vrijdag','Zaterdag');
```

Elke waarde in de array heeft een nummer, de zogenaamde index. Het eerste item in een array heeft nummer 0, het tweede nummer 1, enz. Om uit bovenstaande array de zondag op het scherm te tonen kan het volgende commando gebruikt worden.

Listing 12.2: PHP Array element opvragen (1)

```
1 echo $weekdagen[0];
```

Om uit bovenstaande array de Zaterdag op het scherm te tonen kan het volgende commando gebruikt worden:

Listing 12.3: PHP array element opvragen (2)

```
1 echo $weekdagen[6];
```

Er zijn verschillende methodes die gebruikt kunnen worden om met arrays te werken. Zo is het bijvoorbeeld mogelijk om arrays te sorteren, items aan een array toe te voegen of te verwijderen. Bestudeer de mogelijkheden van arrays op de volgende pagina: https://www.w3schools.com/php/php_ref_array.asp (**w3schools-array**).

Hoofdstuk 13

Verdiepen in programmeren

In dit hoofdstuk heb je kennis gemaakt met programmeren. Zoals je wel zult begrijpen is dit alleen maar een basis. De komende weken, maanden en jaren zul je nog veel meer leren over programmeren en coderen.

Taak : Verdiepen in programmeren

1. Navigeer naar https://www.codecademy.com/learn/learn-how-to-code
2. Start de gratis 'Learn How to Code' cursus.
3. Pas de kennis die je in deze cursus op doet toe in jouw eigen programma's.

Hoofdstuk 14

Eindtaak : Een Quiz

Laten we eens kijken of jij het geleerde in praktijk kunt brengen door het programmeren van een quiz programma.

1. Maak het programma `Quiz.php` dat voldoet aan de volgende requirements:
    - Het programma kiest een willekeurig getal tussen de 1 en 100;
        - De volgende PHP-code genereert een willekeurig getal tussen 1 en 100 en slaat deze op in de variabele $random;
          `$random = rand(1,100)`
    - De gebruiker wordt gevraagd dit willekeurige getal te raden;
    - Als gebruiker een verkeerd antwoord geeft, dan mag gebruiker nogmaals raden;
    - Als de gebruiker een verkeerd antwoord geeft, dan geeft het programma aan of het antwoord hoger of lager is dan het willekeurige getal;
    - Het programma stop als de gebruiker het juiste getal geraden heeft;
    - Als gebruiker het juiste getal geraden heeft, dan geeft het programma aan in hoeveel beurten de gebruiker het juiste getal geraden heeft;
    - Het programma stopt als de gebruiker een getal kleiner dan 1 geeft;
    - Het programma stopt als de gebruiker een getal groter dan 100 geeft;
    - Het programma stopt als de gebruiker het woord stop typt.
    - De programma-uitvoer is logisch en duidelijk.
    - De programma-code is makkelijk leesbaar en gedocumenteerd.

- De programma-code is onderhoudsvrienlijke.

2. Test je programma en controleer of alle gevraagde requirements erin zitten.
3. `Push` `Quiz.php` naar GitHub.

Hoofdstuk 15

Afronding

1. Bekijk of jouw werk volgens de rubric een voldoende of goed scoort.
   Overleg met een klasgenoot of één van de docenten als je twijfelt of jouw werk een voldoende of goed haalt.
2. Bespreek een review moment bij een van de docenten.
3. Deel de code die je met de docenten wilt bespreken via GitHub.

4. Bereid goed je voor op het review moment:
   - Ken jouw code goed.
     Weet hoe jouw programma in elkaar zit. Weet waar de functionliteiten gecodeerd zijn.
     - Wij docenten gaan ervan uit dat je zonder hapering je eigen code kunt uitleggen.
     - Wij docenten gaan ervan uit dat je zonder aarzeling een wijziging kunt doorvoeren in jouw programma.
   - Verwacht vragen over jouw code en de keuzes die je gemaakt hebt.
     Een antwoord in de trant van ”Ja, dat weet ik eigenlijk niet.ïs iets dat wij niet graag horen.
   - Leer de theorie uit je hoofd;
     - Wij docenten gaan ervan uit dat je zonder hapering de gevraagde theorie kunt uitleggen;
     - Wij docenten gaan ervan uit dat je zonder hapering vragen over de theorie kunt beantwoorden;

Hoofdstuk 16

Rubric Programmeerbasis

Opdracht

G=Goed — V=Voldoende — M=Matig — O=Onvoldoende				
	G	*V*	*M*	*O*
Ik heb de voorbeeldcodes overgenomen, en de bijbehorende taken uitgevoerd				
Ik heb git gebruikt tijdens deze opdracht				
Ik heb de code die ik wil bespreken met de docenten gedeeld via GitHub				

Kennis

G=Goed — V=Voldoende — M=Matig — O=Onvoldoende				
	G	*V*	*M*	*O*
Ik kan het `echo` statement uitleggen				
Ik kan het `print` statement uitleggen				
Ik kan het begrip **variabele** uitleggen				
Ik kan het begrip **array** uitleggen				
Ik kan het `readline` statement uitleggen				
Ik kan het `while` statement uitleggen				
Ik kan het `if` statement uitleggen				
Ik kan het `for` statement uitleggen				

Ik kan het begrip **function** uitleggen				

Programmeertalen Welke programmeertalen ken jij naast PHP, Javascript, Html en Css?			
Goed	*Voldoende*	*Matig*	*Onvoldoende*
Ik kan meer dan 15 programmertalen noemen	Ik kan meer dan 10 programmeertalen noemen	Ik kan meer dan 5 programmeertalen noemen	Ik kan geen programmeertalen noemen

Quiz.php

Codereview Kun jij de werking van het programma uitleggen?			
Goed	*Voldoende*	*Matig*	*Onvoldoende*
Ik kan werking van broncode foutloos en zonder hapering uitleggen	Ik moet soms (niet hinderlijk) nadenken voordat ik broncode kan uitleggen	Ik moet veel nadenken of antwoorden opzoeken voordat ik broncode kan uitleggen	Ik kan broncode niet uitleggen **of** het programma werkt niet (foutloos)
Ik kan laten zien dat het programma variabelen gebruikt	Ik kan met enige happeringen laten zien dat het programma variabelen gebruikt	Ik kan met enige hulp laten zien dat het programma variabelen gebruikt	Ik kan niet laten zien dat het programma variabelen gebruikt **of** het programma werkt niet (foutloos)
Ik kan, wanneer gevraagd, zonder aarzeling een wijziging doorvoeren in mijn programma	Ik kan, wanneer gevraagd, na enige nadenken een wijziging doorvoeren in mijn programma	Ik kan, wanneer gevraagd, na lang nadenken een wijziging doorvoeren in mijn programma	Ik kan, wanneer gevraagd, geen wijziging doorvoeren in mijn programma

G=Goed — V=Voldoende — M=Matig — O=Onvoldoende				
	G	*V*	*M*	*O*

Mijn programma kiest een willekeurig getal				
Mijn programma stopt wanneer getal geraden is				
Mijn programma stopt bij invoer kleiner dan 1				
Mijn programma stopt bij invoer groter dan 100				
Mijn programma stopt bij invoer van het woord *stop*				
Mijn programma geeft aan in hoeveel beurten het getal geraden is				
Bij een fout antwoord geeft mijn programma aan of het antwoord hoger of lager is dan het willekeurige getal				
Mijn programma is gebruikersvriendelijk				
Mijn programma is uitvoerig gedocumenteerd				

Index

`echo`, 7, 8
`for`, 7, 21
`function`, 7
`git add`, 9
`git commit`, 9
`git log`, 9
`git push`, 33
`if`, 7, 24
`print`, 7, 8
`rand()`, 32
`readline`, 16
`while`, 7, 18

APL, 5
App, 4
Applicatie, 4
Array, 29
Array index, 29

Basiselementen, 7
Berekening, 4

C, 5
Cli, 5, 8
Codecademy, 31

Eindtaak Quiz, 32

Film, 4
Functie, 26
Function, 26

Git, 9

Method, 26
Muziek, 4

NetFlix, 4

Ontwikkelomgeving, 7

PHP, 7, 16
Pinbetaling, 4
Procedure, 26
Programma, 4
Programmeertaal, 5

Quiz, 32

Random getal, 32

Scenery, 4
Sorteren, 4
Sub-routine, 26

Tool, 4

User invoer, 7
Userinvoer, 16

Variabele, 7, 11

Willekeurig getal, 32

XAMPP, 7

YouTube, 4