Universiteit Leiden
The Netherlands

Opleiding Informatica

Prestatietest Ontwikkeling voor de Evaluatie van Python Vaardigheden van Middelbare Scholieren voor het Vak Nederlands

Melda Malkoc

Supervisors:
Dr.ir. F.F.J. Hermans & Dr. S.H. Yeni

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

28/01/2021

Samenvatting

Het doel van dit onderzoek is het ontwikkelen van een automatisch beoordeelde prestatie test voor Python, geïntegreerd in het vak Nederlands op middelbare scholen. De test is gebaseerd op een lessenreeks waarin programmeerkennis met het vak Nederlands is geïntegreerd. Hierbij leren middelbare scholieren de basis van het programmeren: commentaar schrijven, het gebruik van variabelen, lijsten, prints, gebruikersinput, if-else statements en het herkennen van foutmeldingen. Zo kunnen ze aan de hand van deze lessenreeks een verhaal leren programmeren in Python. Deze studie bevat een theoretisch kader en een praktisch actieonderzoek. Dankzij het literatuuronderzoek is er een eerste versie van de toets gecreëerd bestaand uit negenendertig vragen. Deze is vervolgens per item beoordeeld door zes deskundigen met de focus op de relevantie, formulering, bias, nut en het niveau. Middels de validatie, de Bloom Taxonomie en de leerdoelen is de test verkort. Het gehele onderzoek resulteerde in een test bestaand uit achttien vragen en een handleiding over de creatie van prestatietesten voor eventuele vervolgmodules van het programma.

Inhoudsopgave

1 Inleiding **1**

2 Theoretische kader **3**
2.1 Integratie van Informatica in schoolvakken . . . 3
2.1.1 Voordelen van integratie van CT in talen . . . 4
2.2 Automatische beoordelingssystemen gebruikt bij toetsingen in het onderwijs . . . 5
2.2.1 Voordelen van automatische beoordelingssystemen . . . 6
2.2.2 Nadelen van automatische beoordelingssystemen . . . 6
2.3 Prestatietest creatie voor educatieve doeleinden . . . 7
2.3.1 Soorten en karakteristieken van prestatietests . . . 8
2.3.2 Gevolgde stappen bij het samenstellen van een prestatietest . . . 9
2.4 Richtlijnen voor items schrijven en becijfering van de prestatietest . . . 10
2.4.1 Huidige richtlijnen voor het schrijven van items bij programmeervakken . . . 10
2.4.2 Gebruikte termen bij het opstellen van een toetsing . . . 11
2.4.3 Taxonomieën en richtlijnen voor de becijfering . . . 11

3 Methodologie **12**
3.1 Type onderzoek . . . 12
3.2 Module Nederlands . . . 13
3.3 Opstelling toetsing . . . 13

4 Resultaten **13**

5 Conclusie **24**

6 Tekortkomingen **25**

7 Aanbevelingen **25**

Referenties **26**

Appendices **29**

1 Inleiding

Programmeren komt steeds vaker aan bod in het onderwijs voor jongere groepen. Al op de basisschool wordt technologie geschoold door computervaardigheden en oplossingsgericht denken te onderwijzen. Het huidige onderwijssysteem over de hele wereld kampt met de integratie van digitale geletterdheid in zijn onderwijsprogramma. Een goede integratie van rekenkracht van informatica kan zorgen voor oplossingen in andere domeinen. Hiervoor zijn er verschillende initiatieven ondernomen door onder andere Stichting CodeUur, Google en VHTO (Landelijk expertisebureau meisjes/vrouwen en bèta/techniek).[14][35][48] Voor het meten van de prestatie in het onderwezen domein is een toetsing van belang, omdat het gaat om een groep die grotendeels voor het eerst met de abstracte concepten van programmeren te maken krijgt. Dit zorgt voor een overzicht van de prestatie van de beginners bij het moeilijkste deel van het programmeren. [26]

Dit onderzoek beoogt de creatie van een summatieve toetsing gebruikmakend van een geautomatiseerd beoordelingssysteem dat de kennis van Python, geïntegreerd in het vak Nederlands, kan beoordelen en voorzien van een eindcijfer. De toetsing wordt gecreëerd voor de Module Nederlands, de eerste module onder het programma Computer Science Certificate van de VHTO, een stichting die zich inzet voor de vergroting van participatie van meisjes en vrouwen in bèta vakken, techniek en ICT. [48] Het doel van het programma is het enthousiasmeren van studenten om in de bovenbouw informatica te kiezen als keuzevak.[49] Het programma Computer Science Certificate bestaat momenteel uit drie lessenreeksen om de basis van Python te onderwijzen. De Module Nederlands leert jongeren een verhaal te coderen in Python voor het vak Nederlands. Ze leren hiervoor verschillende concepten die essentieel zijn voor het schrijven van een verhaal. Deze concepten zijn datatype strings, commentaren, variabelen, lijsten, gebruikersinput, if-else statements en foutmeldingen. Doordat programmeren wordt gekoppeld met de inhoud van het vak Nederlands, wordt het voor leerlingen makkelijker hun vaardigheden in verschillende contexten te plaatsen.[49]

Voor het testen van de verkregen vaardigheden na de module is er een grondig onderzoek naar het creëren van een prestatietest gericht op de overeenkomstige inhoud nodig. Er zijn onderzoeken naar de voordelen van geautomatiseerde beoordelingssystemen [29][26][21][1]; over de analyse van dergelijke systemen [11][1]; over richtlijnen voor de creatie van vragen in vakgebieden buiten de informatica.[23][32][39] Daarentegen zijn onderzoeken over het maken van een goede programmeertoets met fundamenten uit pedagogische richtlijnen, benodigde taxonomieën en gebruikmakend van de voordelen van automatische beoordelingssystemen schaars.[42] Dit terwijl deze aspecten benodigd zijn voor een toetsing van hoge kwaliteit, alsmede kennis van vakgebieden, zoals wiskunde, taalkunsten of wetenschap.[15] De onderzoeken die een gedeeltelijke raakvlak hebben zijn meestal regionaal, met ongelijksoortige aspecten zoals programmeerniveau, aangeleerde vaardigheden, leeftijden, vraagstellingen of programmeertalen.

Dit onderzoek speelt een centrale rol in het vervullen van de genoemde gebreken in de wetenschap en zal hulp bieden bij toekomstige initiatieven in programmeren waar toetsing van belang is. Ook wordt er in het onderzoek van Obidiegwu (2005) gesuggereerd docenten door middel van bijscholing, seminars en workshops te onderwijzen over de testconstructie en over de regels voor het schrijven van items, zodat ze uitgebreide, systematische en objectieve instrumenten kunnen ontwikkelen waarmee echte schattingen van de prestaties kunnen worden verkregen.[36] Dit onderzoek zal daar ook een inhoudelijke bijdrage bij kunnen leveren.

De huidige eindopdracht voor het behalen van het certificaat bestaat uit het schrijven van een stukje code waarin een aantal van de geleerde onderwerpen moet worden verwerkt. Uiteindelijk

wordt dit nagekeken door de eigen docent. Deze eindopdracht dekt niet de gehele lesstof. Voor de eerste module is een toetsing nodig die gegrond is op pedagogische fundamenten met richtlijnen in het vakgebied informatica. Deze toetsing moet in een online omgeving plaatsvinden, zodat automatische terugkoppeling mogelijk is, de docenten niet extra worden belast en er geen sprake is van subjectiviteit.

Het doel van het onderzoek is om inzicht te krijgen in hoe een toets wordt opgesteld voor informatica onderwerpen die zijn geïntegreerd in het vak Nederlands, zodat een eindtoets kan worden gemaakt. Met behulp van de bevindingen in dit onderzoek kan de VHTO toetsingen opstellen voor vervolgmodules.

De hoofdvraag luidt als volgt: **Hoe kan een prestatietest voor de evaluatie van Python vaardigheden voor de Module Nederlands worden ontwikkeld?**
Om deze hoofdvraag te beantwoorden is er gebruik gemaakt van de beschikbare literatuur. Tijdens het schrijven van deze paper zijn de volgende vragen in acht genomen:

- Wat is de huidige toestand van integratie van informatica onderwerpen in schoolvakken op het basis- en voortgezet onderwijs?
- Hoe wordt deze integratie toegepast en wat zijn de gevolgen daarvan?
- Wat houdt een automatisch beoordelingssysteem gebruikt bij toetsingen in en wat zijn de voor- en nadelen ervan?
- Wat is een prestatietest en wat wordt er mee gemeten?
- Tot welke soort hoort de ontwikkelde test voor dit onderzoek?
- Wat zijn de gevolgde stappen bij het ontwikkelen van een prestatietest?
- Wat zijn de richtlijnen voor het ontwikkelen van programmeertoetsen?
- Wat zijn de methodes die worden gebruikt door programmeerdocenten om toetsingen te becijferen?

In Hoofdstuk 2 worden deze vragen beantwoord aan de hand van de beschikbare literatuur. Dit hoofdstuk is verdeeld in vier delen, waarvan de eerste deel de vragen over integratie van informatica in schoolvakken beantwoord. In de tweede deel wordt de geschiktheid van automatische beoordelingssystemen behandeld aan de hand van zijn voor- en nadelen.

In de derde deel worden prestatietests onder de loep genomen. Allereerst zullen de soorten prestatietest met hun karakter beschreven worden, waarna er ook aandacht besteedt zal worden aan hoe prestatietest worden ontwikkeld.

Vervolgens wordt de literatuur review afgesloten door in te gaan op de inhoud van een toetsing, de delen waaruit het bestaat en de bijbehorende richtlijnen. Aan het einde van het laatste deel wordt er ook ingegaan op de becijfering van prestatietests.

In de methode, Hoofdstuk 3, wordt er ingegaan op het praktische actieonderzoek, waarbij er een oplossing wordt gegeven voor het probleem ervaren bij de VHTO. Het theoretische kader is een literatuuronderzoek die een basis vormt voor hoofdstuk drie waarin een veldonderzoek

wordt uitgevoerd. In de resultaten, Hoofdstuk 4, wordt beschreven hoe de toetsing is opgesteld gebruikmakend van de methode. Deze prestatietest wordt vervolgens bekeken door zes experts en beoordeeld op het niveau, de aanwezigheid van bias, het nut, relevantie en de formulering in een online omgeving. Hiermee wordt de uiteindelijke toetsing ontwikkeld.

2 Theoretische kader

In het theoretische kader is er een studie gedaan naar de beschikbare literatuur om uiteindelijk de hoofdvraag te kunnen beantwoorden. Alle vier de delen waaruit dit hoofdstuk bestaat, geven een andere invalshoek op het gebied van toetsing bij informatica en programmeren in het huidige educatiesysteem. In het eerste gedeelte wordt er ingegaan op de huidige situatie van integratie van informatica in schoolvakken. Er zal besproken worden hoe dit wordt gedaan en wat de gevolgen hiervan zijn. Hierop volgend worden in de tweede deel de werking en de voor- en nadelen van automatische beoordelingssystemen onder de loep genomen. In de derde deel van dit hoofdstuk wordt er gekeken naar de toetsingsvorm van de prestatietest die geïntegreerd zal worden in het automatische beoordelingssysteem. Onderwerpen die hierbij aan bod zullen komen zijn: wat de vorm inhoudt, wat de soorten zijn en hoe het wordt ontwikkeld. In de laatste deel worden de richtlijnen die gehanteerd zijn bij de ontwikkeling van de toetsing besproken.

2.1 Integratie van Informatica in schoolvakken

Hoewel het gebruik van computers al op jongere leeftijd begint, weten jongeren niet hoe ze er optimaal gebruik van kunnen maken, zowel op creatief als op kritisch vlak. Het laatste decennium richt het onderwijs zich steeds meer op het stimuleren van het ontstaan van actieve ontwikkelaars in plaats van passieve gebruikers van computers om digitale geletterdheid te vermijden.[7] Hierbij geven ze leerlingen de mogelijkheid om toegang te krijgen tot, het analyseren van en zich bezig te houden met dergelijke technologie. Dit gebeurt zowel op kritisch als creatief vlak.[7] Zo kunnen leerlingen optimaal gebruik maken van de technologie en actief meewerken in de ontwikkeling ervan.

Computational thinking (CT) is een methode waarbij gebruik wordt gemaakt van informatica om problemen op te lossen, systemen te ontwerpen en menselijk gedrag te begrijpen. Dit wordt gedaan door abstracte redenatie en algoritmisch denken.[51] Door middel van CT is het mogelijk studenten te verplaatsen naar de producerende zijde.

Onderzoeken tonen aan dat CT integreren in een al bestaand vak zowel de vaardigheden voor het desbetreffende vak, als de informatica vaardigheden verbeterd.[3] Dit bespaard ook tijd, omdat er geen extra vak door de jongeren wordt gevolgd naast het verplichte curriculum. Bij de integratie in schoolvakken is er momenteel geen algemeen geaccepteerde methode, verschillende onderzoeken gebruiken verschillende methodes. Ook zijn er nog vele uitdagingen op pedagogische en psychologische vlakken.[30]

Hsu (2018) heeft een onderzoek gedaan naar 46 onderzoeken over CT implementatie gepubliceerd tussen 2006 en 2017. Het onderzoek rapporteert dat de meeste integraties gebeuren met behulp van probleem-, project- en op groepsverband gebaseerde leerstrategieën. Voor middelbare scholieren, die de doelgroep van deze thesis zijn, is spel-gebaseerd-leren ook populair. Deze activiteiten zijn geïntegreerd in verschillende vakken en dan waarvan met name in de vakken programmeren,

informatica en wiskunde. Ook zijn er geringe aantal integraties in andere vakgebieden zoals taal, natuurkunde en journalistiek.[20]

Hoewel CT ook zonder programmeren geleerd kan worden is het wel de meest populaire methode, vervolgens experimenteren en computer spelletjes. Net als de manier van integratie in vakken is er ook geen consensus over het gebruik van een programmeertaal. Zo worden 20 verschillende talen gebruikt in 46 verschillende onderzoeken. Wel is meer dan de helft ervan block-based programming waaronder bijvoorbeeld Scratch en ALICE.[20] Dit komt doordat ze gebruiksvriendelijk zijn voor beginners.[22]

Volgens CSEDU (2019) is er geen consensus in de beoordeling van CT verkregen uit de 46 onderzoeken. De beoordeling wordt gedaan in kleine groepen met vooral summatieve toetsingen, formatieve toetsingen, vragenlijsten en enquêtes. Zowel duidelijke doelen voor de programma's als voor de toetsingen en pedagogische fundamenten in de ontwikkelingsstadia waren schaars.[30]

Hsu geeft aan dat de onderzoeken naar integratie in taalpakketten zijn gebaseerd op project gebaseerd leren en systematische berekeningsstrategieën.[20] We kunnen bij taal denken aan verschillende toepassingen, zoals: spellingscontrole passend bij het concept algoritme en procedure van CT; hoofdlijnen schrijven bij het concept probleem ontleding; patronen herkennen voor verschillende zinnen bij het concept data analyse.[4]

DISSECT heeft een aantal voorbeelden van integratie waar Lord of the Flies één van is. Met behulp van het boek met dezelfde naam vinden leerlingen alle vormen van de acht symbolen. Deze worden in een gedeelde Google Document gezet en per symbool wordt de ontwikkeling ervan in het verhaal beschreven. Hierbij gebruiken ze het concept abstractie.[34]

Een andere voorbeeld van integratie is bij levenswetenschappen studenten waarbij verschillende domeinen worden gerelateerd aan CT concepten. Zoals de koppeling van biologische netwerken met de Dijkstra algoritme. Zo kunnen studenten problemen waar ze binnen hun studie tegen aanlopen, oplossen door te programmeren. Hierbij is de focus niet op de syntax van de programmeertaal, maar op de CT concept. Uit de resultaten van zulke integraties blijkt dat het perspectief tegenover informatica verandert, het wordt niet meer gezien als een machine maar als een probleemoplosser.[28]

2.1.1 Voordelen van integratie van CT in talen

In het DISSECT onderzoek hebben kinderen een kritische en vergelijkende analyse leren schrijven van geselecteerde literatuur door de integratie van CT in compositie en literatuur. Deze studenten presteerden 15% beter in het schrijfgedeelte dan de controle groep.[34] Bij IProg, waar het koppelen van theorie en praktijk door middel van CT werd aangeleerd, gaf een soortgelijk resultaat. De studenten hadden dankzij de CT implementatie een motiverende omgeving waarin studenten de producenten waren van het materiaal. Hierdoor ontstond er een natuurlijk en objectief leerproces.[2] Een ander programma leidde tot een betere ontwikkeling van CT, poëtisch denken en een volwassener schrijfstijl onder studenten in vergelijking met diegene die niet meededen.[22]

Een onderzoek, dat soortgelijke ontwikkelingen heeft gemeten, meldt dat studenten het gevoel hadden van zelfontwikkeling, zowel in het leren van Engels als het belang van de taal zien, werken met een groep en het verkrijgen van een juiste leerstijl. Deze studenten gaven ook aan dat ze het werken met Scratch leuk vonden.[31] Omdat lezen en schrijven belangrijke onderdelen zijn van leren, begrijpen en communiceren in alle disciplines, zullen zulke cursussen effect hebben op het succes in alle vakgebieden.[34]

Het is belangrijk om te vermelden dat de docenten een grote rol spelen in het succes van

integratie. Hun programmeervaardigheden [31]; bewustheid van het belang van CT [24]; kennis over het verband van CT en de schoolvakken [3]; manier van lesgeven en de inhoud van de lessen [24] hebben effect op de studenten.

Het integreren van CT in vakken heeft gezorgd voor verbetering in het vak zelf; computational thinking; motivatie voor het leren voor het vak; het belang van het vak in te zien en vertrouwen in de informatica vaardigheden.[52][31][34]

Ondanks deze veelbelovende resultaten is er tot op heden geen consensus over de manier van meten van de opgedane kennis.[30] Voor het meten en verbeteren van deze integratie in het onderwijs is daarentegen een toetsing van belang. De ontwikkeling van een toetsing die hiervoor kan worden gebruikt, wordt in de volgende hoofdstukken dieper uitgewerkt.

2.2 Automatische beoordelingssystemen gebruikt bij toetsingen in het onderwijs

Dankzij de ontwikkelende technologie worden steeds meer handelingen geautomatiseerd voor de vergemakkelijking van taken. Een automatisch beoordelingsprogramma is een voorbeeld hiervan, vaak gebruikt voor educatieve doeleinden. Een automatisch beoordelingsprogramma wordt niet alleen toegepast bij eindtoetsen, maar ook bij opdrachten en proeftoetsen. Het zorgt voor de automatische controle van gegeven antwoorden, becijfering en vereenvoudiging van het feedback proces met behulp van software.[29] In dit onderzoek wordt dit middel gebruikt bij de creatie en beoordeling van een eindtoets. Hierbij wordt de kennis in programmeren in Python geïntegreerd in het vak Nederlands getest met behulp van verschillende soorten vragen die automatisch gecontroleerd kunnen worden. Deze beoordelingsvorm brengt daarbij voordelen en nadelen met zich mee, waar in dit hoofdstuk verder op ingegaan wordt.

Automatische beoordelingsprogramma's zijn meestal niet gevoelig voor de inhoud van de opdracht, maar worden vaak gebruikt als informatieverstrekker, beheerder of een algemeen beoordelingsinstrument. [25] De meest simpele manier is het automatisch controleren van meerkeuzevragen, maar er bestaan ook ingewikkeldere vormen die bijvoorbeeld algoritmes kunnen uitvoeren en vergelijken met de kloppende invoer van de docent.[29] Over de laatste 15 jaar zijn er een aanzienlijke aantal programma's gemaakt om bijvoorbeeld de performance of competentie van de student te beoordelen [26], zoals: Smart Lab, CodeInsights, CloudCoder.

Als het echt gaat om een code schrijven bij programmeervakken dan is een systeem die meerkeuze en daarop lijkende vragen beoordeelt niet voldoende. Zulke programma's kunnen meestal alleen met string matching werken. Dat houdt in dat het ingevoerde antwoord letterlijk hetzelfde moet zijn als wat de docent heeft ingevoerd. Echter, code schrijven werkt eenmaal niet zo, want er zijn verschillende methodes om tot dezelfde oplossing te komen.[26] Een voorbeeld daarvoor is een while en for loop. Deze kunnen precies dezelfde oplossing verkrijgen met een andere syntax.

De beoordeling van codes is onder te verdelen in twee algemene vormen; *statisch* en *dynamisch*. De eerste controleert de code zelf zonder het uit te voeren, terwijl de tweede kijkt naar de uitvoer van het programma.[26] Ook is er een speciale vorm van automatisering dat gebruikt wordt voor sommige online beoordelingsprogramma's. Dit heet *semi-automatisering*, wat in het kort betekent dat de onderwijzer ook betrokken moet zijn tijdens de beoordeling.[29]

2.2.1 Voordelen van automatische beoordelingssystemen

Een automatisch beoordelingssysteem helpt zowel onderwijzers als studenten. De beoordeling van een opdracht of toets gaat sneller; het is altijd bereikbaar; het blijft consistent; het is goedkoper en het is objectief. Hiernaast maken onderwijzers tijdwinst, ze hoeven niet iedere leerling apart te beoordelen, wat weer op zijn beurt zorgt voor de mogelijkheid meer opdrachten te geven.[29][1] De kostbare tijd kan worden besteed aan zaken die niet kunnen worden geautomatiseerd. Ook kunnen er informatie en statistieken worden verkregen over de toetsing en houding van de leerlingen. Dit zal interessanter zijn dan alleen een cijfer analyseren.[29]

De feedback die wordt verkregen kan een overzicht geven over het vak en elke leerling apart. Op deze manier kunnen docenten, leerlingen, maar ook buitenstaanders, zien of de leerdoelen worden bereikt.[21] In dit geval kunnen we zeggen dat er geen verschil is tussen automatische en handmatige beoordeling; er kan bij beiden een individuele feedback zijn waardoor er een overzicht is per leerling of een algemene beoordeling van de veel gemaakte fouten van de groep. Echter wordt handmatige beoordeling lastiger naarmate de te beoordelen groep groter wordt. In zulke gevallen moet de hoeveelheid beoordeeld werk worden verminderd met bijvoorbeeld minder opdrachten of efficiëntie op een andere manier. Automatische feedback wordt daarentegen niet beïnvloed door een groeiende groep.[26][21]

Een laatste voordeel dat een automatisch systeem met zich meebrengt is de mogelijkheid tot personalisatie van de opdrachten. Hiermee kunnen studenten verschillende opdrachten ontvangen waardoor plagiaat wordt bestreden en samenwerking bij probleem oplossing wordt aangemoedigd.[29] Ook kan er ingespeeld worden op het niveau van de studenten, zo kunnen sommigen worden uitgedaagd en anderen geleidelijk worden aangestuurd.

De automatisering van de toetsing en beoordeling in dit onderzoek demonstreert een groot deel van de genoemde voordelen. De docenten hebben tijdwinst omdat ze de toets niet beoordelen, hierdoor kunnen ze hun aandacht ergens anders op richten. Dit kan de modulen aantrekkelijk maken voor nieuwe klassen waarvan de grootte ook niet uitmaakt omdat de beoordeling van een kleine groep hetzelfde is als die van een grotere groep. Ook mogen de docenten zelf bepalen wanneer de toetsing zal plaatsvinden omdat het altijd beschikbaar is. Aangezien de feedback niet meer wordt gedaan door de docenten is het objectief en is de beoordeling degelijk. Daarenboven kunnen de oprichters van de modulen door middel van de statistieken aanpassingen maken in de modulen en de toetsing voor vervolg groepen.

2.2.2 Nadelen van automatische beoordelingssystemen

Hoewel automatische beoordelingssystemen verschillende voordelen hebben, hebben ze ook zo hun nadelen. Zoals besproken zijn er meerdere programma's ontwikkeld in de laatste 15 jaar, vooral voor programmeervakken. Vaak zijn ze bedoeld voor dezelfde taal, Java, met de doel de performance of competentie van de student beoordelen. Ihantola, Ahoniemi, Karavirta en Seppala (2010) geven aan dat er al systemen zijn die meestal al voldoen aan de meeste beoordelingsbehoeften. Toch worden jaarlijks nieuwe automatische beoordelingssystemen gecreëerd, vermoedelijk door de beschikbaarheid en levensduur van de applicaties. Volgens hen zijn deze veelal gemaakt voor een onderzoek of een bepaald vak en zijn deze automatische beoordelingssystemen ook goed genoeg om daarvoor te gebruiken, maar ze zijn niet geschikt voor de distributie. Hierdoor zijn er meer onderzoeken over nieuwe programma's dan wat wereldwijd wordt gebruikt.[21]

Het creëren van een opdracht of toets via een automatisch beoordelingssysteem vergt extra aandacht terwijl het simpel klinkt. Een kleine fout in de becijfering kan tot problemen leiden. Zo mogen bij programmeeropdrachten geen ambiguïteiten aanwezig zijn in de probleem specificatie als het systeem ze niet kan onderscheiden. Ook moeten de testgevallen van de code zo ontworpen worden dat alle afwijkingen aan het licht worden gebracht. De verleiding kan er zijn om de beoordelingsinstrumenten alleen maar op te nemen, doordat het relevante informatie geeft zonder de werklast te verhogen. Als een dusdanig systeem de leerlingen beïnvloedt, wegens gebrek aan ervaring en hulp, kan het leiden tot misverstanden en plagiaat. Op de lange termijn kost het minder tijd voor de onderwijzers, maar tijdens het ontwerpen moet grondig worden nagedacht. Dit proces zorgt ervoor dat de essentie van de opdracht meer aandacht krijgt dan de feedback en het cijfer.[1]

De nodige extra aandacht bij het creëren en de mogelijk eerste kennismaking met online toetsing zullen keerzijdes zijn die in de toetsing van dit onderzoek terugkeren. De becijfering wordt gedaan volgens de Taxonomie van Bloom, die verder zal worden behandeld in Hoofdstuk 2.4. Hierbij kunnen fouten optreden omdat het per persoon kan verschillen welke vraag tot welke niveau behoort. Het vermijden van de andere negatieve gevolgen worden besproken in de Methode.

De vele voordelen en weinig nadelen duiden de bruikbaarheid van de automatische beoordeling aan. Het geeft de mogelijkheid om de bespaarde tijd voor andere gebieden te gebruiken, het is altijd bereikbaar, de last bij grote en kleine groepen is overeenkomstig, het is goedkoop en het is objectief. Zulke voordelen kunnen nieuwe deuren openen voor verbetering in de educatie.

2.3 Prestatietest creatie voor educatieve doeleinden

De literatuur over test interpretaties met psychologische fundamenten en methoden is aan het groeien. Echter, terwijl deze van groot belang zijn worden deze test interpretaties nog weinig gebruikt in de praktijk.[40] Er bestaan vele soorten toetsingen met ieder een andere opzet en streven. Voordat er een test wordt gemaakt moet er fatsoenlijk nagedacht worden over het doel en de daarbij behorende psychologische fundamenten. Zo kan er doelgericht worden getoetst en hebben de cijfers meer significantie.

Gebruikmakend van een automatische beoordelingssysteem zal in dit onderzoek een prestatietest worden ontwikkeld. Een prestatietest is een proces waarin de mate van bepaalde kennis, vaardigheid of prestatie over een onderwerp wordt gemeten. In tegenstelling tot andere toetsingen heeft een prestatietest een focus op een specifiek onderwerp domein en verschilt het in zijn karakteristieke beoordeling en gebruik. De opgedane kennis door de populatie waarvoor de toets is bedoeld is het onderwerp domein, hierdoor wijken de vragen niet af van wat er is geleerd. Het gaat in het geheel over prestatie, zonder toetsing van andere onderwerpen zoals persoonlijkheid of irrelevante algemene kennis.[40] Hoewel het in dit onderzoek bedoeld is voor educatieve doeleinden is het ook toepasbaar in andere velden waar onder andere een licentie wordt behaald.[41]

Volgens Reynolds gaat het vaak om een toets voor een grote groep die met enige regelmaat toegediend wordt aan alle deelnemers.[40] Echter is dat niet de enige vorm en kan de afwerking van de test divers zijn: individuele opdrachten, groepsopdrachten, formatieve of summatieve toetsingen, meerkeuze tentamens, open vraag tentamens, essays, mondeling examens en dergelijke.[15]

Zodoende kan de stijl en discipline van de toetsing veranderen, zolang beslissing maken over het kennisniveau en progressie van de deelnemer het oogmerk is. Deze moet op een niveau zitten waarbij het duidelijk is dat de deelnemer bekwaam is, om er uiteindelijk cruciale beslissingen mee te maken. Daarnaast maakt deze toetsing het ook mogelijk de potentie van de deelnemer in te zien,

de kwaliteit van de cursus en de manier van lesgeven te beoordelen.[15][41] Zo kan er door de leerstrategieën en doelen te koppelen aan geïdentificeerde leerbehoeften of vaardigheidstekorten van studenten verbeteringen worden gebracht in de lesprogramma.[27]

De beoordelingen kan absoluut of relatief zijn. Bij de absolute vorm staan de criteria voor de becijfering vast terwijl het bij de relatieve vorm wordt vergeleken met andere examinandi die deelnemen aan dezelfde test.[15][41]

2.3.1 Soorten en karakteristieken van prestatietests

Prestatietests kunnen op basis van hun omvang en toediening op verschillende manieren worden gecategoriseerd. Zo kan er onderscheid worden gemaakt in de toediening en de omvang. De toediening kan op basis van een groep of individueel.[15] In de onderscheiding gebaseerd op de omvang zijn er drie groepen; informele, curriculum gebaseerde en formeel gestandaardiseerde toetsingen. Ieder heeft een eigen vorm van toediening, becijfering en doel. Voor de toetsing wordt er in dit onderzoek gebruik gemaakt van de curriculum gebaseerde benadering.

Een *curriculum gebaseerde meting* (CBM) is de meest volledige curriculum gebaseerde benadering die is ontwikkeld om de uitkomsten van de studenten te evalueren in rekenen en geletterdheid.[10] Deze vorm voorziet een valide en betrouwbare indicator van de voortgang van studenten in simpele academische vaardigheden en kan positieve impact hebben op de prestatie.[8] Het is een tussenvorm, ook wel "hybride" genoemd, van informele en formele toetsingen.[25] Hoewel het formaat en de inhoud kunnen verschillen, zijn er generalisaties mogelijk. Bij deze vorm gaat het om de reguliere metingen die zijn opgesteld met standaard procedures waarmee de voortgang duidelijk wordt afgebeeld.[41][25] Ook wordt de administratie, de richting gegeven aan studenten, becijfering en data benutting met standaard methodes toegepast.[8]

De herhaaldelijke toetsingen gedurende de periode hebben meerdere equivalente formuleringen. De criteria voor de doelen en de voortgangspercentages worden afgeleid uit nationale of lokale normen. De afnames zijn kort en eenvoudig, met valide en betrouwbare metingen van globale vaardigheden in de context. De scores zijn in de relatieve vorm en de verslagen geven de test score met het aantal goede en foute antwoorden. Met alleen een relatieve becijfering ten opzichte van medestudenten is de groei van de student niet duidelijk bij herhaaldelijke toetsing, dus een consistente becijfering is ook belangrijk.[10] De gegevens worden zo in kaart gebracht dat de voortgang eenvoudig te zien is, waardoor beslissingen maken mogelijk is met betrekking tot educatieve planningen en wijzigingen, evenals het verlenen van speciale diensten of gerichte interventies.[41][8] Deze mogelijke wijzigingen waren ook het doel tijdens de ontwikkeling van de CBM. De metingen en de evaluatie procedures moesten leiden tot routinewijs beslissingen over het mogelijk aanpassen van het programma. De kracht ervan zit in het snel kunnen beslissen of de interventie werkt of veranderd dient te worden [8], zo kunnen signalen van kleine fouten op tijd worden afgehandeld voordat grotere fouten optreden.[10]

Een nadeel van de besproken vorm van toetsing kan zijn dat het curriculum zo wordt aangepast dat alleen toets onderwerpen worden gedoceerd of dat studenten aanleren hoe ze deze toetsen moeten halen in plaats van academische vaardigheden te ontwikkelen.[25] Daarnaast moet er ook aandacht worden geschonken aan de ontwikkeling van de beste combinatie van procedures zodat er een effectieve groei is in verschillende type lerende, omdat niet bij ieder dezelfde vooruitgang wordt geboekt.[44] Volgens Fuchs, Fuchs, Hamlett, Phillips en Bentz (1994) is het laatste deels ook mogelijk door onderwijzers te informeren over CBM en aanbevelingen te geven met betrekking tot

het beoordelen.[16]

In tegenstelling tot informele toetsen in de klas die de beheersing van een enkele prestatie toetsen, wordt in elke CBM een grotere curriculum getest. Dus elke prestatie die verwacht beheersd te worden aan het einde van een curriculum, wordt op deze manier getoetst. Hierdoor is het een algehele indicator van de prestatie en voortgang binnen een curriculum.[8]

2.3.2 Gevolgde stappen bij het samenstellen van een prestatietest

Ondanks het feit dat de context kan veranderen, zijn er procedures geadviseerd voor het ontwikkelen en verzekeren van de kwaliteit en doelgerichte toetsing. In het geval van CBM is er een standaardisatie voor het verzekeren van de validiteit en de betrouwbaarheid. Dit gebeurt bij de administratie, de richting gegeven aan studenten, becijfering en data benutting. Hierdoor kan de data van een individuele deelnemer en van een groep worden vergeleken in verschillende tijden.[8]

Na een CBM test plot wordt de score in een grafiek geplot om de positie tot het gemiddelde te zien. Daarnaast wordt er gebruik gemaakt van een vooraf bepaalde regel om te beslissen of de instructies veranderd dienen te worden. Het is belangrijk dat dit van te voren wordt vastgesteld en constant wordt toegepast.[8] De leerdoelen moeten toenemen als een hoger percentage de toetsing haalt dan verwacht, of de instructies moeten veranderd worden als de voortgang lager is dan verwacht.[47]

Gebruik van computer applicaties bij data collectie maken de collectie, behoud en analyse van data eenvoudig en zorgen voor tevredenheid onder docenten.[8]

Bij het ontwikkelen van een test zijn er 21 universele stappen die vaak genomen worden voor het verzekeren van kwaliteit (zie Tabel 1). Door deze stappen na te gaan kan de test geëvalueerd worden, wat wordt gedaan in de Methode. Omdat het een proces is waarbij de scores een indirecte terugkoppeling zijn op de test kunnen stap 5 tot en met 21 telkens herhaald worden gedurende de periode dat de doelen en inhoudsnormen hetzelfde blijven.[15]

Voor formeel gestandaardiseerde toetsingen zijn er een aantal punten waarop gelet moet worden. Bij de afname geldt meestal de regel dat er zo min mogelijk afleiding is en dat alle benodigdheden aanwezig zijn. Daarnaast moet er een wederzijdse begrip tussen de examinator en de examinandi zijn met evenzeer en vlotte en nauwkeurige testadministratie.[41] Voor de afname moeten er protocollen worden opgesteld die bij de examinandi bekend worden gemaakt zodat iedere afname op dezelfde manier verloopt en zodat er fouten tijdens de uitvoering worden voorkomen. In de protocollen staan de toegestane tijd voor afname, de benodigdheden, de manier van vragen beantwoorden en de interactie met de examinator.[37] Deze standaarden zijn ook bruikbaar bij CBM.

De items kunnen zowel objectief (juist-onjuist, meerkeuze) of subjectief (essay vraag en korte antwoord) zijn. In dit geval zijn objectieve vragen het meest voor de hand liggend, omdat ze eenvoudig en goedkoop beoordeeld kunnen worden. Goed geconstrueerde gestandaardiseerde testen moeten valide, betrouwbaar en bruikbaar zijn, zoals alle psychologische of educatieve meetinstrumenten.[36] Hierbij kan er gedacht worden aan content validiteit, de toets moet meten wat er wordt onderwezen.[37] De vragen moeten goed doordacht zijn, want iedere vraagvorm kan een andere vaardigheid nodig hebben. Zo kunnen de antwoorden beïnvloed worden door taal- en cognitieve zwakheden, tekortkoming aan algemene kennis en lage motivatie.[41] Validiteit en betrouwbaarheid toegepast in deze situatie wordt uitvoerig besproken in Hoofdstuk 3.

Stap	Actie
1.	Stel behoefte en doel vast
2.	Bepaal het te testen domein
3.	Ontwikkel specifieke doelstellingen, inhoudsnormen
4.	Beslis over item en test specificaties, formaten, lengte, kosten
5.	Ontwikkel items en taken
6.	Analyseer de toegankelijkheid van de items
7.	Ontwikkel scoregidsen
8.	Voer item- en taakbeoordeling uit (redactie, geschiktheid, uitlijning, gevoeligheid)
9.	Proef-/veldtest items/taken/scoregidsen
10.	Voer analyse uit voor betrouwbaarheid en precisie van item- en testscores
11.	Beoordeel item- en taakprestatie
12.	Maak item pool aan
13.	Stel testformulier(en) samen volgens specificaties
14.	Ontwikkel richtlijnen en materialen voor testadministratie
15.	Stel prestatienormen vast
16.	Beheer operationele testformulieren
17.	Test score
18.	Evalueer het voorlopige item, de taak en de prestaties
19.	Rapporteer scores aan de juiste doelgroep
20.	Evalueer test, documenteer testcyclus
21.	Update item pool, herzien ontwikkelingsprocedures, ontwikkel nieuwe items/taken

Tabel 1: Universele stappen voor het verzekeren van kwaliteit in een toetsing. Vertaald van "Achievement Tests⋆" [31]

2.4 Richtlijnen voor items schrijven en becijfering van de prestatietest

2.4.1 Huidige richtlijnen voor het schrijven van items bij programmeervakken

Een automatisch beoordelingssysteem is pas effectief als het op een goede pedagogische manier is opgericht en deelnemers die beschikken over de benodigde kennis en deelnemers die hier niet over beschikken kan onderscheiden.[23] De algemene kaders voor het opstellen van een toetsing worden behandeld in 2.3.2, evenzeer zijn individuele vragen in een toets van belang voor een aannemelijke toetsing. De ontwikkelde vragen moeten de leerdoelen omvatten. Recente onderzoeken over examinering bij programmeeronderwijs hebben meegedeeld dat er grote variaties zijn in de stijl van de examens die ze hebben bestudeerd, echter is er geen overeenkomstige variatie in de benadering voor het schrijven van deze examens. [42]

Alhoewel er variaties zijn in de benadering voor het schrijven van de examens, hebben de toetsingen als overeenkomst dat alleen een gering aantal algeheel opnieuw wordt opgesteld. Uit het onderzoek naar elf programmeerdocenten van acht universiteiten verspreid over twee landen kwam naar voren dat het schrijven van examens grotendeels gebaseerd was op intuïtie, ervaring en pragmatiek, in plaats van educatieve kennis.[42] De meesten baseerden hun examens niet op pedagogische modellen, maar op eerdere examens waarvan ze dachten dat deze geldige beoordelingsinstrumenten waren. De toetsing baseren op oude toetsingen kwam volgens een aantal

geïnterviewde docenten doordat een nieuwe toetsing opstellen wordt gezien als tijdrovend en lastig.[42] De gegeven informatie toont aan dat er niet één gevalideerde benadering is om studenten te beoordelen.[42]

2.4.2 Gebruikte termen bij het opstellen van een toetsing

Hoewel informatica niet beschikt over een gevalideerde benadering om leerlingen te toetsen zijn er onderdelen in een toetsing die niet veranderen. Door de relevantie van begrip over deze termen, volgt hierna een korte uitleg van de termen.

Een toets bestaat uit *items*, ieder item bevat een uitspraak over een probleem waarop een antwoord gegeven moet worden. Een item kan na de uitspraak leeg zijn (open vraag), mogelijke antwoorden bevatten waaruit kan worden gekozen (meerkeuzevraag) of de antwoorden dienen gecombineerd te worden (matchvraag). Zo zijn er meerdere vormen voor antwoorden mogelijk. Het is hierbij van belang dat inhoudelijke kennis wordt gemeten.[23][39][33][46]

De uitspraak waaruit een item bestaat wordt een *stem* genoemd. Het is een uitspraak over het probleem bestaand uit circa twee zinnen. Naast de uitspraak over het probleem wordt het vraagstuk geformuleerd in de stem. De stem wordt op zijn beurt weer onderverdeeld in een *open en een gesloten stem*. In een open stem is er sprake van een onvolledige of inleidende uitspraak die samen met het gegeven antwoord een geheel moet vormen. Een gesloten stem is daarentegen een volledige vraag met een vraagteken op het einde.[23][39][33]

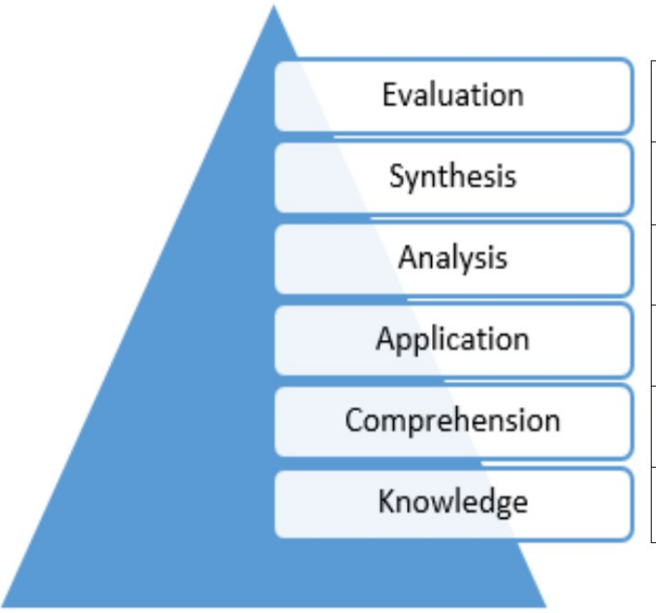
Naast de stem bevat een item ook *opties*, afhankelijk van de soort vraag. Het is een lijst met mogelijke antwoorden op de stem waaruit gekozen kan worden. Het correcte antwoord in de opties wordt de *sleutel* genoemd. De overige antwoorden worden aangeduid als *afleiders*. [23][39][33]

Buiten deze onderdelen is er ook een concept die vaak voorkomt in toetsingen. De onderwezen onderwerpen brengen ook *misconcepties* met zich mee. Een programmeer misconceptie houdt in dat een programmeer concept niet goed is begrepen, dit kan bijvoorbeeld komen door voorkennis uit andere vakgebieden zoals wiskunde en taal. Dit probleem heeft, soms langdurig, nadelig effect op de prestatie van de programmeur.[45] De meest gemaakte fouten of misconcepties bij een onderwerp worden vaak als afleider gebruikt, dit aangezien het een valkuil is voor de meesten terwijl de beste deelnemers het wel juist beantwoorden.

2.4.3 Taxonomieën en richtlijnen voor de becijfering

De taak van een automatisch beoordelingssysteem is nakijken en becijferen, echter om dit te kunnen doen moeten de hoeveelheid punten extern vastgesteld worden voordat de toetsing plaatsvind. Voor de garantie van de betrouwbaarheid en de consistentie is er een geconstrueerde scoregids nodig. Zo kunnen deze twee doelen worden bereikt tussen alle items in de toetsing en tussen de toetsingen van de andere modulen.[32] Deze gids maakt duidelijk hoeveel punten elk item waard is. Hoewel er verschillende automatische beoordelingssystemen zijn voor programmeer vaardigheden, is er geen gezamenlijke scoregids.

Door het ontbreken van een gezamenlijke scoregids is er gekeken naar gebruikte taxonomieën. Terwijl er vele taxonomieën bestaan, waaronder ook een speciaal gecreëerde voor programmeervakken, maken programmeerdocenten vaak gebruik van het cognitieve domein van de Taxonomie van Bloom.[42][26][17] Taxonomieën voor educatieve doelstellingen worden gebruikt voor het beschrijven van leerresultaten en prestaties bij beoordelingen.[17] De benaming cognitieve domein houdt in dat



Evaluation	Gedrag: toont het vermogen om te oordelen over de waarde van het materiaal voor een gegeven doel, gebaseerd op duidelijke criteria en redenen. Competentie: in staat om concept te gebruiken in probleemoplossingen zonder een voorbeeld.
Synthesis	Gedrag: combineert de delen die gecreëerd zijn in analyses tot een nieuwe geheel, anders dan wat het origineel was. Competentie: in staat om het correcte gebruik van het concept te verzekeren.
Analysis	Gedrag: verdeeld het materiaal in zijn samengestelde delen, zodat de organisatiestructuur kan worden begrepen. Competentie: in staat om de bedoeling van het concept in zijn context uit te leggen.
Application	Gedrag: gebruikt data, principes en geleerde theorie om een vraag in een nieuwe omgeving te beantwoorden. Competentie: in staat om gevallen waarin het concept toegepast kan worden te benoemen.
Comprehension	Gedrag: staat toe om het begrip over een zaak te demonstreren gebaseerd op zijn kennis. Competentie: in staat om het concept uit te leggen en toe te passen bij een vergelijkbaar probleem.
Knowledge	Gedrag: het terughalen van materiaal; het herinneren van de passende informatie is het enige wat nodig is. Competentie: in staat om gerelateerde concepten te benoemen.

Figuur 1: Taxonomie van Bloom met bijbehorend gedrag en competentie per categorie, vertaald uit ([26], combinatie van Figuur 1 en Tabel 1)

het gaat om de kennis en ontwikkeling van intellectuele vaardigheden.[5]

De Taxonomie van Bloom bestaat uit zes categorieën (zie Figuur 1), waarbij iedere categorie voortbouwt op de vorige. Elke categorie is een prestatieniveau dat wordt verwacht in de situatie waarbij het is toegepast. Degene die een hogere prestatieniveau kan uitvoeren kan naar verwachting ook lagere niveaus uitvoeren, want pas wanneer de eerste categorie onder de knie is, is de volgende mogelijk.[5] Fuller et al. (2007) beweert dat bij vakken zoals programmeren de levels niet goed geordend zijn voor het leertraject van beginnende programmeurs. Zo vinden sommigen dat het gebruik van kennis bij simpele problemen makkelijker is dan het beschrijven van de kennis terwijl het beschrijven de eerste categorie is bij de taxonomie.[17]

3 Methodologie

Met behulp van de theoretische kader (Hoofdstuk 2) zijn verschillende onderwerpen aan het licht gebracht. De huidige situatie en het belang van integratie van programmeren in vakken is besproken. Automatische beoordelingssystemen zijn onderzocht, wat met de huidige Covid-19 situatie nog meer van belang zal zijn omdat educatie nu vaker online wordt. Ook zijn de huidige richtlijnen in opstelling toetsingen en scoregidsen behandeld. In de methodologie wordt er ingegaan op hoe de literatuur wordt gebruikt om het veldonderzoek ten uitvoer te brengen.

3.1 Type onderzoek

De behoefte aan een doordachte test vanuit de VHTO resulteerde in een praktisch actieonderzoek met gebruik van kwalitatief en kwantitatief onderzoek.[9] Dankzij dit onderzoek wordt een praktische oplossing voor het probleem ontwikkeld.

In dit onderzoek is gebruikgemaakt van literatuuronderzoek en veldonderzoek. Het literatuuronderzoek uitgeschreven in hoofdstuk twee was bedoeld voor de ontwikkeling van de toetsing, hierbij zijn er verschillende papers gebruikt. Na het ontwikkelen hiervan is er een veldonderzoek uitgevoerd om de gecreëerde toetsing te controleren.

In de methode wordt eerst uitgelegd welke stappen zijn ondernomen voor de creatie van de toetsing. Deze toetsing wordt uiteindelijk gecontroleerd door een groep deskundigen tussen 10 tot 21 mei 2020.

3.2 Module Nederlands

De Module Nederlands is een deel van de lessenreeksen van het lesprogramma Computer Science Certificate. Hierbij wordt basiskennis programmeren in Python onderwezen op het voortgezet onderwijs, met name aan eerste en tweedejaars havo/vwo leerlingen. Door kennismaking met het programmeren en de opbouw van zelfvertrouwen erin worden de leerlingen gestimuleerd om in de bovenbouw het keuzevak Informatica te kiezen.[49]

Er zijn drie reeksen met ieder zes lessen, deze zijn Nederlands, kunst en geschiedenis. Het vak wordt geïntegreerd in coderen. Dankzij deze reeksen leren leerlingen het programmeren in verschillende contexten te plaatsen. Zo leggen ze in de eerste module een verband tussen taal en programmeren door interactieve verhalen te coderen. Hiervoor leren ze verschillende concepten van Python, zoals de printstatement waarmee ze de zinnen kunnen printen en de if-else statement waardoor de inhoud van het verhaal kan veranderen (voor alle concepten zie Tabel 2).[49] Hierdoor is het mogelijk voor de leerlingen om ook na de module eigen verhalen te coderen.

Dit onderzoek vormt een basis voor de toetsing van de module Nederlands. Door de stappen bij de creatie van de toetsing te benoemen kunnen de vervolgmodules dezelfde stappen volgen waardoor er een samenhangend geheel wordt gevormd op het gebied van toetsing.

3.3 Opstelling toetsing

Als kader voor de creatie van de toetsing zijn de universele stappen uit Tabel 1 gebruikt. Voor elke stap is er gebruikgemaakt van verschillende onderdelen uit het literatuuronderzoek. Niet alle stappen zijn toepasselijk voor dit onderzoek, waardoor een deel van de stappen eruit is gehaald (stap 9, 10, 11, 16, 17, 20) of is gewijzigd (18, 21). De stappen die eruit zijn gehaald zijn alleen mogelijk met een testgroep, zoals stap 9 ("Proef-/veldtest items/taken/scoregidsen") en stap 11 ("Beoordeel item- en taakprestaties"). Ook zijn een aantal van de stappen eerder gebruikt dan wat er in de figuur wordt aangegeven of zijn ze samengevoegd met een andere stap (5 met 12, 6 met 8, 14 met 19, 18 en 21). Stap 5 en 12, ontwikkelen van items en item pool aanmaken, zijn bijvoorbeeld samengevoegd voor een sneller proces en overzicht van alle items. Vanaf stap 5 ("Ontwikkel items en taken") is het mogelijk om het proces constant te herhalen, zo een proces zorgt voor een verbeterde toetsing.

4 Resultaten

De stappen van Tabel 1 zijn, op de manier zoals beschreven in de methodologie, toegepast. In dit onderdeel staan alle resultaten die dankzij de methode zijn verkregen.

1. **Stel behoefte en doel vast**

   De behoefte komt vanuit de VHTO, ze hebben een automatische toetsing van Python vaardigheden nodig die zijn onderwezen in de module Nederlands. Het doel hierbij is op een makkelijke en pedagogisch-gefundeerde manier de leerlingen die de lessen hebben begrepen te onderscheiden van diegene die het niet hebben begrepen, om ze zo op een eerlijke manier van een certificaat te voorzien.

2. **Bepaal het te testen domein**

   Doordat het gaat om een curriculum gebaseerde methode komen de leerdoelen uit het curriculum, in dit geval de module Nederlands van de VHTO. Deze leerdoelen, met de bijbehorende onderdelen die worden onderwezen, zijn te vinden in Tabel 2

   Door de nadelen worden de mogelijke misconcepties in de toetsing geïntegreerd, zo kan er een inzicht worden verkregen in de aanwezigheid van de misconcepties onder de leerlingen. Daarop gebaseerd kan de module worden aangepast.

   De misconcepties zijn in Tabel 3 te zien. De misconcepties die kunnen ontstaan onder de leerlingen passend bij de leerdoelen, die meer dan een keer worden geciteerd in het onderzoek van Sorva (2012) en voorkomen in het onderzoek van Boulay (1986) zijn in de toetsing en in de tabel opgenomen.[6][43]

3. **Ontwikkel specifieke doelstellingen, inhoudsnormen**

   Alhoewel de module op alle niveaus op de middelbare school kan worden gegeven, was de vraag gericht op havo/vwo eerste- en tweedejaars. Hierdoor zit de leeftijd veelal tussen de twaalf en veertien jaar. Het niveau van de toetsing wordt hierop gebaseerd. Een te hoog niveau zal het halen van een certificaat onder de leerlingen negatief beïnvloeden.

   Daarnaast is het belangrijk dat er geen sprake is van bias. Bias houdt in dat externe factoren met negatieve invloed op de uitkomst van het onderzoek aanwezig zijn. Dit kan het geval zijn wanneer een leerling door iets in de toetsing negatief wordt beïnvloed, wat op zijn beurt weer effect heeft op het resultaat. Voorbeelden van zulk soort factoren zijn bijvoorbeeld geslacht, cultuur, inkomen en afkomst. Bias kan komen door woordkeuze, een uitdrukking of een situatie dat wordt geschetst in een item.[12]

   Naast een passend niveau en bias speelt correcte formulering van de vragen en het nut van de vragen een belangrijke rol. Een verkeerde formulering kan net als bias de leerlingen in de war brengen en het eindcijfer negatief beïnvloeden.

   De vragen moeten passen bij de leerdoelen en de misconcepties. Hierdoor is er van te voren vastgesteld om alle onderdelen van de leerdoelen en misconcepties in de toets te integreren. Dit is gedaan door middel van specifieke vragen gericht op één leerdoel te stellen, maar soms ook vragen die meerdere leerdoelen tegelijk behandelen.

4. **Beslis over item en test specificaties, formaten, lengte**

   Door het missen van een gevalideerde benadering om studenten te toetsen bij programmeervakken (Hoofdstuk 2.4), is er in dit onderzoek gebruikgemaakt van richtlijnen uit andere vakgebieden, zoals natuurkunde, Engels en verpleegkunde. Ook zijn de richtlijnen van TIMSS

Leerdoel	**Onderdelen**
1. Commentaar schrijven	1. Commentaar begint met # 2. Doel: - Boven in de code schrijven wat een programma doet - Uitleg bij een stukje code geven - Een regel code eventjes 'uit' zetten
2. Variabelen / Datatype string	1. Declaratie: <var>= '..' 2. Gebruik: <var>(zonder aanhalingstekens) 3. Voorspellen wat de code met een variabele erin doet 4. Dezelfde variabele later opnieuw declareren in de code
3. Lijst	1. Declaratie: <var>= 2. Lijst begint en eindigt bij declaratie met rechte haken: <var>= [...] 3. Iedere woord in de lijst moet tussen aanhalingstekens 4. Komma's tussen de woorden in de lijst 5. Gebruik: - Rechte haken: <var>[<getal>] - Geen aanhalingstekens 6. Lijst aanroepen mogelijk met getallen tussen: 0 >= <getal>=<(aantal woorden -1)
4. Gebruikersinput	1. Ronde haakjes achter input: input() 2. Variabele gelijk stellen aan input 3. Invoer intypen op invoerscherm
5. *If-Else statement*	1. Lezer kan meebeslissen over het verloop van het verhaal 2. Voor vergelijking in de if-statement: if <var1>== <string>/<var> 3. Achter de regel met if en else: ":" 4. De regels na de if en else statement 2 spaties inspringen
6. *Foutmeldingen*	Herkennen van de volgende fouten: 1. Een aanhalingsteken vergeten: "SyntaxError: EOL while scanning string literal" 2. Beide aanhalingstekens vergeten: "NameError: name <string>is not defined" 3. Ronde haakjes vergeten: "SyntaxError: unexpected EOF while parsing" 4. een haakje bij input() vergeten: "SyntaxError" 5. Beide haakjes bij input() vergeten: geen error maar een rare output 6. Een ":"of "="vergeten bij if-else statement: "SyntaxError" 7. De twee spaties vergeten in de regel na if/else statement: "IndentationError" 8. Fout bij aanwijzer/index van lijst
7. *Print*	1. Gebruik van ronde haakjes: print() 2. Aanhalingstekens voor en achter het woord, zolang het geen variabele is 3. Komma's tussen de woorden, mag ook zonder, zolang het geen variabele is 4. Werken met printopdrachten die onder elkaar staan

Tabel 2: Leerdoelen van de Module Nederlands met alle onderdelen die erbij worden onderwezen

Leerdoel	**Misconceptie**
Variabelen / Datatype string	5. Variabele kan meerdere waardes bevatten 6. Declaratie werkt de andere richting op 7. Normale taak beinvloedt de waarde van de variabele 8. Uitvoer volgorde van de code
Gebruikersinput	4. Effect van de input-functie op uitvoer van het programma
If-Else statement	5. False bij if eindigt programma als er geen branch is 6. Als statement false is start de code geheel van voren af

Tabel 3: Misconcepties die mogelijk kunnen worden ontwikkeld bij de leerdoelen (de bijbehorende getallen zijn een vervolg op dezelfde onderdelen in Tabel 2)[6][43]

benut. Het TIMSS stelt wereldwijd toetsingen op over wiskunde- en wetenschapskennis van studenten.[46]

Uit de vijf artikelen die als referentie zijn gebruikt zijn de regels die bij minstens twee artikelen overeenkomen en die toepasselijk zijn bij programmeervragen overgenomen. Als laatst worden de conclusies die getrokken zijn in het onderzoek van Haladyna, Downing, & Rodriguez (2002) bestudeerd, bij een tegenspraak in dit artikel over de overeenkomstige richtlijnen gegeven in de eerder genoemde artikelen, wordt hun conclusie meegenomen. Dit omdat de onderzoek van Haladyna, Downing, & Rodriguez (2002) gebaseerd is op de validiteit van richtlijnen terwijl de anderen zijn gebaseerd op het gebruik ervan.[18]

In Tabel 4 worden de overeenkomstige en nodige richtlijnen, verkregen uit de artikelen, aangegeven voor de item creatie van dit onderzoek. De regels zijn onderverdeeld in hoofdgroepen. In de eerste groep worden de algemene regels waar de test aan moet voldoen behandeld, daaropvolgend worden de richtlijnen voor opties, items en afleiders onder de loep genomen. Elke item hoort zich te houden aan deze richtlijnen. Omdat programmeren een ander vakgebied is kunnen in sommige gevallen de vragen van de formaat afwijken. Zo kunnen er codes zijn in de opties die pas bij het zien van de optie te beantwoorden zijn in plaats van wat richtlijn 2 van de stem aanbeveelt. Daarbij komt ook dat de code bevattende opties niet de aanbevolen interpunctie kunnen hebben. Hierbij moet zorgvuldig nagedacht worden tijdens het schrijven van de item. Daar waar het niet mogelijk is om dezelfde interpunctie formaat aan te houden, zal er worden afgeweken van de richtlijn.

Zoals eerder ook aangegeven moeten de vragen zonder bias, nuttig, op niveau, correct geformuleerd en relevant zijn. Zo worden mogelijke vragen tijdens de toetsingen vanuit de examinandi ook voorkomen.

De lengte van de toetsing is 45 minuten, wat vaak een lesuur is op middelbare scholen. Het aantal vragen in de toetsing maakt niet uit zolang het past binnen de tijd. Verschillende vormen van vraagstellingen zijn mogelijk zolang het automatisch door het beoordelingssysteem kan worden geaccepteerd, er geen extra controle nodig is en kan worden gemaakt door beginnende programmeurs.

Er is gekozen voor verschillende vraagstellingen, deze zijn: meerkeuze, meerdere antwoord, open, combinatie en juist-onjuist vragen. Bij elke vorm zijn er buiten de richtlijnen van items schrijven nog extra richtlijnen. De bijpassende richtlijnen voor deze vraagvormen in dit

Onderdeel in test	Richtlijnen
Algemeen	1.Houd rekening met de maximale duur van de test. 2.Houd rekening met de mogelijke vraagstukken, in dit geval bij het automatische beoordelingssysteem. 3.Vermijd moeilijk taalgebruik die de test onnodig complex maakt. 4.Voorkom onnodige extra informatie dat overbodige leestijd kost. 5.Voorkom het testen van te gedetailleerde en/of te algemene kennis.
Item	1.Houd rekening met de tijd nodig voor de item. 2.Houd rekening met de leerdoelen. 3.Houd rekening met het niveau van de doelgroep. 4.Vermijd mogelijke vooroordelen/bias (cultuur, gender, inkomensniveau). 5.Ontwikkel voor elk item een beoordeling. 6.Geef alle informatie die nodig is om de vraag goed te kunnen beantwoorden. 7.Test met elk item een specifieke kennis van de leerdoelen. 8.Voorkom afhankelijkheden tussen items.
Stem	1.Voorkom indirectheid of dubbelzinnigheid door specifiek te zijn, - Geef het zelfstandige naamwoord aan bij "Welke van de volgende <Actie>.." vragen. - Vermijd "Welke van de volgende opties klopt (niet) over ....". 2.De stem moet op zichzelf al te beantwoorden kunnen zijn, zonder de mogelijke opties te lezen. 3.De bedoeling van de test is niet om iets aan te leren, voorkom op deze manier geïntroduceerde vragen. 4.Schrijf geen negatief verwoorde vragen, het gaat om het testen van kennis en niet om alertheid. - Als het toch moet, laat dit woord opvallen, onderstreep of maak het bijvoorbeeld dikgedrukt. - Voorkom dubbele ontkenning in de opties door ze allemaal positief te maken. 5.Schrijf bij vragen die omschrijving nodig hebben eerst de omschrijving dan de vraag. 6.Vermijd aanwijzingen naar het antwoord. - Voorkom herhaalde term in stem en goede antwoord, of herhaal de term ook bij een afleider. 7.Laat woorden, die niet over het hoofd mogen worden gezien, opvallen. 8.Als het geen universeel geaccepteerde antwoord is, gebruik dan "van de volgende" of iets dergelijks. 9.Voorkom vragen die met een verkeerde methode hetzelfde antwoord geven. 10.Beginnen met "wat" wanneer er maar één mogelijke antwoord is. 11.Beginnen met "welke" wanneer er meerdere mogelijke antwoorden zijn, maar de focus ligt op het enige juiste antwoord in de opties. 12.Vermijd herhaling aan het begin van elke optie, door het in de stem op te nemen. 13.Formaat: - Open stem om zin te completeren: a.stem heeft geen interpunctie op het einde. b.eerste woord van iedere optie begint met kleine letters. c.iedere optie eindigt met een punt. d.alle opties sluiten grammaticaal aan bij de stem. - Open stem met een term of een woord als antwoord: a.stem eindigt met een dubbele punt. b.geen interpunctie aan het einde van de opties. - Gesloten stem: a.eindigt met een vraagteken. b.eerste woord van iedere optie begint met kleine letters, tenzij elke optie een complete zin is. c.bij een complete zin eindigt iedere optie met een punt. d.alle opties sluiten grammaticaal aan bij de stem.
Opties	1.Drie/vier opties zijn genoeg bij meerkeuzevragen. 2.De sleutel moet de enige en/of beste goede antwoord zijn. - Er moet bewijs zijn dat het de enige of het beste antwoord is. 3.Alle opties moeten grammaticaal in overeenstemming zijn met de stem. - Dezelfde stijl en vorm, - Dezelfde lengte, - Dezelfde hoeveelheid detail, - Bij minimaal één afleider. 4.Varieer de locatie van de sleutel afhankelijk van het aantal opties. 5.Zet de opties in een logische of numerieke volgorde. 6.Zorg dat er geen overlap is tussen antwoorden, ze moeten onafhankelijk zijn. - Bij numerieke antwoorden moeten de antwoorden geen overlap hebben om meerdere sleutels te voorkomen. 7.Vermijd de "geen van de antwoorden" optie. 8.Vermijd de "alle antwoorden kloppen" optie. 9.Houd de opties positief, voorkom woorden zoals "niet/geen". 10.Geef geen aanwijzing in de opties; - Woorden zoals: altijd, nooit, absoluut, helemaal. - Termen die letterlijk ook in de stem voorkomen. - Sleutel die opvalt.
Afleiders	1.Mogelijke afleiders: - Vaak gekozen uit veelgemaakte fouten of verkeerd begrepen aspecten. - Of gekozen uit een optie dat correct is maar bij een andere vergelijkbare probleem hoort, dus hier niet van toepassing is. - Mogelijk kan een afleider ook gekozen worden uit een overdrijving van een normaliter correct antwoord. 2.Stijl: - Als de opties een lijst bevatten, verspreid de termen in de juiste antwoord over de afleiders. Herhaal term in afleider in andere afleiders. - Gebruik geen termen zoals "nooit", "altijd", "soms", "vaak", etc. Dit verzwakt de afleider. - Gebruik geen afleiders die weinig verschillen met de sleutel.

Tabel 4: Richtlijnen voor het schrijven van onderdelen van een test, vanuit [23][32][39][33][18]

onderzoek, verkregen van het boek geschreven door Hawkes, Lindquist en Mann (1936), zijn als volgt.[19]

Meerkeuzevragen zijn de meest waardevolle en algemeen toepasbare vraagstellingen. Het is een directe vraag gevolgd door opties waarbij alleen één het goede antwoord is of dat meerdere antwoorden mogelijk zijn, maar de sleutel is de beste optie. Het kan ook een incomplete uitspraak zijn die compleet wordt gemaakt dankzij de goede optie. De opties bestaan vaak uit drie of vier mogelijkheden; er zouden potentieel meer mogelijkheden kunnen zijn, maar dit maakt de kwaliteit van de vraag niet beter. Deze vorm benodigd extra aandacht om geen hints in de opties of stem mee te geven die tot het goede antwoord leiden. De volgende richtlijnen voor deze vraagstellingen zijn van belang, naast de eerder genoemde richtlijnen voor items:

- Bij een incomplete uitspraak is het in te vullen gedeelte op het einde beter dan midden in.
- Alle opties moeten aannemelijk zijn zodat ze allemaal eerst worden gelezen. Een optie die door geen enkele deelnemer wordt gekozen is niet aannemelijk.

Voor combinatievragen zijn er vele toepassingen mogelijk. Directe vragen combineren met antwoorden, incomplete compleet maken met de opties of combinaties van woorden in beide groepen. De aantallen kunnen ook verschillen, de ene kolom kan gelijk zijn aan de andere maar een van de twee kolommen kan ook langer zijn waardoor sommige opties van de andere kolom niet worden gekozen of juist vaker gekozen.

- Vijf of zeven opties is het beste en twaalf is het maximum, bij een grote aantal zijn de examinandi te veel bezig met het zoeken naar het goede antwoord.
- Meer items dan opties is niet aanbevolen, maar meer opties dan items wel. Zo is er geen kans om de laatst overgeblevenen toe te wijzen.
- Er moet van te voren goed uitgelegd worden wat voor matching het betreft.

Juist-onjuist vragen:

- Het aantal juiste en onjuiste antwoorden moeten ongeveer gelijk zijn.
- Kleine foute elementen moeten niet verstopt zitten in een zin waar het algemeen correcte wordt benadrukt, leerlingen zien de kleine foute element over het hoofd.
- Het belangrijkste deel moet over het algemeen aan het einde worden weergegeven.
- Het valse element in een verklarende uitspraak, als die er is, moet alleen in de verklaring staan.
- Zelfde taalgebruik als de lesstof moet worden vermeden.

Buiten de afbakening van de test specificaties, formaten en lengte is het ook belangrijk om te vermelden wat nodig is tijdens de toetsing. Wegens het feit dat het online wordt gehouden is het voor de hand liggend dat er een computer aanwezig is. Bovendien is het hebben van pen en papier handig voor leerlingen die aantekeningen willen maken. Als er wordt gewerkt met laptops is het nodig dat ze minimaal 45 minuten mee kunnen en niet in de tussentijd opgeladen hoeven te worden.

item [nummer] onderdeel [nummer van leerdoel/misconceptie]	
bedoeling	
sleutel	
Bloom Taxonomie	
redenering foute antwoorden	

Figuur 2: Onderdelen die bij elk item ingevuld dienen te worden

5. **Ontwikkel items en taken, en maak item pool aan**

   Dankzij de leerdoelen zijn de onderwerpen van de toets afgebakend. Met behulp van de oefenopgaves en eigen voorkennis is er voor iedere onderdeel van de leerdoelen en misconcepties één tot drie vragen ontwikkeld. Sommige items bevatten meerdere onderdelen en sommige zijn specifiek gericht op een onderdeel. Dit zorgde voor een totaal aantal van 34 vragen te vinden in Appendix A. De gemaakte oefenopgaves zijn als voorbeeld gebruikt om voor leerlingen niet een geheel andere vraagstelling te introduceren waardoor de toetsing moeilijker zou kunnen worden.

   Deze items komen in een item pool, die later wordt beoordeeld door experts. Zo worden meteen alle vragen beoordeeld in plaats van na het beoordelen nieuwe vragen aan te maken die nog een beoordeling behoeven.

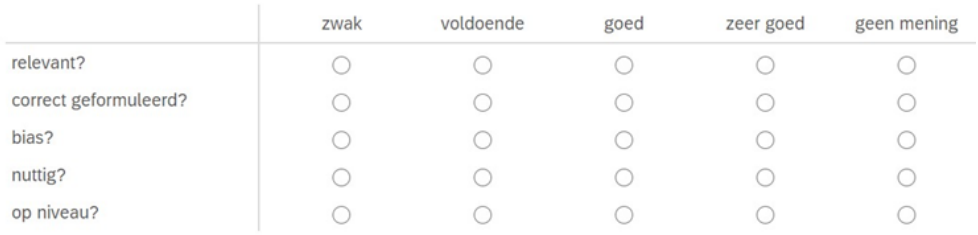
   Voor elk item is tijdens de ontwikkeling een tabel gecreëerd met aanvullende informatie over het item, zoals te zien in Figuur 2. Deze tabel bevat de volgende informatie: het item (stem en opties), het behandelde leerdoel, de bedoeling, de sleutel, de bijbehorende Bloom Taxonomie en redenering voor alle afleiders. Deze tabel vereenvoudigd de taak van de testmaker, de volgende aspecten zijn ongecompliceerd te zien: controle of alle leerdoelen zijn behandeld, of de locatie van de sleutels varieert en of alle niveaus van de taxonomie zijn verwerkt. Bovendien wordt de kwaliteit en nut van de vraag uitgediept tijdens het schrijven van de bedoeling van het item en de beredenering van de afleiders.

6. **Ontwikkel scoregidsen**

   Hoewel er verschillende automatische beoordelingssystemen zijn voor programmeer vaardigheden, is er geen gezamenlijke scoregids (Hoofdstuk 2.4.3). Het ontbreken van een scoregids, in combinatie met het belang van consistentie heeft ertoe geleid dat het becijferen van de items in dit onderzoek gebeurt volgens de categorieën van Bloom. Hiermee kan er ook gezorgd worden voor verschillende prestatieniveaus tussen items binnen een enkele toetsing.

   Elk item behoort tot een van de zes categorieën van deze taxonomie. Een vraag waarvan het niveau past bij de eerste categorie krijgt één punt bij het goed beantwoorden. Hierdoor kan er dus bij elke toetsing het totaal aantal te verdienen punten veranderen; het gaat hierdoor niet meer om een totaal van honderd punten.

   Bij juist-onjuist vragen wordt de vraag becijferd met het verschil tussen kloppende en fout beantwoorde antwoorden. Als een punt niet is beantwoord krijgt de kandidaat noch



Figuur 3: Vijf inhoudsnormen waar experts op hebben beoordeeld

punten noch aftrek.[19] Meerkeuzevragen met meerdere sleutels worden daarentegen becijferd door de categorie nummer van Bloom te delen door het aantal goede opties.

7. **Voer itembeoordeling uit (redactie, geschiktheid, uitlijning, gevoeligheid)**

Voor de itembeoordeling zijn er verschillende deskundigen benaderd. Dit is gedaan via persoonlijke contacten, de Slackgroep van de docenten die eerder de modulen hebben gegeven in hun klassen en de Slackgroep van de Programming Education Research Lab (PERL). Tijdens de benadering werd er nadrukkelijk gezocht naar informatica docenten of docenten met kennis van programmeren werkend op een middelbare school havo of vwo, of studenten/academici op het gebied van informatica. Dit om representativiteit te kunnen creëren. Kennis van programmeren was van belang zodat de test werd begrepen en mogelijke fouten gevonden konden worden. Docenten waren van belang omdat ze ervaring hebben met het maken van toetsingen en eventueel het niveau van de leeftijdsgroep kunnen inschatten. Studenten informatica waren van belang door hun kennis over het programmeren en ervaring van maken van programmeertoetsen.

Vijf van de benaderde deskundigen hebben de gehele itempool beoordeeld en een persoon heeft de itempool gedeeltelijk beoordeeld. In Figuur 2 uit het onderzoek van Dumas, Sorze en Virzi (1995) is te zien dat bij ongeveer zes experts een groot deel van de problemen in het system worden gevonden.[13] Dit als referentie nemend is er genoegen genomen in dit onderzoek met zes reacties. Hoewel alle antwoorden dezelfde waarden hebben is de expertise van belang voor de validiteit van het onderzoek. De groep bestond uit bachelor studenten informatica, een postdoc, een PhD kandidaat op het gebied van programmeren en gender, en een universitaire hoofddocent Informatica met ervaring in lesgeven op middelbare scholen.

De items zijn online opgestuurd en beoordeeld via een automatisch beoordelingssysteem Qualtrics.[38] De item met de sleutel(s) uit Appendix A zijn hierin beoordeeld op vijf onderdelen die als inhoudsnormen zijn vastgesteld in onderdeel 3, deze zijn te zien in Figuur 3.

Om de betrouwbaarheid te waarborgen en verschillende interpretaties te vermijden is de introductie van de vragenlijst voorzien van een uitleg over de beoordeelde onderdelen.

- relevant: is de vraag passend bij de leerdoelen?
- correct geformuleerd: zijn er taalfouten of onduidelijke zinnen?
- bias: is er een oneerlijke begunstiging van iemand gebaseerd op geslacht/leeftijd/etniciteit/e.d. (zeer goed: geen bias, zwak: bias aanwezig)

Item	Zwakke punt
3	Formulering, 1/3 van de deskundigen
4	Laag niveau, 1/3 van de deskundigen
5	Nut, 1/3 van de deskundigen
8	Formulering, 1/2 van de deskundigen Relevantie en nut, 1/3 van de deskundigen
10	Formulering en nut, 1/3 van de deskundigen

Tabel 5: Items die naar voren kwamen in de expert review met de bijbehorende zwakke onderdelen

- nuttig: heeft de vraag een praktische voordeel om begrijpende leerlingen en niet begrijpende te onderscheiden?
- op niveau: is het duidelijk voor leerlingen havo/vwo eerste en tweede klas (twaalf - veertien jaar)?

Aan het einde was het ook mogelijk om aanvulling op de toets te geven in eigen woorden. Hierbuiten zijn er ook reacties ontvangen via e-mail en telefoongesprekken. De resultaten uit dit onderzoek hebben invloed gehad op de vorming van een nieuwe item pool en de uiteindelijke toetsing. Dit kwantitatief onderzoek resulteerde in het wegschrappen van vijf items. Als uit de resultaten bleek dat één van de vijf punten waarop gelet wordt door één derde van de deskundigen als zwak werd beoordeeld, is de vraag eruit gehaald. Eerst was het de bedoeling het aan te passen gebaseerd op de punt van zwakte, maar vanwege een kleine aantal items die tot deze groep behoorden zijn de vragen meteen eruit gehaald (zie Tabel 5).

Als extra commentaar over het geheel is het gebrek aan consensus tussen de opties van de vraagstellingen benadrukt. Daarnaast zijn bij sommige vragen in woorden geformuleerd wat er veranderd kan worden, zoals: betere formulering in de stem, een duidelijke bias die aanwezig is en zwakke opties die verandert dienen te worden. Deze terugkoppelingen zijn meegenomen bij het updaten van de item pool.

8. **Stel testformulier(en) samen volgens specificaties**

De test wordt op een automatisch beoordelingssysteem uitgevoerd. Er zijn voor dit onderzoek verschillende passende automatische beoordelingssystemen onderzocht. Er is gekozen voor een web-based automatisch beoordelingssysteem, dit omdat het downloaden op schoolcomputers voor mogelijk eenmalige toetsing redundant zal zijn. Ook zijn de meeste docenten niet gespecialiseerd in zulke programma's. Daarnaast was het van belang dat het een veilige omgeving is qua gebruik en data; hierbij is er gekeken naar de certificaten die de sites bezitten. Tussentijds opslaan van de antwoorden en data analyse van de antwoorden zijn ook van belang.

Omdat dit onderzoek zich focust op de toetsing van de eerste module en de doelgroep grotendeels voor het eerst kennis maakt met programmeren is hier niet gekozen voor een statische of dynamische toetsing. In plaats daarvan is er een focus gelegd op de simpele automatische beoordelingssystemen die werken met bijvoorbeeld meerkeuze en match vragen. Echter zal bij de toetsing van de vervolgmodules statische en dynamische beoordeling wel

van pas komen, en dan met name geheel automatische beoordelingen in plaats van semi-automatische. Dit zodat er geen subjectiviteit vanuit docenten bij het beoordelen ontstaat.

De meest toepasselijke sites zijn Qualtrics, SoGoSurvey, TypeForm, Survey Gizmo en LimeSurvey. Voor dit onderzoek is er gekozen voor Qualtrics door zijn oneindige mogelijkheid aan vragen en uitgebreide data analyse.

De toetsing is alleen mogelijk op uitnodiging (*Qualtrics* → *Distributions* → *Emails*) en heeft een wachtwoord; zo is het afgeschermd voor anderen en wordt ongeautoriseerde verspreiding voorkomen. In de testformulier komen eerst vragen over de deelnemer: naam, klas, niveau, school, leeftijd en gender. Daarna komen de items. Voor een duidelijk overzicht zijn de vragen over pagina's verspreid met op iedere pagina maximaal zes vragen, waarbij de voortgang ook gemakkelijk te zien is. Iedere vraag heeft zijn vraagnummer en leerlingen kunnen terug naar eerdere vragen zoals een gewoonlijke toetsing. Daarnaast kunnen ze ook opslaan en binnen een uur weer verder gaan zonder dat de antwoorden definitief worden opgeslagen.

De gekozen items met becijfering worden in het systeem gezet. De becijfering (*Tools* → *Scoring*) wordt gedaan volgens het geheel en op categorie van leerdoelen. Zo kan er analyse worden gedaan wanneer het nodig is.

9. **Ontwikkel richtlijnen en materialen voor test administratie en rapportage**

   Aan het einde van de toetsing worden de resultaten van de toetsing gerapporteerd per leerling. Deze worden per mail opgestuurd naar de eindverantwoordelijke. Om de score automatisch te kunnen mailen moet er in de Survey Flow verschillende onderdelen worden toegevoegd (zie Figuur 4). Deze instellingen worden gebruikt in de condities van de triggers (*Tools* → *Triggers* → *Email Triggers*) (zie Figuur 5). Er zijn twee triggers aanwezig, gebaseerd op de eindscore. Daarin staat of de deelnemer het heeft gehaald of niet met het eindresultaat en punten per leerdoel. Al de data wordt ook opgeslagen in het systeem, waar altijd in kan worden gekeken voor individuele analyse of vergelijking met medestudenten.

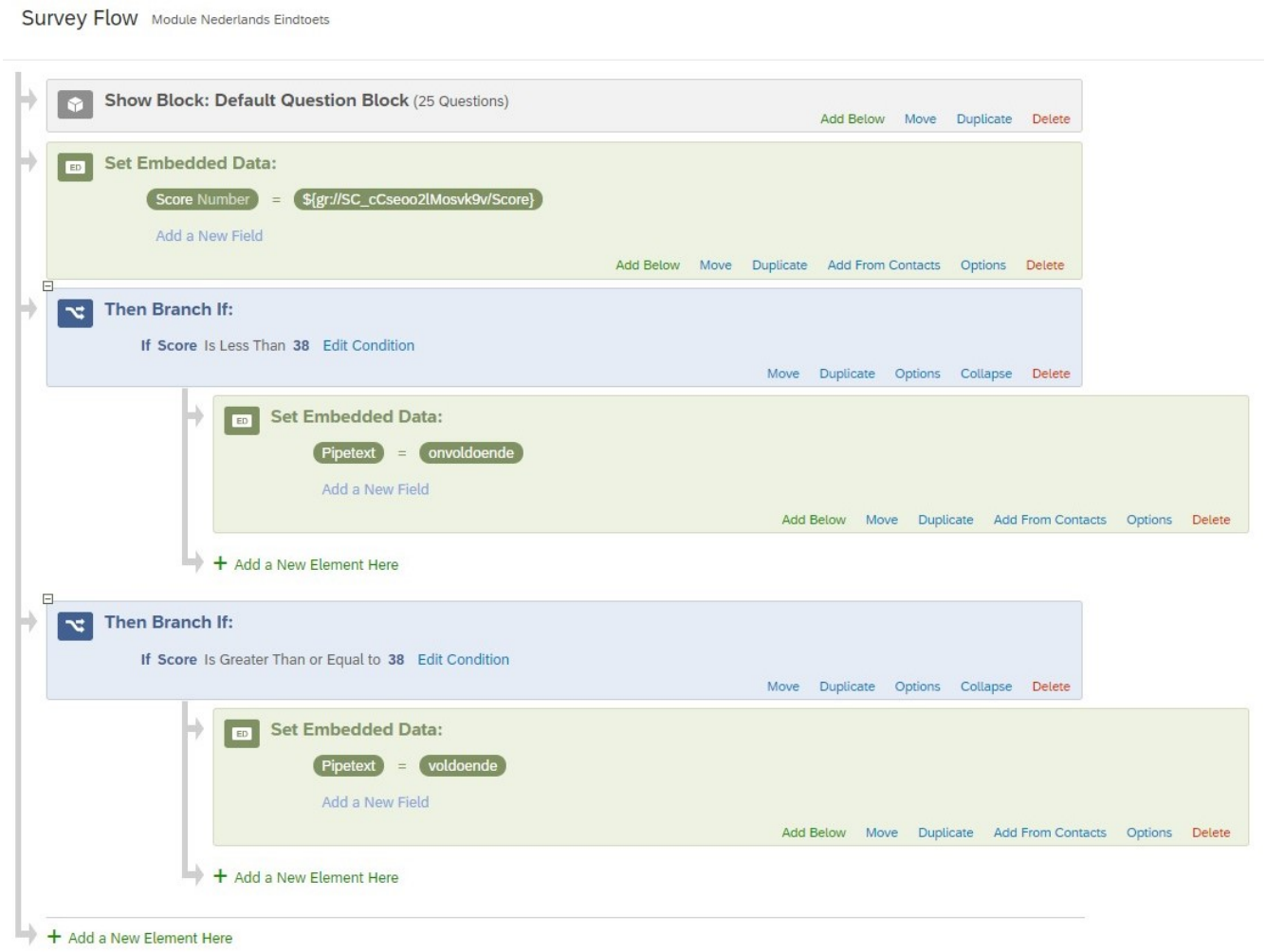
   Het is mogelijk de eindresultaten samen met de vragen te versturen naar de leerlingen, waarbij de mogelijkheid tot doorsturen naar andere klassen niet over het hoofd moet worden gezien. Het is het beste om alleen het aantal goede en foute antwoorden te rapporten zoals CBM aanbeveelt. Hierbij kan ervoor gekozen worden om ook de laagste categorie aan te geven zodat de deelnemer weet wat zijn valkuilen zijn.

10. **Stel prestatienormen vast**

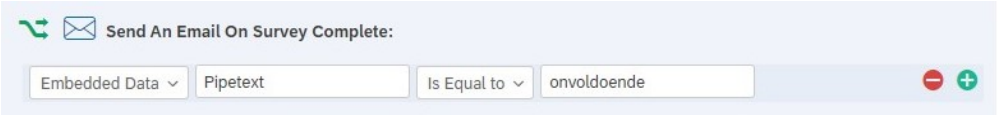
    Voor een certificaat is de helft van het totale aantal punten de minimum eis. Dit moet in de Survey Flow worden aangegeven, zodat er een goede beoordeling kan worden gemaakt door het systeem.

11. **Evalueer voorlopige items en update item pool**

    De item pool bestaat uit 34 vragen die alle leerdoelen en misconcepties bevatten. Dankzij de resultaten uit het onderzoek onder deskundigen zijn 5 vragen geëlimineerd, hierdoor blijven er 29 items over. De tekstuele terugkoppeling vanuit de deskundige over consensus tussen de antwoordopties en een betere stem is te gebruiken over de hele item pool. Hier is gedetailleerd



Figuur 4: Survey Flow in Qualtrics voor het automatiseren van e-mails



Figuur 5: Automatiseren e-mails met Triggers en Survey Flow in Qualtrics

naar gekeken. Dit resulteerde in de verandering van veertien items van de eerste item pool, onderverdeeld in drie groepen te zien in Tabel 6.

Opties, veranderd voor consensus	Inhoudelijk veranderd	Stem verbeterd
1, 11, 12, 13, 18, 19, 20, 28, 32	1, 14, 33	2, 5, 9

Tabel 6: Gewijzigde onderdelen van items uit de eerste item pool

Bij het weghalen van opties is er gelet op het weghalen van de zwakste afleiders. Optie D in vraag 1 is veranderd vanwege mogelijke verkeerde interpretatie. Vraag 33 is lastiger gemaakt door de meerkeuzevraag te veranderen in een open vraag. Ook is vraag 14 veranderd, dit om bias te voorkomen in de zin "huis kopen in New York" wat niet mogelijk is voor families van sommige deelnemers.

De uiteindelijke toetsing zal bestaan uit 18 vragen om ongeveer tweeënhalf minuut per vraag te hebben. Hiervoor worden nog negen vragen weggelaten.

Hiervoor zijn er verschillende methodes gehanteerd. In plaats van de vraag welke items weggelaten kunnen worden is er nagedacht over welke vragen belangrijk zijn en in de toetsing moeten komen. Hierbij is de eis gesteld om elke categorie van Bloom minimaal een keer voor te laten komen. Daarnaast is er gekeken welke vraag bij welke leerdoel en/of misconceptie hoorde, omdat alle leerdoelen en misconcepties in de toetsing aanwezig moeten zijn. Zo waren er vragen die als enige een bepaald leerdoel behandelden, waardoor deze vragen niet geëlimineerd konden worden. Na de meest belangrijke vragen geselecteerd te hebben was er nog ruimte voor meer vragen. Hierbij is er gekeken naar welke onderdelen in de leerdoelen en misconcepties het minst zijn behandeld en daarop geselecteerd; het was hierbij belangrijk dat elk onderdeel minimaal twee keer voorkwam. Dit resulteerde in een toetsing met 18 vragen.

5 Conclusie

In dit onderzoek is een antwoord op de volgende vraag nagestreefd: 'Hoe kan een prestatietest voor de evaluatie van Python vaardigheden voor de Module Nederlands worden ontwikkeld?' Voor een antwoord op deze vraag is een kwalitatief onderzoek in de vorm van literatuuronderzoek uitgevoerd over de integratie van informatica in schoolvakken, automatische beoordelingssystemen, prestatietests, richtlijnen voor het schrijven van items en becijfering van toetsen. Om de, met behulp van literatuuronderzoek, gecreëerde toetsing te kunnen beoordelen, is er uiteindelijk een kwantitatief veldonderzoek gedaan onder deskundigen.

Uit de onderzoeksresultaten kan worden geconcludeerd dat in het vakgebied van informatica weinig te vinden is over het creëren van een goede toetsing; ook was dit het geval bij richtlijnen van het schrijven van items en becijfering. Dit terwijl het vakgebied niet nieuw is. Het is hinderlijk om onderscheid te maken tussen leerlingen die het weten en degenen die het niet weten zonder een valide en betrouwbare toetsing. Ook is onderscheid maken tussen valkuilen en voordelen van de integratie zonder zo een toetsing lastig. De groei van Computational thinking integratie in vakken zorgt evenzeer voor een groei in het belang van goede prestatietests waarbij dit onderzoek van belang is. Door de afwezigheid van de richtlijnen in informatica is er gekeken bij andere vakgebieden.

De curriculum gebaseerde meting bleek het meest toepasselijke toetsing; door gebruik van deze vorm van toetsing is het mogelijk om de voortgang van de studenten te meten op een valide en

betrouwbare manier en een impact te hebben op de prestatie. Uit het veldonderzoek is gebleken dat de 21 universele stappen afkomstig uit de literatuur bruikbaar zijn in de creatie van de toetsing van Python vaardigheden voor de Module Nederlands. Deze stappen samen met de richtlijnen voor de ontwikkeling van items en de Taxonomie van Bloom voor becijfering zijn onmisbaar bij het ontwikkelen van de toetsing in dit vakgebied. Hierbij is er ook gebruikgemaakt van een automatisch beoordelingssysteem, wat voordelig bleek te zijn uit de literatuur.

6 Tekortkomingen

Ondanks het feit dat het aantal respondenten vanuit deskundigen in de methode volgens Dumas et al. (1995) genoeg is, zou een hogere aantal zorgen voor een betere validiteit van het onderzoek.[13] Dit kan voor de volgende keer worden verhoogd door bij speciale tijdschriften onder informatica docenten een berichtgeving te publiceren of door langs te gaan bij middelbare scholen. Het doel van dit onderzoek was het testen in de klaslokaal nadat de toetsing was opgesteld. Dit zou ervoor zorgen dat de toetsing en validiteit verbeterd kon worden. De toestanden die zijn ontstaan door Covid-19, zoals het sluiten van scholen, het veranderen van lesprogramma's en de onbereikbaarheid van leerlingen zorgde ervoor dat dit niet meer mogelijk was. Als het wel was gelukt zouden alle 21 stappen kunnen worden gebruikt. Na het testen zou een betere beeld verkregen kunnen worden of Qualtrics een goede testomgeving is voor de studenten. Ook zou het mogelijk zijn om het niveau van de toetsing te testen door middel van een moeilijkheidsindex gebruikt in het onderzoek van Wajeeha (2018).[50] Met de uitkomsten zou het mogelijk zijn geweest om de item pool aan te passen.

7 Aanbevelingen

Een aanbeveling op Hoofdstuk 6 zou zijn om in vervolgstudies een grotere groep deskundigen, met name meer docenten met kennis van informatica en ervaring in toetsing op middelbare scholen, te includeren. In het huidige onderzoek is er gebruik gemaakt van een kleine groep wat beperkingen met zich mee brengt. Daarnaast is in dit onderzoek vooral gekeken naar het aantal negatieve reacties van de deskundigen op de items. Door de grote item pool en weinig uitspringende gebrekkige items zijn de vragen er meteen uitgehaald. Bij een vervolgonderzoek met een fysieke afname zou er, in plaats van alleen op het einde, per vraag naar een onderbouwing gevraagd kunnen worden, wat kan leiden tot kleine wijzigingen, in plaats van het schrappen van hele items.

Een ander advies is om alle onderdelen van de universele stappen uit te voeren, met name de stappen die betrekking hebben op het testen in een populatie en de items te verbeteren met behulp van een moeilijkheidsindex.

Referenties

[1] K. M. Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102, 2005.

[2] R. Almeida, A. Gomes, E. Bigotte, and Teresa Pessoa. *iProg: Getting started with Programming: Pilot experiment in two elementary schools.* 2017.

[3] T. Barcelos, R. Munoz, R. Villarroel, E. Merino, and I. Silveira. Mathematics Learning through Computational Thinking Activities: A Systematic Literature Review. *JOURNAL OF UNIVERSAL COMPUTER SCIENCE*, 2018.

[4] V. Barr and C. Stephenson. *Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?* ACM Inroads, 2011.

[5] B. S. Bloom, M. D. Engelhart, E. J. Furst, W. H. Hill, and D. R. Krathwohl. *Taxonomy of Educational Objectives*. Handbook I: The Cognitive Domain, New York, 1956.

[6] Du Boulay and B. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1), 1986.

[7] Q. Burke. The markings of a new pencil: Introducing programming-as-writing in the middle school classroom. *The National Association for Media Literacy Education's Journal of Media Literacy Education*, 4:2, 2012.

[8] S. Clarke. Using curriculum-based measurement to improve achievement. *Principal*, 2009.

[9] J. W. Creswell. *Educational research: planning, conducting, and evaluating quantitative and qualitative research (4th edition).* Pearson, 2012.

[10] S. Deno. The nature and development of curriculum-based measurement. *Preventing School Failure: Alternative Education for Children and Youth*, 36:2, 1992.

[11] C. Douce, D. Livingstone, and J. Orwell. Automatic test-based assessment of programming. *Journal on Educational Resources in Computing*, 5(3), 2005.

[12] Steven M. Downing and Thomas M. Haladyna. Test item development: Validity evidence from quality assurance procedures, applied measurement in education. 10(1):61–82, 1997.

[13] J. Dumas, J. Sorce, and R. Virzi. *Expert Reviews: How Many Experts is Enough?*, volume 39. 1995.

[14] S. van Duyl. Ook op de basisschool krijgt programmeren voorrang, 2016. Algemeen Dagblad. Geraadpleegd op 21/07/2020 via https://www.ad.nl.

[15] S. N. Elliott. Achievement tests⋆. reference module in neuroscience and biobehavioral psychology. 2017.

[16] L. S. Fuchs, D. Fuchs, C. L Hamlett, N. B. Phillips, and J. Bentz. Classwide curriculum-based measurement: Helping general educators meet the challenge of student diversity. *Exceptional Children*, 60(6):518–537, 1994.

[17] U. Fuller, C. Riedesel, E. Thompson, C. G. Johnson, T. Ahoniemi, D. Cukierman, I. Hernán-Losada, J. Jackova, E. Lahtinen, Lewis, L. Tracy, and D.M Thompson. Developing a computer science-specific learning taxonomy. *ACM SIGCSE Bulletin*, 39(4), 2007.

[18] T. M. Haladyna, S. M. Downing, and M. C. Rodriguez. A review of multiple-choice item-writing guidelines for classroom assessment. *Applied Measurement in Education*, 15(3), 2002.

[19] H. E. Hawkes, E. F. Lindquist, C. R. Mann, and American Council on Education. *The construction and use of achievement examinations: A manual for secondary school teachers*. Houghton Mifflin Co, Boston, 1936.

[20] T. C. Hsu, Shao-Chen Chang, and Y. T. Hung. *How to learn and how to teach computational thinking: Suggestions based on a review of the literature*. Computers Education, 2018.

[21] Ihantola, Ahoniemi P., Karavirta T., V., and O. Seppälä. *Review of recent systems for automatic assessment of programming assignments*. Proceedings of the 10th Koli Calling International Conference on Computing Education Research - Koli Calling, 2010.

[22] C. Jenkins. Poem Generator: A comparative quantitative evaluation of a microworlds-based learning approach for teaching English. *International Journal of Education and Development using ICT*, 11(2), 2015. Geraadpleegd in oktober 2020 via http://ijedict.dec.uwi.edu/viewarticle.php?id=1972.

[23] Pediatric Nursing Certification Board (July. *Item Writing Manual*. 2019.

[24] U. Kale, M. Akcaoglu, T. Cullen, D. Goh, L. Devine, N. Calvert, and K. Grise. *Computational What? Relating Computational Thinking to Teaching*. TechTrends, 2018.

[25] L. J. Katz and G. T. Slomka. *Achievement Testing*. Handbook of Psychological Assessment, 149–182, 2000.

[26] A. Lajis, S. A. Baharudin, D. A. Kadir, N. M. Ralim, H. M. Nasir, and N. A. Aziz. A review of techniques in automatic programming assessment for practical skill test. *Journal of Telecommunication, Electronic and Computer Engineering*, 10:109–113, 2018.

[27] R. L. Linn. Issues of validity for criterion-referenced measures. *Applied Psychological Measurement*, 4(4):547–561, 1980.

[28] J. Lockwood and A. Mooney. Computational thinking in education: Where does it fit? a systematic literary review. *International Journal of Computer Science Education in Schools*, 2, 2017.

[29] L. Malmi, A. Korhonen, and R. Saikkonen. Experiences in automatic assessment on mass courses and issues for designing virtual courses. *ACM SIGCSE Bulletin*, 34(3), 2002.

[30] L. H. Martins-Pacheco, C. A. G. von Wangenheim, and N. C. Alves. Assessment of computational thinking in k-12 context: Educational practices, limits and possibilities-a systematic mapping study. *Proceedings of the 11th International Conference on Computer Supported Education (CSEDU*, 2019.

[31] J. Moreno-Leon and G. Robles. *Computer programming as an educational tool in the English classroom.* IEEE Global Engineering Education Conference (EDUCON), 2015.

[32] I. V. S Mullis and M. O Martin. *Item Writing Guidelines.* TIMSS PIRLS International Study Center, 2013.

[33] R. Nedeau-Cayo, D. Laughlin, L. Rus, and J. Hall. Assessment of item-writing flaws in multiple-choice questions. *Journal for Nurses in Professional Development*, 29(2), 2013.

[34] N. Nesiba, E. Pontelli, and T. Staley. *DISSECT: Exploring the relationship between computational thinking and English literature in K-12 curricula.* IEEE Frontiers in Education Conference (FIE), 2015.

[35] NOS. Google leert 'toekomstig' personeel programmeren op basisscholen, 2017. Geraadpleegd op 21/07/2020 via https://www.nos.nl.

[36] U.J. Obidiegwu. *Construction of standardized achievement test for adult learners in formal education: a necessity for adult learning.* Department of Adult Education Nnamdi Azikiwe University, Awka, 2005.

[37] J. A. Poteet. An outline for standardized achievement testing. *Diagnostique*, 16(1), 1990.

[38] Qualtrics, 2020. Geraadpleegd in juni 2020 via https://www.qualtrics.com/.

[39] T. Regan. Item-writing guidelines for physics. *The Physics Teacher*, 53(8):485–487, 2015.

[40] C. R. Reynolds. *Fundamentals of Measurement and Assessment in Psychology.* Comprehensive Clinical Psychology, 33–55, 1998.

[41] D. Schneider and N. Mather. *Achievement Testing.* The Encyclopedia of Clinical Psychology, 2015.

[42] J. Sheard, Simon, A. Carbone, D. D'Souza, and M. Hamilton. *Assessment of programming: pedagogical foundations of exams.* Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '13., 2013.

[43] J. Sorva. Visual program simulation in introductory programming education. 2012.

[44] P. M. Stecker, L. S. Fuchs, and D. Fuchs. Using curriculum-based measurement to improve student achievement: Review of research. *Psychology in the Schools*, 42(8), 2005.

[45] A. Swidan, F. Hermans, and M. Smit. *Programming Misconceptions for School Students.* 2018.

[46] TIMSS and PIRLS International Study Center. *About TIMSS PIRLS International Study Center.* 2019. Bekeken via: https://timssandpirls.bc.edu/about.html.

[47] G. Tindal. Curriculum-based measurement: A brief history of nearly everything from the 1970s to the present. 2013.

[48] VHTO. Missie en visie. Geraadpleegd op 21/07/2020 via https://www.vhto.nl.

[49] VHTO. Over computer science certificate. Bekeken via https://www.vhto.nl op 21/07/2020.

[50] D. Wajeeha, S. Alam, U. Hassan, T. Zafar, R. Butt, S. Ansari, and M. Rizvi. *Difficulty Index, Discrimination Index and Distractor Efficiency in Multiple Choice Questions*. annals of pims, 2018.

[51] J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3), 2006.

[52] U. Wolz, M. Stone, K. Pearson, S. M. Pulimood, and M. Switzer. Computational thinking and expository writing in the middle school. *ACM Transactions on Computing Education*, 11(2), 2011.

Appendices

A Eerste versie item pool

item 1 onderdeel 1.2	Geef bij de onderstaande stellingen aan of ze juist of onjuist zijn. Een stukje commentaar in een programmeertaal wordt geschreven om A. een stukje code uit te voeren. B. uitleg bij een stukje code te geven. C. boven het programma te schrijven wat het doet. D. om bij elke regel code uitleg te geven. E. een regel code 'eventjes' uit te zetten.
bedoeling	Kijken of de leerlingen doorhebben wanneer een stukje commentaar nodig is in de code
sleutel	onjuist, juist, juist, onjuist, juist
Bloom tax.	application
redenering foute antwoorden	- A. zwakste afleider, toch goed om te testen om te weten of ze wel doorhebben dat commentaar niet wordt uitgevoerd. - B. [sleutel] - C. [sleutel] - D. dit is een beginnersfout wat vaak wordt gedaan, teveel uitleg is niet nodig - E. [sleutel]

**item 2
onderdeel 1.1
& 7**
	Wat print deze code?
	
	#Dit is test code om iets te printen
	print('test 1, 2, 3')
	#print('yey, het werkt')
	
	A. test 1, 2, 3
	B. Dit is test code om iets te printen
	test 1, 2, 3
	yey, het werkt
	C. test 1, 2, 3
	yey, het werkt
	D. Dit is test code om iets te printen
	print('test 1, 2, 3')
	print('yey, het werkt')

bedoeling	Kijken of leerling het verschil ziet tussen code en commentaar, en ziet dat commentaar niet wordt geprint
sleutel	A
Bloom tax.	Analysis
redenering foute antwoorden	A. [sleutel] B. Leerling ziet niet in dat commentaar niet moet worden geprint, maar maakt wel onderscheid dat woorden in een print moeten worden geprint en niet het woord print zelf. C. Leerling ziet in dat de eerste regel commentaar is, maar print wel beide prints. D. Leerling ziet niet in dat commentaar noch letterlijk 'print' niet worden geprint

item 3 onderdeel 1 & 7	Combineer de volgende 3 regel codes met de 3 mogelijke uitvoeren.
	codes:
	A. print(‘test’)
	#print(‘test’)
	print(‘test’)
	B. #test
	print(‘test’)
	C. print(#‘test’)
	#test
	uitvoeren:
	a. test
	b. deze code geeft een fout
	c. test
	test

bedoeling	Met deze vraag zien we of de leerlingen fouten herkent, de uitvoer met code kan combineren en wee dat code na de “#” niet wordt uitgevoerd. Dit is een soortgelijke opgaven uit de werkbladen.
sleutel	Ac, Ba, Cb
Bloom tax.	Analysis
redenering foute antwoorden	- Aa → leest op een of andere manier alleen 1 test, misschien met de gedachte dat de regels na commentaar ook niet worden gelezen. - Ab → ziet een fout die er niet is. - Bb → ziet een fout die er niet is. - Bc → print ook het commentaar. - Ca → ziet de fout in regel 1 niet, leest over het symbool heen of denkt dat dat gewoon kan, of denkt dat dat niet wordt geprint maar de tweede regel wel. - Cc → ziet noch de fout noch het commentaar, print ze uit ondanks de syntax.

item 4 onderdeel 4.1 & 6.4	Wat print de code uit? naam = input print(naam) ---------- input: Stijn A. Stijn B. FOUT C. naam
bedoeling	fout bij regel 1 input zonder ronde haakjes herkennen
sleutel	B
Bloom tax.	Synthesis
redenering foute antwoorden	A. fout niet herkennen en denken dat het doet zoals wanneer er “input()” zou staan B. [sleutel] C. denken dat naam letterlijk wordt geprint

item 5 onderdeel 4.2	Wat doet de volgende code?

& 4.3 & 4.4	naam = input() print(‘Hallo ik ben’, naam) A. wacht op een input B. print: Hallo ik ben input() C. print: Hallo ik ben naam
bedoeling	begrijpen dat het programma niet verder gaat zolang gebruiker geen input geeft. Dit is een misconceptie zie misconcepties gebruikersinput.
sleutel	A
Bloom tax.	Analysis

redenering foute antwoorden	- Blank
	- “Hallo ik ben <een naam>”
	  - niet weten dat er moet worden gewacht op input
	- “Hallo ik ben input()”
	  - niet weten dat er moet worden gewacht op input en
	  - niet weten dat naam input wordt en niet letterlijk “input()”

item 6 onderdeel 4.1+4.2+4.3	In de volgende code wordt er “Hallo naam” geprint. Welk woord in de code moet worden veranderd om per gebruiker de naam te kunnen printen? 1. naam = ‘naam’ 2. print(‘Hallo’, naam) A. naam uit regel 1. B. ‘naam’ uit regel 1. C. naam uit regel 2.
bedoeling	Begrijpen dat naam uit regel 2 van de variabele naam uit regel 1 komt en dat niet gelijk is aan naam. Om per gebruiker de naam te kunnen printen moet ‘naam’ uit regel 1 ingeruild worden met input().
sleutel	B
Bloom tax.	Synthesis
redenering foute antwoorden	A. Hierbij zou een leerling de variabelen dus verwarren. Of hij/zij denkt dat de variabele naam dat is wat letterlijk wordt geprint. Of hij verwart de variabele en inhoud. B. [sleutel] C. In principe klopt dit, maar het is niet netjes noch hebben ze het zo geleerd.

item 7 onderdeel 2.1 & 2.2 & 2.3 & 4.1 & 4.2 & 4.3 & 7.3 & 7.4	Je krijgt een een aantal codes met een uitvoer. Geef bij elke optie aan of de uitvoer juist of onjuist is. A. Code: print('Suiker', 'of', 'melk?') in_de_koffie = 'melk' input() print('Okee', 'ik', 'doe', 'er', in_de_koffie, 'in')

	--- Uitvoer: Suiker of melk? input: suiker Uitvoer: Okee ik doe er suiker in B. Code: print('kat', 'of', 'hond?') mijn_huisdier = input print('Mijn', mijn_huisdier, 'is', 'weggelopen') --- Uitvoer: kat of hond? input: hond Uitvoer: Mijn hond is weggelopen C. Code: print('Hoe laat is het?') tijd = input() print('Het is nu', tijd, '.') --- Uitvoer: Hoe laat is het? input: twaalf uur Uitvoer: Het is nu twaalf uur. D. Code: print('Hoe gaat het?') humeur = input() print('Goed', 'om', 'te', 'horen', 'dat', 'je', 'je, 'humeur', 'voelt!') --- Uitvoer: Hoe gaat het? Input: goed Uitvoer: Goed om te horen dat je je goed voelt
bedoeling	Onderscheidt maken tussen leerlingen die echt de syntax begrijpen en goed hebben geoefend. De kleine fouten kunnen inzien, dit zorgt ervoor dat fout detecteren bij een error makkelijk wordt.
sleutel	A. onjuist B. onjuist C. juist D. onjuist
Bloom tax.	Synthesis
redenering foute antwoorden	A. input heeft geen werking hier, behalve dat het iets ophaalt. Het wordt niet een variabele opgeslagen. Leerling kan achterliggende gedachten hebben dat alles wat wordt ingenomen als input ook werkelijk wordt geprint.

	B. haakjes bij de input niet meegenomen. Dit is een zwakke afleider maar belangrijk om te testen. C. [juist], gekeken of leerling inziet dat aanhalingstekens niet per se bij elke woord hoeven. D. aanhalingstekens bij variabele 'humeur' waardoor 'Goed om te horen dat je je humeur voelt' wordt geprint. Bij artikel [31] had bijna 40% van de Python studenten deze vraag fout. Lijkt een zwakke afleider maar is het dus niet.

item 8 onderdeel 7.2	Welke van deze printopdrachten print **Welkom** uit volgens de modules? A. print('Welkom") B. print("Welkom") C. print('Welkom') D. print(Welkom) E. print("Welkom')
bedoeling	Begrip
sleutel	C
Bloom tax.	Synthesis
redenering foute antwoorden	A. mix van aanhalingstekens, leerling weet wel dat er aanhalingsteken moet, maar niet welke. Leerlingen met dit als antwoord hebben het zeer slecht begrepen B. Dit is in principe een werkende code, leerlingen met voorkennis zullen dit waarschijnlijk weten. Maar de vraag zegt expliciet: "volgens de modules". Dit is niet de manier waarop ze het hebben geleerd. C. [sleutel] D. Geen aanhalingstekens, weinig kennis. Leerling weet überhaupt niet dat er iets moet komen. E. Om B te versterken is het tegenovergestelde ook nodig, zodat het de leerlingen die het antwoord proberen te raden misleidt.

item 9 onderdeel 7.1 & 7.2 & 7.3 & 2.1 & 2.2	Welke drie opties van deze printopdrachten print **Hello World 1.0** uit? A. print Hello World 1.0 B. print('Hello World 1.0') C. print('Hello', 'World', 1.0) D. print 'Hello World 1.0' E. print('Hello', 'World', '1.0') F. versie = '1.0' print('Hello World', versie)
bedoeling	Alle verschillende mogelijkheden met print weten.
sleutel	B, E, F
Bloom tax.	Knowledge
redenering foute antwoorden	A. geen haakjes, weet leerling dat het zonder haakjes niet wordt geprint. B. [sleutel] C. 1.0 moet aanhalingstekens, kleine fout maar belangrijk om te zien, anders krijg je te maken met errors die moeilijk zijn te zien. D. haakjes afwezig, versterkt A, anders was A overduidelijk niet een goede afleider, verschil is

	alleen in de aanhalingstekens. E. [sleutel] F. [sleutel]

item 10 onderdeel 2.1	Wat moet er op de puntjes in regel 1 van de code komen voor een correcte uitvoer volgens de modules? dier = …. print('In de dierentuin bewonderden we een', dier) A. giraffe B. 'giraffe' C. "giraffe"
bedoeling	variabele goed kunnen definiëren
sleutel	B
Bloom tax.	Synthesis
redenering foute antwoorden	A. geen aanhalingstekens, makkelijk te verwarren/vergeten B. [sleutel] C. normaliter kloppend, maar niet aangegeven in de modules

item 11 onderdeel 2.1, 2.3, 2.4, 2.5, 2.8,	Wat print de volgende code? eten = 'pizza' eten = 'macaroni' print('Ik eet vandaag', eten) A. Ik eet vandaag macaroni B. Ik eet vandaag pizza C. Ik eet vandaag pizza & macaroni D. Ik eet vandaag pizza Ik eet vandaag macaroni E. Deze code geeft een fout
bedoeling	Testen of leerling begrijpt dat de vorige inhoud wordt vervangen wanneer variabele opnieuw wordt gedefinieerd.
sleutel	A
Bloom tax.	Analysis
redenering foute antwoorden	A. [sleutel] B. Leerling denkt dat de tweede definitie niets doet. C. Leerling denk dat variabele meerdere waardes vasthoudt. misconceptie 1 van variabelen. D. Leerling denkt dat bij elke definitie gelijk naar de print wordt gegaan, of omdat er twee keer wordt gedefinieerd dat er ook twee keer een print moet komen. E. Leerling ziet een fout die er niet is.

item 12 onderdeel 2.3, 2.4, 2.5, 2.6, 2.8, 7.2	Wat print de volgende code? achternaam = 'Hermans' print('Hallo', 'mevrouw', Hermans) achternaam = ‘van den Berg’ A. Hallo mevrouw Hermans B. Hallo mevrouw van den Berg C. Hallo mevrouw achternaam D. Hallo mevrouw Hermans Hallo mevrouw van den Berg E. FOUT
bedoeling	De valkuil misconceptie testen en de herkenning van de fout in print
sleutel	E
Bloom tax.	Synthesis
redenering foute antwoorden	A. Ziet of niet in dat **Hermans** aanhalingstekens moet hebben, of denkt dat Hermans een variabele is. B. Als leerling denkt dat de code van beneden naar boven loopt en dus **van den Berg** als eerst wordt gelezen en ook geprint. Of omdat **van den Berg** als laatst staat en dus de variabele naar dat wordt omgezet en zo wordt geprint. C. Denkt dat Hermans een variabele is en achternaam de waarde. D. Denkt dat beide variabelen worden geprint, de code gaat heen en weer, of variabele heeft meerdere waardes (misconceptie) E. [sleutel]

item 13 onderdeel 2.2, 2.3, 2.4, 2.5, 2.8, 7.2, 7.4	Wat print de volgende code? dier = 'dolfijn' print('In', 'de', ‘dierentuin’, ‘zagen’, ‘we’, ‘een’, dier) dier = ‘olifant’ print(‘Ook’, ‘was’, ‘er’, ‘een’, ‘baby’, dier) A. In de dierentuin zagen we een dier Ook was er een baby dier B. In de dierentuin zagen we een dolfijn Ook was er een baby olifant C. In de dierentuin zagen we een olifant Ook was er een baby olifant D. In de dierentuin zagen we een dolfijn Ook was er een baby dolfijn E. In de dierentuin zagen we een dolfijn Ook was er een baby olifant F. FOUT
bedoeling	Misconceptie 1, opnieuw toewijzen van een waarde aan een variabele
sleutel	E

Bloom tax.	Analysis
redenering foute antwoorden	A. Letterlijk printen van de variabele namen, afwezigheid van de aanhalingstekens over het hoofd gezien of niet wetend dat de waarde moet worden geprint. B. Qua waarde print kloppend, maar prints kunnen niet naast elkaar, bij deze keuze weet de leerling dit blijkbaar niet. C. Misconceptie, leerling denkt bij deze keuze dat de laats gegeven waarde overal in komt. D. Misconceptie, leerling denkt dat variabele alleen een waarde kan hebben. E. [sleutel] F. Fout zien die er niet is.

item 14 onderdeel 2.3 , 7.2	Wat print de volgende code? a = ‘een’ print(‘We hebben’, ‘a’, ‘huis gekocht in New York’) A. We hebben a huis gekocht in New York B. We hebben een huis gekocht in New York C. FOUT
bedoeling	de variabele naam als string printen in plaats van zijn waarde
sleutel	A
Bloom tax.	Analysis
redenering foute antwoorden	A. [sleutel] B. Bij leerling die voor deze optie kiezen kan het volgende afspelen: a. Hij/zij weet waarschijnlijk niet dat een woord/zin met aanhalingstekens letterlijk wordt geprint b. Leerling ziet de aanhalingstekens over het hoofd C. Leerling ziet een fout die er niet is

item 15 onderdeel 2.1, 2.2, 2.3, 2.7, 7.2	Wat print de volgende code? kort = ‘World’ lang = ‘a’ print(kort, lang) A. World a B. a a C. kort lang D. a World
bedoeling	De natuurlijke taal semantiek van variabele namen misconceptie testen
sleutel	A
Bloom tax.	Synthesis

redenering foute antwoorden	A. [sleutel] B. Variabele lang is later gedefinieerd waardoor alles ‘a’ wordt. C. Letterlijk printen van de variabele namen, afwezigheid van de aanhalingstekens over het hoofd gezien of niet wetend dat de waarde moet worden geprint. D. Misconceptie, letter a is kort waardoor erbij kort ‘a’ wordt geprint en bij lang ‘World’.

item 16 onderdeel 2.2, 2.3, 2.4, 2.5, 2.8, 7.2	Wat print de volgende code uit? a= ‘vis’ b= ‘hond’ a= ‘kat’ print(a, b, b) A. vis hond hond B. vis hond kat C. a b b D. kat hond hond
bedoeling	opnieuw waarde toewijzen aan variabele *a* met drie geprinte variabelen [31]
sleutel	D
Bloom tax.	Synthesis
redenering foute antwoorden	A. Eerste waarde van *a* is niet veranderd. Student denkt dat alleen de eerste declaratie blijft en de tweede declaratie wordt genegeerd. [31] B. Volgens [31] meest gekozen optie. Student denkt dat alle variabelen geprint moeten worden. C. Letterlijk printen van de variabele namen, afwezigheid van de aanhalingstekens over het hoofd gezien of niet wetend dat de waarde moet worden geprint. D. [sleutel]

item 17 onderdeel 2.1, 2.2, 2.3, 2.4, 2.5, 2.8, 7.2	Wat print de volgende code uit? a= ‘vis’ b= ‘hond’ b= ‘kat’ print(a, b, b) A. vis hond kat B. a b b C. vis kat kat D. vis hond hond
bedoeling	opnieuw waarde toewijzen aan variabele *b* met drie geprinte variabelen [31]
sleutel	C
Bloom tax.	Analysis
redenering	A. Student denkt dat alle variabelen geprint moeten worden of denkt dat b meerdere waardes

foute antwoorden	bevat. B. Letterlijk printen van de variabele namen, afwezigheid van de aanhalingstekens over het hoofd gezien of niet wetend dat de waarde moet worden geprint. C. [sleutel] D. Eerste waarde van *b* is niet veranderd. Student denkt dat alleen de eerste declaratie blijft en de tweede declaratie wordt genegeerd. [31]

item 18 onderdeel 3.1, 3.2, 3.3	Welke declaratie van de lijst dieren klopt? A. dieren = (konijn, dolfijn, leeuw) B. dieren = (‘konijn’, ‘dolfijn’, ‘leeuw’) C. dieren = [konijn, dolfijn, leeuw] D. dieren = [‘konijn’, ‘dolfijn’, ‘leeuw’] E. dieren = konijn, dolfijn, leeuw F. dieren = ‘konijn’, ‘dolfijn’, ‘leeuw’
bedoeling	Testen van herkenning van een goede declaratie
sleutel	D
Bloom tax.	Synthesis
redenering foute antwoorden	A. verkeerde haakjes en geen aanhalingstekens, bij deze optie heeft de leerling geen gedetailleerde kennis B. aanhalingsteken kennis klopt, maar haakjes niet. C. zelfde formaat als a met andere haakjes, bij deze optie weet leerling de goede haakjes maar vergeet de aanhalingstekens D. [sleutel] E. Leerling doet alsof het een string variabele is en denkt dat het zonder haakjes wordt gedeclareerd. F. leerling heeft alleen kennis van aanhalingstekens Herhaling van dezelfde soort woorden met elke keer een kleine verandering verminderd de kans op een goede gok en haalt de leerlingen die het echt weten eruit.

item 19 onderdeel 3.5, 3.6	We hebben een random lijst genaamd **kleuren** met 3 kleurnamen erin. Welke print klopt? A. print(‘Mijn kamer is’, kleur, ‘van kleur’) B. print(‘Mijn kamer is’, kleur[1], ‘van kleur’) C. print(‘Mijn kamer is’, kleur(1), ‘van kleur’) D. print(‘Mijn kamer is’, ‘kleur’, ‘van kleur’) E. print(‘Mijn kamer is’, ‘kleur[1]’, ‘van kleur’) F. print(‘Mijn kamer is’, ‘kleur(1)’, ‘van kleur’)
bedoeling	Herkenning van gebruik van een lijst.
sleutel	B
Bloom tax.	Synthesis
redenering	A. Leerling verward lijst met variabelen of vergeet gebruik van haakjes.

foute antwoorden	B. [sleutel] C. Leerling weet dat er haakjes moet worden gebruikt maar gebruikt de verkeerde. D. Leerling weet dat kleur apart moet komen, maar print als normale tekst. E. Leerling weet gebruik syntax van lijsten, maar zet onnodige haakjes, dus weet niet dat haakjes ervoor zorgen dat het letterlijk wordt geprint. F. Om afleider C en E te versterken.

item 20 onderdeel 3.7, 6.8	We hebben een random lijst genaamd **kleuren** met 3 kleurnamen erin. Geef bij elke print aan of het een kloppende uitvoer zal geven? A. print('Mijn kamer is', kleur[1], 'van kleur') B. print('Mijn kamer is', kleur[3], 'van kleur') C. print('Mijn kamer is', kleur[-1], 'van kleur') D. print('Mijn kamer is', kleur[2], 'van kleur') E. print('Mijn kamer is', kleur[0], 'van kleur')
bedoeling	Kennis van de randvoorwaarden bij printen van lijsten getest.
sleutel	juist, onjuist, onjuist, juist, juist
Bloom tax.	Synthesis
redenering foute antwoorden	A. [sleutel] B. aantal woorden -1 is maximum, het is iets wat makkelijk vergeten wordt, maar uiterst belangrijk is. C.

item 21 onderdeel 3.5, 3.6, 3.7, 6.8	Match de print met de uitvoer fruit = ['appel', 'peer', 'banaan'] A. print(fruit[0]) B. print(fruit[1]) C. print[fruit[2]) D. print(fruit[3]) 1. banaan 2. appel 3. peer 4. FOUT
bedoeling	Testen of de leerling weet welke getal welke woord voorsteld.
sleutel	A2 B3 C1 D4
Bloom tax.	Synthesis
redenering foute antwoorden	Er is veel mogelijk, maar als de syntax wel is begrepen is de grootst mogelijke foute antwoord het volgende: A4, B2, C3, D1. Denken dat lijst met 1 begint en zo doorgaat.

item 22 onderdeel 3.7	Wat is de juiste invoer die op "..." in de code moet komen om dezelfde uitvoer te genereren? code: talen = ['Python', 'JavaScript', 'HTML', 'C++', 'C', 'Assembly'] print('Dit programma heb ik gemaakt met', ……) print('Daarnaast kan ik programmeren in', …...) print('en binnenkort ga ik leren programmeren in', ……) Uitvoer: Dit programma heb ik gemaakt met Python Daarnaast kan ik programmeren in Javascript en binnenkort ga ik leren programmeren in Assembly Mogelijke antwoorden: A. talen[1], talen[2], talen[5] B. talen[0], talen[1], talen[5] C. talen[1], talen[2], talen[6] D. talen[0], talen[1], talen[6]
bedoeling	Kennis testen van correcte verwijzingen naar woorden van een lijst.
sleutel	B
Bloom tax.	Synthesis
redenering foute antwoorden	A. Weet niet dat de lijst van 0 start, maar geeft de laaste lijst item wel de goede getal. B. [sleutel] C. Weet niet dat de lijst van 0 start, maar weet wel dat de getallen achterelkaar komen. D. Deze afleider is om zowel B en C te versterken. Leerling start goed maar eindigt verkeerd.

item 23 onderdeel 4.1, 6.3, 6.4, 6.7	Welke foutmelding krijg je bij de volgende code? achternaam = input(------ input = Dane A. NameError B. IndentationError C. SyntaxError
bedoeling	Fout identificeren en ook kunnen benoemen.
sleutel	C
Bloom tax.	Evaluation
redenering foute antwoorden	A. Een error die vaak voorkomt, maar hier niet het geval, leerlingen met deze keuze zien de fout niet, of weten niet welke foutmelding bij welke fout hoort. B. Een error die bij if-else komt (ook bij vele andere plaatsten, maar leerling heeft dit nog niet geleerd). C. [sleutel]

item 24 onderdeel 5.3, 6.3, 6.6, 6.7	Welke foutmelding krijg je bij de volgende code? if in_de_thee == 'suiker' print(‘Thee met suiker’) A. NameError B. IndentationError C. SyntaxError
bedoeling	Fout identificeren en ook kunnen benoemen.
sleutel	C
Bloom tax.	Evaluation
redenering foute antwoorden	A. Een error die vaak voorkomt, maar hier niet het geval, leerlingen met deze keuze zien de fout niet, of weten niet welke foutmelding bij welke fout hoort. B. Een vaak verkregen error bij if-else, als leerling denkt dat alle fouten hieraan toe te schrijven zijn heeft hij een foutbeeld gecreeerd. C. [sleutel]

item 25 onderdeel 6.3, 6.7	Welke foutmelding krijg je bij de volgende code? print(Welkom) A. NameError: name 'Welkom' is not defined” B. IndentationError: expected an indented block C. SyntaxError: EOL while scanning string
bedoeling	fout in de aanhalingstekens zien en weten dat dat een variabelen moet zijn als het geen aanhalingstekens heeft. Hierdoor → not defined error
sleutel	A
Bloom tax.	Evaluation
redenering foute antwoorden	A. [sleutel] B. Error gegeven wanneer er moet worden ingesprongen bij bijvoorbeeld een if. Hier totaal geen sprake. Als werkblad 5a laatste vraag niet is gezien als een hele code maar echt aparte stukjes kan de leerling denken dat het deze error is die hierbij hoort. C. Verkregen wanneer er iets deels fout is in de code. In dit geval zijn de aanhalingstekens afwezig maar het is pas een SyntaxError als eentje mist, nu wordt het door de code gezien als een variabele die niet is aangemaakt.

item 26 onderdeel 5.4, 6.3, 6.7	Welke foutmelding krijg je bij de volgende code? if in_de_thee == 'suiker':

	print(‘Thee met suiker’) else: print(in_de_thee) A. NameError B. IndentationError C. SyntaxError
bedoeling	Fout identificeren en ook kunnen benoemen.
sleutel	B
Bloom tax.	Evaluation
redenering foute antwoorden	A. Een error die vaak voorkomt, maar hier niet het geval, leerlingen met deze keuze zien de fout niet, of weten niet welke foutmelding bij welke fout hoort. B. [sleutel] C. Een error die vaak voorkomt, maar hier niet het geval, leerlingen met deze keuze zien de fout niet, of weten niet welke foutmelding bij welke fout hoort.

item 27 onderdeel 4.1, 6.3, 6.5, 6.7	Welke foutmelding krijg je bij de volgende code? achternaam = input ------ input = Dane A. Een rare output B. IndentationError C. SyntaxError
bedoeling	Fout identificeren en ook kunnen benoemen.
sleutel	A
Bloom tax.	Evaluation
redenering foute antwoorden	A. [sleutel] B. Een error die bij if-else komt (ook bij vele andere plaatsten, maar leerling heeft dit nog niet geleerd), dus hier helemaal niet het geval. C. Een error die vaak voorkomt, maar hier niet het geval, leerlingen met deze keuze zien de fout niet, of weten niet welke foutmelding bij welke fout hoort.

item 28 onderdeel 2.1, 4.1, 6.1, 6.2, 6.3, 6.4, 6.5, 7.1, 7.2	Welke van de volgende stukje codes geven een SyntaxError? A. print(Waarden) B. print(‘Waarden) C. print(‘Waarden’) D. print(‘Waarden’ E. naam = input F. naam = ‘input’

	G. naam = ‘input H. naam = input(
bedoeling	Fout identificeren en ook kunnen benoemen.
sleutel	B, D, G, H
Bloom tax.	Synthesis
redenering foute antwoorden	A. Fout matched niet met foutmelding, hier geval van NameError. B. [sleutel] C. Correcte uitvoer, leerling ziet een fout die er niet is. D. [sleutel] E. Fout matched niet met foutmelding, hier geval van een rare output. F. Correcte uitvoer, leerling ziet een fout die er niet is. G. [sleutel] H. [sleutel]

item 29 onderdeel 5.1	Wat is de uitvoer van de volgende code?
	
	print('suiker', 'of', 'melk?')
	in_de_thee = input()
	if in_de_thee == 'melk':
	  print('gieten')
	else:
	  print('schudden')
	--- 	
	Input: melk
	
	A. suiker of melk gieten
	B. suiker of melk schudden
	C. suiker of melk
	D. FOUT

bedoeling	Volgorde van uitvoer in if-else herkennen en weten welke statement wordt genomen.
sleutel	A
Bloom tax.	Analysis
redenering foute antwoorden	A. [sleutel] B. Neemt de else-statement, herkend niet dat de if moet worden genomen. C. Neemt geen enkele statement, herkend niet dat de if moet worden genomen. D. Ziet een fout die er niet is.

item 30 onderdeel 5.5, 5.6	Wat is de uitvoer van de volgende code? print(‘opdracht 1’) taal = ‘Nederlands’

	if taal == 'Duits':
	  print('Gutenabend!')
	print('Laten we snel beginnen')
	
	A. opdracht 1
	  Gutenabend
	B. opdracht 1
	  Gutenabend
	  Laten we snel beginnen
	C. opdracht 1
	  Laten we snel beginnen
	D. opdracht 1
	  opdracht 1
	  Laten we snel beginnen
	E. opdracht 1
	  opdracht 1

bedoeling	De volgorde en manier van uitvoer kunnen lezen bij een niet genomen if-statement
sleutel	C
Bloom tax.	Analysis
redenering foute antwoorden	A. leerling neemt de if-statement terwijl dat niet hoort en stopt ook nadat dat is gelezen. Of hij neemt aan dat de andere regel een else is, of leest het over het hoofd of denkt echt dat het na de if stopt. B. leerling neemt als enige fout de if-statement. Of hij denkt dat de if altijd wordt genomen of ziet niet in dat taal Nederlands is in plaats van Duits of hij denkt dat de print niks te maken heeft met de if. C. [sleutel] D. misconceptie 5.6, door de false if statement gaan we naar het begin en alles opnieuw doen maar de if overslaan. E. misconceptie 5.6 weer maar dan ook nog denken dat door de false statement niets erna wordt gelezen.

item 31 onderdeel 4.4, 5.1, 5.5, 5.6	Hebben de volgende twee codes dezelfde uitvoer?
	
	code 1:
	print('suiker', 'of', 'melk?')
	in_de_thee = input()
	if in_de_thee == 'melk':
	  print('gieten')
	else:
	  print('schudden')
	--- 	
	Input: koffie
	--- 	
	code 2:
	print('suiker', 'of', 'melk?')
	in_de_thee = input()

	if in_de_thee == 'melk':
	print('gieten')
	else:
	print('schudden')
	--- 	
	Input: suiker
	---------- 	
	A. Ja
	B. Nee

bedoeling	Begrijpen dat er geen controle is bij de else, dus al heb je een invoer van iets anders dat het wordt geaccepteerd
sleutel	A
Bloom tax.	Comprehension
redenering foute antwoorden	A. [sleutel] B. de trick van de else niet begrijpen of denken dat ze alleen suiker of melk mogen intypen.

item 32 onderdeel 5.2, 5.3, 5.4, 6.6, 6.7	Zijn de volgende codes juist of onjuist?
	
	1. print('Nederlands of Engels')
	taal = 'Nederlands'
	if taal = 'Engels'
	print('Hello', 'good morning!')
	else
	print('Hallo', 'goedemorgen!')
	2. print('reptiel of zoogdier?')
	diersoort = 'reptiel'
	if diersoort == 'reptiel':
	print('legt', 'een', 'ei')
	else:
	print('geeft', 'melk')
	3. print('hond of kat')
	dier = 'hond'
	if dier = 'hond':
	print('waf')
	else:
	print('miauw')
	4. slapen = 'ja'
	if slapen == 'ja':
	print('Slaap lekker')
	else:
	print('laten we een film kijken')
	5. jarig = 'vandaag'
	if jarig == 'vandaag':
	print('Gefeliciteerd!')
	6. print('koe of kikker')
	dier = 'kikker'

	if dier == ‘koe’
	  print('boe')
	7. print('Duits', 'of', 'Nederlands')
	taal = ‘Duits’
	if taal == 'Duits':
	  print('Gutenabend!')
	else:
	  print('Goedeavond!')

bedoeling	De kleine fouten die gemaakt worden in de if-statements herkennen
sleutel	1/2/3/6 onjuist, 4/5/7 juist
Bloom tax.	Synthesis
redenering foute antwoorden	1. Geen dubbel ‘=’ gebruikt bij vergelijking en de dubbele punten na de else en de if weggehaald. Kleine foutjes die snel over het hoofd worden gezien. Beiden worden apart versterkt door andere afleiders. 2. Er wordt niet ingesprongen, design fout 3. Alles goed behalve de afwezigheid van de vergelijkingsteken 6. Dubbele punten van de if afwezig

item 33 onderdeel 5.2	Wat moet er op de puntjes? naam = input() if naam …. ‘Elsa’: print(‘Hallo Elsa!’) ------- input = Elsa A. = B. == C. niks D. ()
bedoeling	De syntax van if-statement begrijpen, dus weten dat daar een vergelijking wordt uitgevoerd en dankzij wat dat gebeurt.
sleutel	B
Bloom tax.	Knowledge
redenering foute antwoorden	A. Syntax bijna correct, maar missend. B. [sleutel] C. Weet niet dat er uberhaupt iets moet komen. D. Verward met andere syntaxen.

item 34 onderdeel 5.1, 5.5, 4.4	Welke opties kan je meegeven als input om ‘kwak’ als uitvoer te krijgen? Er zijn **twee** goede opties, geef beiden.

	print('koe of kikker')
	dier = input()
	if dier == ‘koe’
	  print('boe')
	else:
	  print(‘kwak’)
	
	A. koe
	B. kikker
	C. niks als invoer meegeven
	D. alle worden mogelijk behalve het woord koe

bedoeling	De syntax van if-else statement begrijpen, dus weten wat wanneer wordt uitgevoerd. En vooral weten dat alles wat de if niet matched de else neemt.
sleutel	B, D
Bloom tax.	Application
redenering foute antwoorden	A. Leerling weet niet dat koe de if laat uitvoeren, of denkt dat al wordt de if uitgevoerd else altijd wordt uitgevoerd. B. [sleutel] C. Misconceptie, leerling weet niet dat een geen antwoord geven zorgt voor het eeuwig wachten. D. [sleutel]

B Uiteindelijke toetsing

1. Geef bij de onderstaande stellingen aan of ze juist of onjuist zijn.

   Een stukje commentaar in een programmeertaal wordt geschreven om
   A. een stukje code uit te voeren.
   B. boven het programma te schrijven wat het doet.
   C. letterlijk de code te herhalen
   D. een regel code ‘eventjes’ uit te zetten.

2. Welke declaratie van de lijst dieren klopt?

   A. dieren = [konijn, dolfijn, leeuw]
   B. dieren = konijn, dolfijn, leeuw
   C. dieren = [‘konijn’, ‘dolfijn’, ‘leeuw’]
   D. dieren = ‘konijn’, ‘dolfijn’, ‘leeuw’

3. Wat print de onderstaande code?

   ```
   #Dit is test code om iets te printen
   print(‘test 1, 2, 3’)
   #print(‘yey, het werkt’)
   ```

   A. test 1, 2, 3
   B. Dit is test code om iets te printen
      test 1, 2, 3
      yey, het werkt
   C. test 1, 2, 3
      yey, het werkt
   D. Dit is test code om iets te printen
      print(‘test 1, 2, 3’)
      print(‘yey, het werkt’)

4. Je krijgt een aantal codes met een uitvoer. Geef bij elke optie aan of de uitvoer juist of onjuist is.

   A. Code:
      ```
      print('Suiker', 'of', 'melk?')
      in_de_koffie = 'melk'
      input()
      print('Okee', 'ik', 'doe', 'er', in_de_koffie, 'in')
      ```

      Uitvoer:
      ```
      Suiker of melk?
      ```
      input:
      ```
      suiker
      ```
      Uitvoer:
      ```
      Okee ik doe er suiker in
      ```
   B. Code:
      ```
      print('kat', 'of', 'hond?')
      mijn_huisdier = input
      print('Mijn', mijn_huisdier, ‘is’, ‘weggelopen’)
      ```

      Uitvoer:
      ```
      kat of hond?
      ```
      input:
      ```
      hond
      ```

Uitvoer:
Mijn hond is weggelopen

C. Code:

```
print('Hoe laat is het?')
tijd = input()
print('Het is nu', tijd, '.')
```

Uitvoer:
Hoe laat is het?
input:
twaalf uur
Uitvoer:
Het is nu twaalf uur.

D. Code:

```
print('Hoe gaat het?')
humeur = input()
print('Goed', 'om', 'te', 'horen', 'dat', 'je', 'je, 'humeur', 'voelt!')
```

Uitvoer:
Hoe gaat het?
Input:
goed
Uitvoer:
Goed om te horen dat je je goed voelt!

5. Welke opties van deze printopdrachten print **Hello World 1.0** uit? Geef alle mogelijkheden.
   A. print Hello World 1.0
   B. print('Hello World 1.0')
   C. print('Hello', 'World', 1.0)
   D. print 'Hello World 1.0'
   E. print('Hello', 'World', '1.0')
   F. versie = '1.0'
      print('Hello World', versie)

6. Wat print de volgende code?

```
achternaam = 'Hermans'
print('Hallo', 'mevrouw', Hermans)
achternaam = 'van den Berg'
```

   A. Hallo mevrouw van den Berg
   B. Hallo mevrouw achternaam
   C. Hallo mevrouw Hermans
      Hallo mevrouw van den Berg
   D. FOUT

7. Wat print de volgende code?

```
kort = 'World'
lang = 'a'
print(kort, lang)
```

   A. World a

B. a a
C. kort lang
D. a World

8. Wat print de volgende code uit?

```
a= ‘vis’
b= ‘hond’
a= ‘kat’
print(a, b, b)
```

A. vis hond hond
B. vis hond kat
C. a b b
D. kat hond hond

9. We hebben een random lijst genaamd **kleuren** met 3 kleurnamen erin. Geef bij elke print aan of het een kloppende uitvoer zal geven.

A. print(‘Mijn kamer is’, kleur[3], ‘van kleur’)
B. print(‘Mijn kamer is’, kleur[-1], ‘van kleur’)
C. print(‘Mijn kamer is’, kleur[2], ‘van kleur’)
D. print(‘Mijn kamer is’, kleur[0], ‘van kleur’)

10. Match de print met de uitvoer

```
fruit = [‘appel’, ‘peer’, ‘banaan’]
```

A. print(fruit[0])
B. print(fruit[1])
C. print[fruit[2])
D. print(fruit[3])

1. banaan
2. appel
3. peer
4. *FOUT*

11. Wat is de juiste invoer die op "..." in de code moet komen om dezelfde uitvoer te genereren?

code:

```
talen = ['Python', 'JavaScript', 'HTML', ‘C++’, ‘C’, ‘Assembly’]
print(‘Dit programma heb ik gemaakt met’, ……)
print(‘Daarnaast kan ik programmeren in’, …...)
print(‘en binnenkort ga ik leren programmeren in’, ……)
```

Uitvoer:

```
Dit programma heb ik gemaakt met Python
Daarnaast kan ik programmeren in Javascript
en binnenkort ga ik leren programmeren in Assembly
```

Mogelijke antwoorden:

A. talen[1], talen[2], talen[5]
B. talen[0], talen[1], talen[5]

C. talen[1], talen[2], talen[6]
D. talen[0], talen[1], talen[6]

12. Wat moet er op de puntjes?

```
naam = input()
if naam …. ‘Elsa’:
   print(‘Hallo Elsa!’)
-------
input = Elsa
```

13. Welke foutmelding krijg je bij de volgende code?

```
achternaam = input(
------
input = Dane
```

A. NameError
B. IndentationError
C. SyntaxError

14. Welke van de volgende stukje codes geven een SyntaxError?

A. print(Waarden)
B. print(‘Waarden)
C. print(‘Waarden’
D. naam = input
E. naam = ‘input’
F. naam = ‘input

15. Welke foutmelding krijg je bij de volgende code?

```
if in_de_thee == 'suiker':
print(‘Thee met suiker’)
else:
  print(in_de_thee)
```

A. NameError
B. IndentationError
C. SyntaxError

16. Hebben de volgende twee codes dezelfde uitvoer?

```
code 1:
print('suiker', 'of', 'melk?')
in_de_thee = input()
if in_de_thee == 'melk':
   print('gieten')
else:
   print('schudden')
---
Input: koffie
-----------------------------------
code 2:
```

```
print('suiker', 'of', 'melk?')
in_de_thee = input()
if in_de_thee == 'melk':
   print('gieten')
else:
   print('schudden')
---
Input: suiker
```

A. Ja
B. Nee

17. Zijn de volgende codes juist of onjuist?

1.
```
print('reptiel of zoogdier?')
diersoort = ‘reptiel’
if diersoort == 'reptiel':
print('legt', 'een', 'ei')
else:
print('geeft', 'melk')
```
2.
```
slapen = ‘ja’
if slapen == ‘ja’:
   print('Slaap lekker')
else:
   print('laten we een film kijken')
```
3.
```
jarig = ‘vandaag’
if jarig ==  ‘vandaag’:
   print('Gefeliciteerd!')
```
4.
```
print('koe of kikker')
dier = ‘kikker’
if dier == ‘koe’
   print('boe')
```

18. Welke opties kan je meegeven als input om ‘<u>kwak’</u> als uitvoer te krijgen? Er zijn **twee** goede opties, geef beiden.

```
print('koe of kikker')
dier = input()
if dier == ‘koe’:
   print('boe')
else:
   print(‘kwak’)
```

A. koe
B. kikker
C. niks als invoer meegeven
D. alle worden mogelijk behalve het woord koe

C Antwoordblad uiteindelijke toetsing

Antwoordblad toetsing en aantal punten:

1- Onjuist, juist, onjuist, juist [3, -0.75 per foute antwoord]
2- C [5]
3- A [4]
4- Onjuist, onjuist, juist, onjuist [5, -1.25 per foute antwoord]
5- B, E, F [1, -0.5 per foute antwoord]
6- D [5]
7- A [5]
8- D [5]
9- Onjuist, onjuist, juist, juist [5, -1.25 per foute antwoord]
10- A2, B3, C1, D4 [5, -1.5 per foute antwoord]
11- B [5]
12- == [1]
13- C [6]
14- B, C, F [5, -2 per foute antwoord]
15- B [6]
16- A [2]
17- Onjuist, juist, juist, onjuist [5, -1.25 per foute antwoord]
18- B, D [3, -2 per foute antwoord]

Vragen van versie 1 die na expert review eruit zijn gehaald:

3- Zwakke formulering.
4- Laag niveau.
5- Weinig nut.
8- Irrelevant, zwakke formulering en weinig nut.
10- Zwakke formulering en weinig nut.