

Les 2 - Programmeertalen

[« Vorige les](#)[» Volgende les](#)

Huiswerk

Programmeren voor Beginners - Les 2

Resultaat: 8,0

[Bekijk huiswerk](#)

Je hebt 1 vraagmoment. 1 moment over.

[Ga naar stel een vraag](#)

Inhoudsopgave

- 2.1** Inleiding
- 2.2** Het nut van programmeertalen
- 2.3** Programmeertaal 'vertalen'
- 2.4** Evolutie van programmeertalen
- 2.5** Talen en hun toepassingen
- 2.6** Samenvatting
- 2.7** Vraagmoment
- 2.8** Oefenopgaven
- 2.9** Huiswerkopgaven

2.1 Inleiding

Na het doornemen van de vorige les, weet u wat programmeren is: het schrijven van een programma dat een computer kan uitvoeren. Dit programma wordt geschreven in een programmeertaal.



Een programma wordt geschreven in programmeertaal.



Bij het beschrijven van de geschiedenis van programmeren heeft u al wat informatie over programmeertalen gekregen, maar in deze les gaan we dieper in op 'programmeertaal'. U leert wat het nut van programmeertalen is en hoe ze zijn geëvolueerd. Daarnaast besteden we aandacht aan de grammaticaregels en bouwblokken van een programmeertaal en aan de manier waarop een computer een ingevoerde code begrijpt.

Leerdoelen

Aan het einde van deze les weet u:

- het nut van programmeertalen;
- hoe de computer uw code vertaalt naar binaire code;
- het verschil tussen een *interpreter* en een *compiler*;
- hoe programmeertalen in de loop der tijd geëvolueerd zijn;
- wat *object-oriented programming* is;
- welke talen tegenwoordig populair zijn;
- de kenmerken en toepassingen van deze talen.



Enkele afbeeldingen in deze les zijn moeilijk leesbaar in de printversie van deze les. Wilt u een afbeelding uitvergroten? Bekijk dan de digitale versie van de les op Plaza. Klik op de afbeeldingen om ze te vergroten.

2.2 Het nut van programmeertalen

Computers kunnen niet zonder programma's. Een (gebruiker van een) programma vertelt de computer immers wat hij moet doen. In de vorige les leerde u dat deze instructiegegevens op de een of andere manier worden omgezet in een elektrische stroom. Daarvoor wordt het binaire systeem gebruikt, waar u in de vorige les al kennis mee maakte.

Hoewel een computer precies weet wat hij met al die nullen en enen moet doen, is programmeren in dit systeem geen haalbare kaart, zeker niet voor een beginner. We hebben daarom een tussenstap nodig:

Wens van de programmeur
Wat u wil dat de computer doet



Programmeertaal
Een soort 'tussenstap' tussen
mensentaal en computertaal

A screenshot of a code editor window titled "hallo.py". The code consists of two lines: "print("Hallo")" and an empty line below it, indicated by a cursor. The background of the window is dark blue.

Eindresultaat
De wens van de programmeur,
vertaald in nullen en enen (computertaal)



De tussenstap is het schrijven van de instructies in een programmeertaal. In dit voorbeeld is de instructie geschreven in de taal Python, maar we hadden ook een andere taal kunnen kiezen, zoals C# of JavaScript. In alle gevallen bereiken we ons doel: de computer op een gemakkelijker manier voorzien van instructies.

Als beginner kunt u aan een reeks nullen en enen niet zien wat ermee bedoeld wordt, maar u kunt zich wel een voorstelling maken bij `print("Hallo")`. Kortom: voor u als beginnend programmeur zijn programmeertalen enorm nuttig (al zijn ook gevorderde programmeurs erg blij dat ze niet hoeven te programmeren in binaire code).

2.3 Programmeertaal 'vertalen'

In de vorige paragraaf schreven we dat de code die u programmeert, een tussenstap is. Om te kunnen functioneren, zal een computer uiteindelijk een vertaalslag moeten maken: omzetten naar nullen en enen, zodat hij iets met deze code kan.

Er zijn grofweg twee methoden om dat te doen:

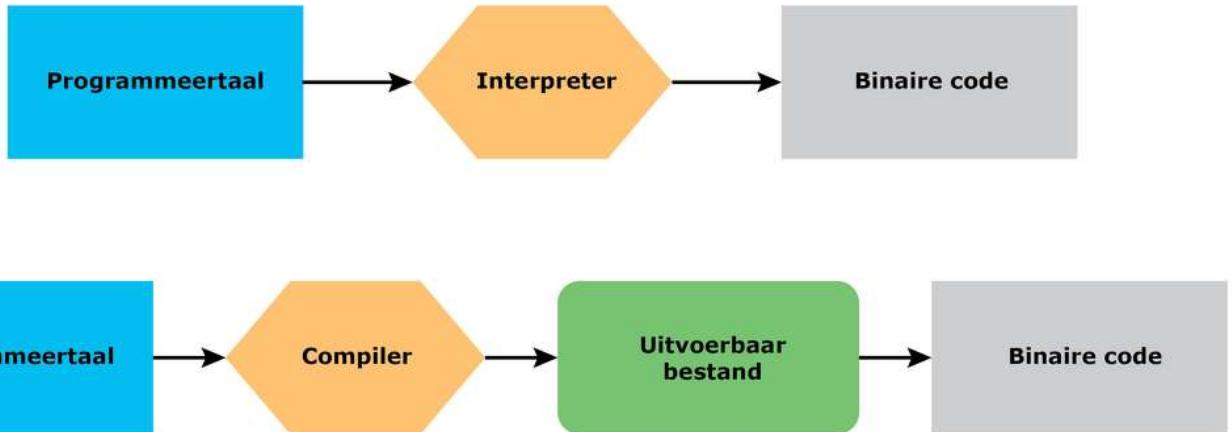
- **Interpreter**

Bij deze vertaalmethode wordt elke opdracht aan de computer commando voor commando vertaald naar de nullen en enen en direct uitgevoerd.

Het gebruik van een interpreter gaat heel snel en is overzichtelijk, maar de uiteindelijke uitvoering van het programma is trager dan bij het gebruik van een *compiler* (de tweede methode). Er moet immers gewacht worden op vertalingen, omdat deze in delen worden uitgevoerd.

- **Compiler**

Bij deze vertaalmethode wordt het programma als geheel geanalyseerd en dan pas vertaald naar de nullen en enen. Er wordt een *executable file* (uitvoerbaar bestand) aangemaakt, meestal met de extensie *.exe of *.dll. Het vertaalproces duurt daardoor langer, maar het levert uiteindelijk een sneller werkend programma op.



Schematische weergave van vertalen met de interpreter (boven) en compiler (onder).

Met behulp van deze basiskennis van de twee methodes moet u een eerste stap kunnen zetten in het kiezen van een passende programmeertaal voor uw plannen:

- Is het belangrijk dat het uiteindelijke programma razendsnel is? Ga dan voor een taal die werkt met een compiler, zoals C of C++.
- Is de snelheid van het uiteindelijk uit te voeren programma voor u niet zo belangrijk en heeft u meer baat bij een snelle vertaling (zodat u sneller kunt testen en aanpassen of snel van concept naar een uitvoerbaar programma kunt geraken)? Kies dan voor een taal die werkt met een interpreter, zoals JavaScript, Python en Ruby.

Voorbeeld

We gaan eens kijken hoe het er in de praktijk aan toe gaat. We beginnen met een code in Python, die met een **interpreter** wordt vertaald:

```
def print_10():
    for i in range(10):
        print(i + 1)

print_10()
```

Dit programma hebben we opgeslagen onder de naam 'test_Python.py'. De volgende afbeelding toont hoe we de computer vervolgens moeten vragen om deze opdracht uit te voeren:

```
python test_Python.py
```

Deze 'vraag' bestaat uit twee delen:

Het aanroepen van de Python-interpreter	python
De naam van het bestand	test_Python.py

Deze opdracht kan de computer alleen uitvoeren als Python is geïnstalleerd op deze computer. De opdracht wordt uitgevoerd zodra u na het typen van deze regel op Enter drukt. De code wordt dan in zijn geheel geïnterpreteerd en uitgevoerd. De computer zal de getallen 1 tot en met 10 onder elkaar op het scherm te zetten:

```
python test_Python.py  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

Dat ziet er tamelijk simpel uit, maar voor het uitvoeren van deze opdracht was het installeren van Python een vereiste. Dat is op zich niet heel moeilijk, maar het omslachtige van deze manier van werken is dat u ook van anderen (met wie u deze code deelt) verlangt dat zij Python op hun computer hebben staan.

Een ander nadeel van deze manier van werken is dat deze code gemakkelijk na te maken is door iemand met wie de code wordt gedeeld. Er staat immers stap voor stap (in het Engels) beschreven wat de computer moet doen.

Iets anders gaat het als u werkt met een **compiler**. We gaan dit keer uit van een code in C, die hetzelfde resultaat zal hebben als de eerdere code in Python:

```
#include <stdio.h>  
  
int main() {  
    int i;  
  
    for (i = 1; i < 11; ++i)  
    {  
        printf("%d \n", i);  
    }  
    return 0;  
}
```

Dit programma hebben we opgeslagen onder de naam 'c_test.c'.

We kunnen dit programma niet direct uitvoeren. We moeten het eerst compileren, waarbij een nieuw bestand wordt gemaakt (de *executable*). Dat nieuwe bestand kan vervolgens uitgevoerd worden:

```
gcc c_test.c -o test.exe
```

```
test  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

Het nieuwe bestand (dat we de naam 'test.exe' gaven) maakten we aan met behulp van het commando `gcc c_test.c -o test.exe`.

Daarna typten we `test` en drukten we op Enter, waarna het programma werd uitgevoerd. (Het typen van `test` is voldoende, omdat het systeem geen extra programma nodig heeft om dit .exe-bestand uit te voeren.)

Dat het programma niet direct wordt uitgevoerd, is niet het enige verschil ten opzichte van de voorgaande methode. Bij deze methode kunt u namelijk volstaan met het delen van het bestand 'test.exe', iemand anders hoeft geen software te installeren om het te laten werken. Wij moesten dus wel het programma GCC installeren, maar degene met wie u de code deelt, hoeft dat niet.



Ga naar Plaza en bekijk een video over deze uitleg

2.4 Evolutie van programmeertalen

U zag in de vorige les al een mooi overzicht van programmeertalen door de jaren heen. Deze beschrijving was niet volledig. Dat deden we bewust, want dat zou onnodig veel ruimte beslaan in deze cursus.

Wat in de vorige les niet aan bod kwam, maar wel nog interessant is, is het kijken naar gemeenschappelijke kenmerken. Welke patronen kunnen we ontdekken als we kijken naar de evolutie van programmeertalen?

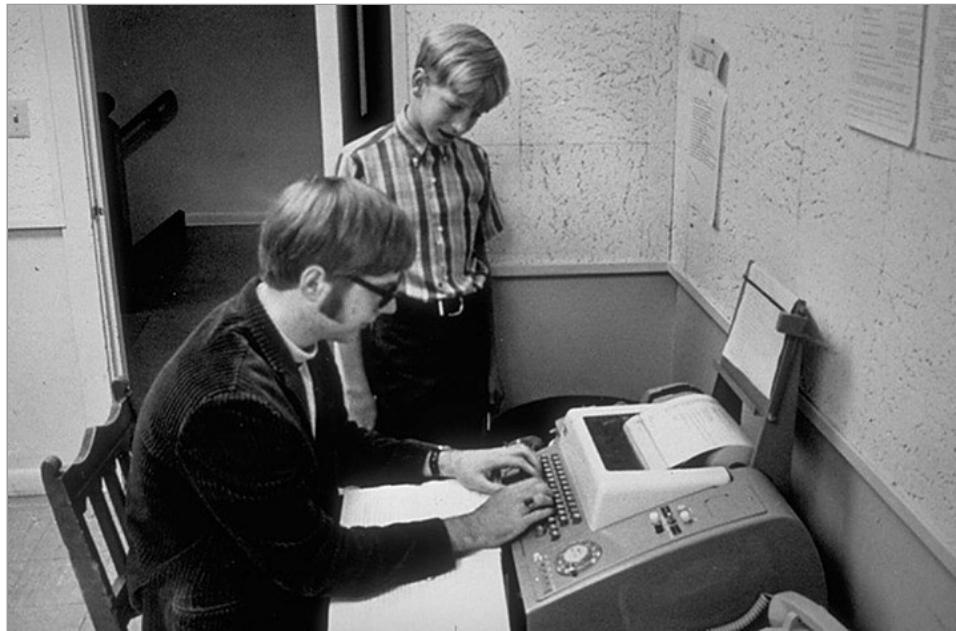
2.4.1 Patroon 1: hoog of laag

Met dit eerste patroon maakte u al kennis in de vorige les. We gaven toen aan dat we, vanaf het moment dat *Assembly Language* werd ontwikkeld, computertalen konden onderverdelen in 'lagere' en 'hogere' talen. Ter herhaling:

- Met een **hogere taal** bedoelen we een taal die dichter bij de menselijke taal staat, waardoor deze gemakkelijk is voor de menselijke gebruiker. Het nadeel is wel dat programma's die zijn geschreven in zo'n taal, iets trager zijn, omdat ze eerst (met een of meerdere tussenstappen) vertaald moeten worden naar de nullen en enen die voor de computer te begrijpen zijn.
- Een **lagere taal** staat juist 'dichter bij de computer'. Deze taal lijkt dus minder op de Engelse mensentaal en meer op iets wiskundigs en voor een leek onbegrijpelijs. Het vergt dus meer kennis en moeite voor de programmeur om de taal te beheersen en te gebruiken. Het grote voordeel is dat er minder vertaalslagen nodig zijn om het programma uiteindelijk door de computer te laten uitvoeren. Het programma is daardoor sneller uit te voeren. Games worden bijvoorbeeld vaak in een lagere programmeertaal geprogrammeerd.

De eerste computertalen stonden dichter bij de computer, het waren dus 'lagere' talen. Iemand die met deze talen werkte, was Bill Gates. In de dagen dat hij zich de kunst van het programmeren eigen maakte, programmeerde hij en zijn maat Paul Allen door ponskaarten in te voeren in een computer genaamd CDC 6400. Deze computer had nog geen beeldscherm. Ook het gebruik van toetsenborden kwam pas later op gang. In een interview vertelde Bill Gates over programmeren in die tijd (vrij vertaald):

Computers in die tijd hadden geen beeldschermen. We maakten de computerversie van het spel 'boter, kaas en eieren', waarbij we op een toetsenbord de zetten typten en moesten wachten tot de printer het resultaat had uitgeprint.



Bill Gates aan het werk in zijn jongere jaren.

Net als auto's zijn ook programmeertalen in de loop der tijd steeds gebruiksvriendelijker gemaakt. Daardoor konden niet alleen hooggeleerde wetenschappers en wiskundigen gebruikmaken van de computer, maar ook anderen.

Dat is echter niet de enige reden waarom 'hogere' programmeertalen zijn ontwikkeld. Door de komst van hogere talen kunnen we sneller van een concept naar een werkend computerprogramma gaan (zeker als u een scripttaal gebruikt, waarover later meer).



Het is niet zo dat door het ontstaan van 'hogere' programmeertalen de 'lagere' programmeertalen overboord gegooied zijn. Voor toepassingen zoals spellen is het zaak dat ze razendsnel uitgevoerd kunnen worden. Een programma dat gemaakt is in een 'lagere' programmeertaal, is vrijwel altijd sneller in de uitvoering dan een programma dat is gemaakt in een 'hogere' programmeertaal!

2.4.2 Patroon 2: lineair of object-georiënteerd

We kunnen van een taal niet alleen aangeven of deze hoog of laag is, maar ook of deze lineair of object-georiënteerd is. Wat het verschil is, maken we in eerste instantie duidelijk aan de hand van een voorbeeld:

Stelt u zich voor dat u een spel moet maken, een computergame. Er is één raket die schiet op meerdere tegenstanders – laten we zeggen dat het er zeven zijn. We zouden dan elke seconde kunnen controleren of tegenstander 1 wordt geraakt door onze kogel, of tegenstander 2 wordt geraakt door onze kogel, enzovoorts (= lineaire werkwijze).



Het is echter handiger om een soort sjabloon van onze tegenstander te maken. Die zijn immers allemaal hetzelfde: ze hebben dezelfde eigenschappen en gedragingen. We programmeren het sjabloon dusdanig dat elke kopie van dit sjabloon uit zichzelf elke seconde checkt of die geraakt wordt door een kogel. Vervolgens kunnen we ‘kopietjes’ van dit sjabloon (oftewel **objecten**) maken, zoveel als het aantal tegenstanders wil.

Simpel gezegd lopen lineaire programma's stap voor stap een aantal programmaregels met gegevens door. Bij object-georiënteerde programma's zijn de volgorde en gegevens georganiseerd rondom objecten.

Deze manier van werken, die ook wel *Object Oriented Programming* (OOP) wordt genoemd, kan veel werk schelen en uw code veel overzichtelijker en beter leesbaar maken. Ook zijn veranderingen veel gemakkelijker door te voeren.

2.4.3 Patroon 3: programmeertaal of scripttaal

Eerder gaven we al aan dat een computertaal via een *interpreter* of *compiler* vertaald kan worden naar nullen en enen. Deze vertaalmiddelen vormen de basis voor het derde patroon:

- Computertalen waarbij een *compiler* wordt gebruikt, worden over het algemeen **programmeertalen** genoemd.
- Computertalen waarbij een *interpreter* wordt gebruikt, worden over het algemeen **scripttalen** genoemd.

De eerste computertalen werkten met een *compiler*, dat was nu eenmaal de techniek van die tijd. De eerste *compiler* werd in 1952 ontwikkeld door Grace Hopper, voor de taal A-0.

Vanaf 1958, toen Steve Russell een interpreter ontwikkelde voor de taal LISP, werden de scripttalen ontwikkeld, vanwege de eerder genoemde voordelen: gemakkelijker voor niet-wetenschappers, snel resultaat en korte tijd tussen het coderen en testen.

In de twee bullets eerder in deze paragraaf gebruikten we twee keer de woorden ‘over het algemeen’. De splitsing is op basis van vertaalmiddel klopt namelijk niet voor 100%: er zijn ook computertalen waarvoor zowel een *interpreter* als een *compiler* bestaan, zoals de computertaal LISP.

In theorie kan elke computertaal geïnterpreteerd of gecompileerd worden. Als een bepaalde computertaal in de praktijk meestal gecompileerd wordt, spreken we voor het gemak vaak van een programmeertaal.

2.4.4 Patroon 4: frameworks

Er kunnen een of meer frameworks worden gebaseerd op een bepaalde programmeertaal. U kunt de programmeertaal zien als een basisstructuur voor het framework, die de ontwikkeling van een programma aanzienlijk kan versnellen. Iemand die werkt met een framework, hoeft namelijk niet vanaf nul te beginnen. Ook bevat een framework oplossingen voor veelvoorkomende problemen.

Bijvoorbeeld:

- *Bootstrap* is een framework dat op CSS (en Javascript) is gebouwd.
- De frameworks *Vue.js* en *React.js* zijn gebouwd op Javascript.
- Het Microsoft-framework *.NET* is gebouwd op C++.

In een latere les gaan we dieper in op het onderwerp ‘frameworks’.

2.5 Talen en hun toepassingen

Bij het beschrijven van de geschiedenis van programmeren, in de vorige les, zijn er al verschillende namen van programmeertalen kort de revue gepasseerd. In deze paragraaf gaan we dieper in op de programmeertalen die tegenwoordig veel gebruikt worden. U zult zien dat ze allemaal zo hun eigen toepassingen hebben.

2.5.1 HTML

Hoog of laag?	Hoog
Lineair of object-georiënteerd?	n.v.t.
Programmeer- of scripttaal?	n.v.t. (markuptaal)
Framework(s) op basis van HTML	n.v.t.

Om op het internet informatie te bekijken, gebruikt u een internetbrowser, ook wel 'webbrowser' of kortweg 'browser' genoemd. Door in de adresregel van het programma een webadres in te typen, roept u de startpagina van de website op. De webpagina is geschreven in een speciale opmaaktaal: **HTML**, wat staat voor *HyperText Markup Language*. De browser kan deze codering vertalen naar een grafische weergave op uw beeldscherm.

HTML ondersteunt **hypertekst**. Dat wil zeggen dat het mogelijk is om documenten en bestanden te verbinden door volgbare verwijzingen, die ook wel bekend zijn onder de naam **hyperlinks**.

Daarnaast is HTML een opmaaktaal. Door bepaalde codes kunt u angeven waar u bijvoorbeeld koppen, tabellen en alinea's wilt plaatsen. Ook afbeeldingen (of andere vormen van multimedia) kunt u dankzij HTML opnemen in uw webpagina.

```
1 <html>
2
3   <head>
4     <title>DevOps Demo 01</title>
5     <link rel="stylesheet" href="style.css">
6   </head>
7   <body>
8     <h1>DevOps Demo 01</h1>
9     <p>Made by [name]</p>
10    </body>
11
12 </html>
```

Voorbeeld van HTML.

2.5.2 CSS

Hoog of laag?	Hoog
Lineair of object-georiënteerd?	n.v.t.
Programmeer- of scripttaal?	n.v.t.
Framework(s) op basis van CSS	<i>Bootstrap, Foundation, Tailwind, ...</i>

Een pagina die alleen in HTML-taal is geschreven, ziet er erg simpel en eenvoudig uit. Om een aantrekkelijke pagina te creëren, wordt daarom ook gebruikgemaakt van **CSS**.

CSS staat voor *Cascading Style Sheets*. Terwijl u HTML gebruikt om een webdocument te structureren, helpt CSS bij het specificeren van de stijl van uw document. Met CSS kunt u de vormgeving van elk element in een webpagina bepalen (zoals lay-out, kleur, uitlijning, vorm, lettertype of animatie).

```
1 body{
2   text-align: center;
3   font-family:'Trebuchet MS', sans-serif;
4 }
5
6 h1{
7   font-size: 200%;
8   margin-top: 20px;
9 }
```

Voorbeeld van CSS.

2.5.3 JavaScript

Hoog of laag?	Hoog
Lineair of object-georiënteerd?	Object-georiënteerd mogelijk
Programmeer- of scripttaal?	Scripttaal
Framework(s) op basis van JavaScript	Vue.js, React.js, jQuery

Het gebruik van HTML en CSS is nog maar het begin. Wie een website ‘slim’ wil maken en als *content management-systeem* wil laten functioneren, moet andere programmeertalen toevoegen die deze acties vergemakkelijken. De meest voorkomende oplossingen voor deze taken zijn te vinden in programmeertalen als JavaScript, PHP, C#. In deze subparagraaf lichten we de taal JavaScript toe. Informatie over PHP en C# volgt verderop.

Terwijl HTML vooral de structuur van een pagina beschrijft, is JavaScript bedoeld om een pagina interactiever te maken. Enkele populaire toepassingen zijn bijvoorbeeld animaties, formuliervalidatie, uitrolmenu’s en fotocarrousels. Veel bekende websites maken gebruik van JavaScript, zoals *Office 365*, *Facebook*, *Instagram* en *Gmail*. JavaScript is zo populair, dat het door alle browsers uitgevoerd kan worden.

JavaScript werd in eerste instantie geschreven om functionaliteiten toe te voegen aan websites, vandaar dat het vaak genoemd wordt in combinatie met HTML en CSS. De taal is echter zo populair geworden, dat er inmiddels ook manieren bedacht zijn om JavaScript te gebruiken voor andere toepassingen.

```
function getRecommendations(products, purchasedProducts, numRecommendations) {
    const weights = [];
    for (let i = 0; i < products.length; i++) {
        const product = products[i];
        const weight = getWeight(product);
        weights.push({id: product.id, weight});
    }
    weights.sort((recommendation1, recommendation2) =>
recommendation2.weight - recommendation1.weight);
    return weights.slice(0, Math.min(numRecommendations,
weights.length));
}
```

Voorbeeld van JavaScript.

2.5.4 PHP

Hoog of laag?	Hoog
Lineair of object-georiënteerd?	Object-georiënteerd mogelijk
Programmeer- of scripttaal?	Scripttaal
Framework(s) op basis van PHP	Laravel, Symfony, CakePHP

Zonder te diep in te gaan op de werking van het internet, moet u begrijpen dat er bij een internetverbinding dingen gebeuren aan de kant van de server (de server van *YouTube* bijvoorbeeld) en aan de kant van de *client* (uw computer of mobiele telefoon). PHP is gemaakt om programmatuur te schrijven voor de kant van de server. De taal genereert HTML-pagina’s en de toepassingen voor PHP zijn dus vooral te vinden op het gebied van webpagina’s.

PHP is een concurrent van Python. Beide talen worden gebruikt voor de achterkant van een website (*back-end*). Een grappig detail is dat Python tegenwoordig vooral in de VS erg populair is en in Nederland op veel plaatsen PHP wordt gebruikt, terwijl Python is uitgevonden door een Nederlander (Guido van Rossum).

```

1 <?php
2 // single line comment 1
3 # single line comment 2
4 /* multi line comment - ?> */
5 /** doc-style comment - ?> */
6 function printNumber()
7 {
8     $number = 1234;
9     print "The number is $number,\n<?xml
10    for ($i = 0; $i <= $number; $i++) {
11        $x++;
12        $x--;
13        $x += 1.0;
14        $z = 2.; // error
15    }
16    if (PRINT_SUMMARY_CONSTANT)
17        echo <<< custom_HEREDOC
18 <br />
19 Summary:<br />
20 The \$x variable has value: $x
21 <br />
22 custom_HEREDOC;
23 }
24 ?>

```

Voorbeeld van PHP.

2.5.5 C++

Hoog of laag?	Laag
Lineair of object-georiënteerd?	Object-georiënteerd mogelijk
Programmeer- of scripttaal?	Programmeertaal
Framework(s) op basis van C++	Qt, Boost, Visual C++

Deze taal is een afgeleide van de taal C. C++ kan nog altijd beschouwd worden als een 'lagere' taal en wordt gebruikt voor het maken van games, mobiele apps, computersoftware en webapplicaties.

```

1 #include<iostream>
2 using namespace std;
3
4 int main() {
5     cout<<"Enter The Size Of Array:   ";
6     int size;
7     cin>>size;
8
9
10    int array[size], key,i;
11
12    // Taking Input In Array
13    for(int j=0;j<size;j++) {
14        cout<<"Enter "<<j<<" Element: ";
15        cin>>array[j];
16    }

```

Voorbeeld van C++.

2.5.6 C#

Hoog of laag?	Hoog
Lineair of object-georiënteerd?	Object-georiënteerd
Programmeer- of scripttaal?	Programmeertaal
Framework(s) op basis van C#	.NET, Visual C#, WPF

Ook deze programmeertaal is een afgeleide van C, maar deze variant is ontwikkeld tot een 'hogere' taal. Voor Microsoft is het de belangrijkste taal om zijn applicaties in te programmeren. Ook als u games wilt maken in de bekende game-engine *Unity*, is het leren van C# aan te bevelen.

```
void MyFunction () {  
  
    myLight.enabled = !myLight.enabled;  
}  
  
int AddTwo (int var1, int var2) {  
    int returnValue = var1 + var2;  
    return returnValue;  
}
```

Voorbeeld van C#.

2.5.7 Java

Hoog of laag?	Hoog
Lineair of object-georiënteerd?	Object-georiënteerd mogelijk
Programmeer- of scripttaal?	Programmeertaal
Framework(s) op basis van Java	Spring, Apache Wicket, Grails

Het voordeel van Java is dat de applicaties die u met deze taal maakt, op verschillende soorten devices uitgevoerd kunnen worden (dus op mobiele telefoons, tablets en pc's). Veel apps op mobiele telefoons zijn geschreven in Java.

```
package rentalStore;  
import java.util.Enumeration;  
import java.util.Vector;  
  
class Customer {  
    private String _name;  
    private Vector<Rental> _rentals = new Vector<Rental>();  
  
    public Customer(String name) {  
        _name = name;  
    }  
    public String getMovie(Movie movie) {  
        Rental rental = new Rental(new Movie("", Movie.NEW_RELEASE), 10);  
        Movie m = rental._movie;  
        return movie.getTitle();  
    }  
    public void addRental(Rental arg) {  
        _rentals.addElement(arg);  
    }  
    public String getName() {  
        return _name;  
    }  
}
```

Voorbeeld van Java.

2.5.8 SQL

Hoog of laag?	n.v.t.
Lineair of object-georiënteerd?	n.v.t.
Programmeer- of scripttaal?	n.v.t
Framework(s) op basis van SQL	-

Deze taal is een beetje 'vreemde eend in de bijt': hij wordt gebruikt om bepaalde typen databases aan te maken en te bewerken. (Een database is een gestructureerde verzameling data.)

```
SELECT TOP (1000) [id]
    ,[code]
    ,[firstName]
    ,[lastName]
    ,[locationID]
FROM [SampleDB].[dbo].[employees]
```

Voorbeeld van SQL.

2.5.9 Ruby

Hoog of laag?	Hoog
Lineair of object-georiënteerd?	Object-georiënteerd mogelijk
Programmeer- of scripttaal?	Scripttaal
Framework(s) op basis van Ruby	(Ruby on) Rails, Scorched, Cuba

Ook deze taal wordt toegepast bij het maken van websites/webapplicaties. Ruby kenmerkt zich door het feit dat een programmeur snel tot resultaat kan komen. De taal wordt veel gebruikt met het framework *Rails*.

Ruby is een **opensourcetaal**. Dat wil zeggen dat deze niet in beheer is van een bepaald bedrijf, maar in principe door iedereen kan worden uitgebreid en mede ontwikkeld.

```
1      invoke  active_record
2      create   db/migrate/20151212160304_create_books.rb
3      create   app/models/book.rb
4      invoke  resource_route
5          route   resources :books
6      invoke  scaffold_controller
7      create   app/controllers/books_controller.rb
8      invoke  erb
9      create   app/views/books
10     create   app/views/books/index.html.erb
11     ...
```

Voorbeeld van Ruby.

2.5.10 Swift

Hoog of laag?	Hoog
Lineair of object-georiënteerd?	Object-georiënteerd mogelijk
Programmeer- of scripttaal?	Programmeertaal
Framework(s) op basis van Swift	Quick, Vapor, SwiftMonkey

Dit is de taal waarmee de meeste applicaties voor *Apple-devices* worden geschreven (zoals de *iPhone* en *iMac*). Het is een relatief nieuwe taal met veel elementen uit Python en Ruby.

```

struct Player {
    var name: String
    var highScore: Int = 0
    var history: [Int] = []

    init(_ name: String) {
        self.name = name
    }
}

var player = Player("Tomas")

```

Voorbeeld van Swift.

2.5.11 Python

Hoog of laag?	Hoog
Lineair of object-georiënteerd?	Object-georiënteerd mogelijk
Programmeer- of scripttaal?	Scripttaal
Framework(s) op basis van Python	Django, Flask, Pyramid, Enthought Tool Suite

Python is een hogere programmeertaal die werd ontwikkeld door de Nederlander Guido van Rossum. Deze taal is zeer leesbaar en relatief eenvoudig te leren. Vandaar ook dat we in het vervolg van deze cursus met Python zullen werken.

Toepassingen van deze taal zijn *machine learning*, *datamining* en allerhande computerapplicaties. Ook gebruiken wetenschappers deze taal veel voor het maken van berekeningen.

Door gebruik te maken van frameworks als *Flask* en *Django* kan Python zelfs gebruikt worden als *back-end language* voor websites. Een bekend voorbeeld van een website die dat doet, is *YouTube*.

```

num = int(input("Enter the value of n: "))
hold = num
sum = 0

if num <= 0:
    print("Enter a whole positive number!")
else:
    while num > 0:
        sum = sum + num
        num = num - 1;
    # displaying output
    print("Sum of first", hold, "natural numbers is: ", sum)

```

Voorbeeld van Python.

2.6 Samenvatting

Programma's worden geschreven in programmeertalen. Dat is maar goed ook, want programmeren in binaire code is een tijdrovende en onoverzichtelijke klus (en voor beginners absoluut geen haalbare kaart). De programmeertaal is de tussenstap tussen mens en computer.

De computer zal de door u geschreven code uiteindelijk omzetten naar nullen en enen, zodat hij iets met deze code kan. Dat doet de computer met behulp van een *interpreter* of *compiler*. Bij de ene methode verloopt het vertalen sneller, bij de andere methode is het programma juist razendsnel.

In de loop der jaren zijn er allerlei programmeertalen ontwikkeld, ieder met zijn eigen kenmerken. In deze kenmerken zijn enkele patronen te ontdekken:

- Een taal is hoog of laag.
- Een taal is lineair of object-georiënteerd.
- Een taal is een programmeertaal of een scripttaal.
- Er kunnen frameworks gebaseerd zijn op een programmeertaal.

Aan de hand van deze patronen zijn er enkele programmeertalen die momenteel populair zijn, zoals HTML, CSS, JavaScript, PHP, C++, C#, Java, SQL, Ruby en Swift. We benoemden hun toepassingen en enkele specifieke kenmerken.

2.7 Vraagmoment

Heeft u vragen over deze les? Dan kunt u die stellen via een vraagmoment.

U herkent een vraagmoment aan het spreekballonnetje dat op Plaza in het overzicht "Lessen" bij een les staat. Door op deze spreekballoon te klikken, komt u op een nieuwe pagina, waar u uw vraag in het tekstvak kunt typen.



Het tekstvak mag maximaal vijfhonderd tekens bevatten. Zorg er dus voor, dat u uw vraag/vragen bondig formuleert.

Nadat u op "Verstuur vraag" heeft geklikt, wordt uw vraag automatisch naar uw docent verstuurd. Zodra uw docent de vraag heeft beantwoord, verschijnt er op uw dashboard bij de opleiding een spreekballoon.

U kunt per les één keer vragen stellen. Verzamel dus uw vragen over de les, voordat u deze instuurt. Als u gebruik heeft gemaakt van het vraagmoment, verdwijnt het spreekballonnetje bij de les.

2.8 Oefenopgaven

De volgende opgaven werkt u als oefenopgaven voor uzelf uit. De uitwerkingen van deze opgaven kunt u meteen zelf controleren.

Waarom hebben we programmeertalen nodig?

→ Bekijk antwoord

Wat is het verschil tussen een *compiler* en een *interpreter*?

→ Bekijk antwoord

Wat is het verschil tussen een hogere programmeertaal en een lagere programmeertaal?

→ Bekijk antwoord

Lees de volgende stelling:

Sinds we hogere programmeertalen kennen, werken we nauwelijks nog met lagere programmeertalen.

Is deze stelling juist of onjuist?

 [Bekijk antwoord](#)

Geef van de volgende talen aan of het een 'hogere' of een 'lagere' taal is:

- a. C
- b. C++
- c. C#
- d. Python

 [Bekijk antwoord](#)

Wat is het verschil tussen lineaire en object-georiënteerde programma's?

 [Bekijk antwoord](#)

Maakt een scripttaal over het algemeen gebruik van een *interpreter* of een *compiler*?

 [Bekijk antwoord](#)

Wat is een framework?

 [Bekijk antwoord](#)

Op welke taal is het framework .NET gebouwd?

 [Bekijk antwoord](#)

2.9 Huiswerkopgaven

Maak deze huiswerkopgaven en stuur ze via Plaza naar uw docent. Deze opgaven worden nagekeken en voorzien van een cijfer. Mocht u een onvoldoende cijfer behalen, dan krijgt u een herkansing.

1. Lees de volgende casussen. Geef per casus aan welke taal er het beste gebruikt voor kan worden. Licht uw keuze kort toe.

- a. *Jeremy krijgt een opdracht van NHA Opleidingen. Hij moet een database aanmaken waarin de huiswerkcijfers van cursisten kunnen worden bijgehouden, zodat de prestaties van cursisten beter gemonitord kunnen worden.*
- b. *Tessa krijgt een opdracht van de lokale sportschool. Die heeft al een reserveringsapp voor Android-telefoons, maar wil ook een applicatie ontwikkelen voor het maken van een applicatie voor iPhone-gebruikers.*

- c. Hans heeft een eigen bedrijf met een zelfgemaakte website. Die website kan wel een opfrisbeurt gebruiken, ze moet gestyled worden. Hans huurt hiervoor een webdesigner in.
2. Heeft u zelf toekomstige programmeerplannen? Omschrijf die plannen dan kort en benoem welke taal volgens u het beste bij deze plannen past. Licht uw keuze kort toe.

(Heeft u nog geen concrete programmeerplannen? Benoem dan waarom het volgen van deze cursus voor u van belang is.)

3. In deze les leerde u over frameworks.

- a. Zoek uit op welke taal het framework *Flask* is gebouwd.
- b. Zoek uit op welke taal het framework *Laravel* is gebouwd.
- c. Zoek een ander, nog niet genoemd voorbeeld van een framework dat is gebaseerd op een programmeertaal.

4. Bekijk het volgende deel van een code:

```
var getal1=5;  
var getal2=8;  
var som=getal1 + getal2;  
document.write(som);
```

- a. Zoek met behulp van het internet (of andere bronnen) uit in welke programmeertaal deze code is geschreven.
- b. Een voordeel van programmeertalen is dat ze dicht bij de Engelse taal liggen, waardoor u vaak aan de code kunt zien wat deze zal doen. Maak deze inschatting voor deze code. Wat zal er na uitvoering van deze code op het scherm te zien zijn?