Bachelorscriptie
Informatica

Radboud Universiteit

Leren programmeren in Python

Auteur:
Tessa Schlief
s4207696

Inhoudelijk begeleider:
prof. dr. Erik Barendsen
e.barendsen@cs.ru.nl

Tweede begeleider:
dr. Sjaak Smetsers
s.smetsers@science.ru.nl

18 april 2017

Samenvatting

Dit is een casestudy waarin gekeken wordt naar een cursus leren programmeren in Python voor beginners. De vooruitgang van leerlingen is gemonitord aan de hand van de voorkennis van de leerlingen, vragen over de stof voor en na de bijeenkomsten, analyse van uitwerkingen van de opdrachten, opdrachten uit de literatuur en opnames van leerlingen aan het werk. Hieruit blijkt dat leerlingen wel vorderen en voor een deel in de concreet operationele fase eindigen. Dit zou betekenen dat leerlingen in staat zijn om te programmeren aan het einde van de Pythoncursus. Echter bevond een deel van de leerlingen zich nog in de pre-operationele fase zitten van het leren programmeren. Dit kan betekenen dat leerlingen wel kennis hebben van de theorie, maar het programmeren in de praktijk nog lastig is.

Voorwoord

Dit onderzoek is uitgevoerd voor een bachelor scriptie voor de studie Informatica aan de Radboud Universiteit in Nijmegen. Graag wil ik hier mijn scriptiebegeleiders Erik Barendsen en Sjaak Smetsers bedanken voor al hun hulp door middel van het doen van suggesties voor onderzoek en door feedback op mijn geschreven stukken. Ook wil ik Koen, Marijn en Tanja bedanken voor hun hulp met het verzamelen van de data. Zij hebben er iedere bijeenkomst voor gezorgd dat de leerlingen mijn vragenlijst konden vinden en ze hebben er ook voor gezorgd dat alles zo veel mogelijk ingevuld werd.

Inhoudsopgave

Inleiding

Er is veel aandacht voor programmeren in het onderwijs. Deze aandacht richt zich onder andere op de leeftijd waarop leerlingen kunnen beginnen met programmeren (Clements & Gullo, 1984; Webb, Davis, Bell, Katz, Reynolds, Chambers & Sysło, 2016) en op manieren waarop programmeren aangeleerd kan worden (Kahn, 1999; Matthíasdóttir, 2006). Er zijn heel veel verschillende programmeertalen die men kan leren zoals Java, Python en C++. Het proces van leren programmeren kan moeilijkheden opleveren en dat wordt door verschillende onderzoekers bevestigd (McCracken et al., 2001; Lister, 2016).

Een praktijkvoorbeeld van leren programmeren is de cursus programmeren voor beginners in de taal Python. Dit is een cursus die wordt gegeven aan de Radboud Universiteit in Nijmegen voor middelbare scholieren. De leerlingen voor deze cursus zijn in eerste instantie geselecteerd op basis van hun score voor de Beverwedstrijd. Dit is een wedstrijd voor leerlingen van het basis- en middelbaar onderwijs met vragen waarin belangrijk concepten van de Informatica en Wiskunde zijn verwerkt. De cursus is al een aantal keer gegeven, maar er is nog niet onderzocht hoe effectief de cursus is. Daarom is het nu de vraag wat de deelnemers van de cursus leren en of er misschien verbeterpunten voor de cursus zijn.

Dit heeft geleid tot de hoofdvraag van dit onderzoek:
Hoe effectief is de cursus programmeren voor beginners in Python?

Om dit te meten wordt er vooral gekeken naar de ontwikkeling van de leerlingen tijdens de Pythoncursus. Om de ontwikkeling van de deelnemers te monitoren wordt er in dit onderzoek onder andere gebruik gemaakt van de theorie van Lister (2016) en van McCracken et al. (2001). Uitleg over deze theorieën, en anderen, wordt gegeven in het Theoretisch kader (p. 7). Vervolgens worden er aan de hand van de theorie een aantal concrete deelvragen gesteld. De details over de Pythoncursus worden daarna ook gegeven. In de methodesectie (p. 14) staat beschreven welke tests er gedaan zijn en hoe die uitgevoerd zijn. De resultaten van alle tests worden daarna weergegeven. Dit document eindigt met het antwoord op de hoofd- en deelvragen in de Conclusie (p. 27). Een eventuele verklaring voor de verschillen tussen de theorie en de praktijk van de Pythoncursus wordt gegeven in de Discussie (p. 28).

Door dit onderzoek worden de volgende bijdragen geleverd:

- Een analyse van de ontwikkeling en mogelijke invloeden op de ontwikkeling van leerlingen tijdens deze beginnerscursus programmeren in Python;
- Een evaluatie van manieren waarop de ontwikkeling van leerlingen tijdens een programmeercursus gemonitord kan worden;
- Een concrete bijdrage aan het verbeteren van het programmeeronderwijs in de vorm van een evaluatie van de cursus zoals die nu is.

Theoretisch kader

Leren programmeren is voor de meeste mensen geen makkelijke opgave. Onderzoekers willen graag weten waar beginnende programmeurs moeite mee hebben. Er is onder andere door Lister (2016) en McCracken et al. (2001) onderzoek gedaan naar het proces van leren programmeren bij beginners. Beiden hebben een eigen theorie over bepaalde onderdelen van het leerproces. Eerst zal de theorie van McCracken et al. uitgelegd worden en daarna de theorie van Lister.

Stappenplan

McCracken et al. (2001) hebben vijf stappen gedefinieerd voor het schrijven van een programma, of ook wel het oplossen van een probleem:

1. *Abstraheer* het probleem van de beschrijving in de opdracht. Het probleem wordt meestal gegeven aan de hand van een concreet voorbeeld. Leerlingen moeten dus eerst zelf het algemene probleem uit het specifieke voorbeeld halen.
2. Genereer *sub-problemen*. Om een groot probleem op te kunnen lossen, wordt het probleem eerst opgedeeld in kleinere (makkelijker oplosbare) problemen.
3. Maak *sub-oplossingen* voor de sub-problemen. Los eerst de kleinere problemen op.
4. *Voeg* de sub-oplossingen *samen* tot een werkend programma. Maak hier dus een samenhangend geheel van alle sub-oplossingen.
5. *Evalueer* en *itereer*. Zolang blijkt dat het probleem nog niet volledig opgelost is, herhaal je (een deel van) de vorige stappen.

Het uit kunnen voeren van deze vijf stappen representeert het probleemoplossend vermogen van de leerlingen in het onderzoek van McCracken et al. (2001). Uit dit onderzoek kwam naar voren dat één onderdeel van het stappenplan het vaakst werd overgeslagen door de leerlingen. Dit was stap 1: het abstraheren van het probleem uit de beschrijving in de opdracht. Dit zou erop wijzen dat het probleemoplossend vermogen van de leerlingen op dat moment nog niet groot genoeg was om programmeeropdrachten te kunnen voltooien. In dit onderzoek is onder andere gekeken naar het gebruik van de stappen uit dit stappenplan door beginnende programmeurs.

Fases in het leerproces

In het onderzoek van Lister (2016) wordt ook het leerproces besproken. Net als McCracken et al. (2001), onderzoekt Lister welke vaardigheden leerlingen beheersen (of nog niet beheersen) die nodig zijn om te kunnen leren programmeren. Voorbeelden van vaardigheden die noodzakelijk zijn volgens de theorie van Lister zijn:

- Het begrijpen van basisprincipes van het programmeren en
- Het systematisch kunnen uitvoeren van code in de vorm van traceren.

Als je code traceert, kijk je regel voor regel naar code en je bekijkt dan per regel wat er gebeurt in het programma.

Om te testen of het klopt dat leerlingen bovengenoemde vaardigheden missen, gebruikt Lister (2004) twee soorten opdrachten.

1. Het voorspellen van de uitkomst van een programmaatje.
2. Het bepalen welk stukje ingevuld moet worden op een open plek in een programma om het programma te laten werken.

Door Lister (2016) wordt het wel of niet beschikken over bepaalde vaardigheden gekoppeld aan fases. Door Jean Piaget worden vier fases onderscheiden van leren/denken bij jonge kinderen: sensorimotor, pre-operationeel, concreet operationeel en formeel operationeel. Deze fases gebruikt Lister om te redeneren over kinderen die leren programmeren. De vier fases worden voor dat doeleinde als volgt uitgelegd (Lister, 2016):

1. Sensorimotor (ook wel pre-traceerfase): in deze fase hebben leerlingen een onsamenhangend begrip van code/programma's.
2. Pre-operationeel (ook wel traceerfase): Leerlingen kunnen code met de hand volgen en daarmee in gedachten of hardop uitvoeren (traceren). Ze zoeken uit wat een programma doet door regel per regel te kijken wat er gebeurt in de code en door het verschil tussen de invoer en de uitvoer van het programma te analyseren.
   Er kan binnen deze fase nog onderscheid worden gemaakt tussen het begin en het einde van de fase:
   Beginnend pre-operationeel: hier voeren leerlingen nog maar één trace uit, met maar één set initiële variabelen.
   Gevorderd pre-operationeel: aan het einde voeren de leerlingen al meerdere traces uit en dit met verschillende initiële waardes. Er worden dan ook bewust verschillen in uitkomst gezien door leerlingen bij het kiezen van een andere invoer.
3. Concreet operationeel (ook wel post-traceerfase/abstracte traceerfase): Hier laten leerlingen al een doelgerichte aanpak zien om code te gaan schrijven. In plaats van het kijken naar specifieke initiële variabelen, worden hier klassen van initiële waardes ingevoerd. Dan gaat het bijvoorbeeld over traceren met a < b, a = b en a > b als het gaat over variabelen a en b. Er wordt dus niet meer nagedacht over specifieke initialisatie, maar over voorwaardes voor en beperkingen van de mogelijke waardes van de variabelen.
4. Formeel operationeel: Experts, (bijna) volleerde programmeurs, zitten in deze fase. Onderdelen van deze fase zijn het kunnen reflecteren op en debuggen van code. Hiervoor moet de programmeur logisch, consequent en systematisch kunnen redeneren over code.

Uit het onderzoek van Lister (2016) blijkt dat de moeilijkheden in het proces van leren programmeren al beginnen bij de taken die hierboven beschreven zijn (traceren en code aanvullen). Dat leerlingen deze taken nog niet uit konden voeren zou, op het moment van onderzoeken, wijzen op een gebrek aan vereiste vaardigheden bij desbetreffende leerlingen om te kunnen programmeren. De leerlingen bevonden zich tijdens het onderzoek in de eerste of tweede fase van Piaget/Lister en waren dus nog niet in de fase waarin ze in staat zijn om te gaan programmeren (de concreet operationele fase). Lister concludeert dat het kunnen lezen, begrijpen en aanvullen van code een vereiste is om te kunnen programmeren. In dit onderzoek is onder andere ook bekeken in welke fase deelnemers van een beginnerscursus programmeren zich bevinden. Hiervoor zijn een aantal opdrachten uit het onderzoek van Lister gebruikt zoals de opdracht die hieronder beschreven wordt.

Tijdens zijn keynote voordracht voor de Workshop in Primary and Secondary Computing Education (WiPSCE) van 2016 haalt Raymond Lister de volgende voorbeeldopdracht aan:

> The purpose of the following three lines of code is to swap the values in variables a and b, for any set of possible values stored in those variables. Assume that variables a, b and c have been declared and initialized.
>
> ```
> c = a;
> a = b;
> b = c;
> ```
>
> . . . In one sentence . . ., describe the purpose of the variable "c" in the above code. (Lister 2016)

Het doel van variabele c, het antwoord op de vraag, is het tijdelijk opslaan van een waarde. Het is een tijdelijke variabele. Uit het antwoord dat op deze vraag gegeven wordt zou je een idee kunnen krijgen of de ondervraagde de code begrijpt. In zijn presentatie op het WiPSCE stelt Lister (2016) dat

leerlingen in de concreet operationele fase moeten zitten om een correct antwoord te kunnen geven op deze vraag. In het onderzoek van Lister gaf 63% van de studenten het goede antwoord (N=98).

Regenprobleem

Het Regenprobleem is een voorbeeld van een relatief simpel probleem waar desondanks veel beginnende programmeurs moeite mee hebben. Een regenprobleem is een probleem waar een programma getallen op moet tellen tot het een bepaalde waarde tegenkomt. Als die waarde wordt gezien, moet het programma het gemiddelde berekenen van alle voorgaande waarden behalve de laatste. Een voorbeeld van een regenprobleem:

> The Averaging Problem
> Write a program that repeatedly reads in integers, until it reads the integer 99999. After seeing 99999, it should print out the correct average. That is, it should not count the final 99999. (Regenprobleem, Soloway, Bonar & Ehrlich, 1983)

In een onderzoek van Soloway & Bonar (1982) werd deze opdracht voorgelegd aan leerlingen aan het einde van een beginnerscursus programmeren in de taal Pascal. Slechts 38% van de leerlingen gaf een correcte oplossing van het probleem. Vervolgens werd de vraag herhaald bij leerlingen die al een tweede programmeercursus hadden gevolgd. Hieruit kwam maar een percentage van 60% correcte oplossingen. Ook in dit onderzoek zijn oplossingen van het regenprobleem van beginnende programmeurs geanalyseerd.

Beverwedstrijd

De Beverwedstrijd is een wedstrijd waarin kennis van leerlingen wordt getest. De proefpersonen in dit onderzoek zijn in eerste instantie geworven aan de hand van hun hoge scores voor deze wedstrijd. Uitleg over wat de Beverwedstrijd precies inhoudt:

> . . . informaticawedstrijd voor alle leerlingen uit groep 5, 6, 7 & 8 van de basisschool en klas 1 t/m 6 van het voortgezet onderwijs.
> De wedstrijd is ontwikkeld om leerlingen kennis te laten maken met informatica.
> Misschien ontdekken ze wel dat ze er talent voor hebben!
>
> Deelnemers krijgen verschillende vragen die te maken hebben met informatica, algoritmen, structuren, informatieverwerking, logica, puzzels en toepassingen. Pittige onderwerpen, maar je hebt geen voorkennis nodig.
> De opgaven zijn zó gemaakt dat iedereen ze met logisch en goed nadenken kan oplossen.
> De vraag is of je dit binnen de gegeven tijd lukt... (http://www.beverwedstrijd.nl)

De bevervragen van een aantal afgelopen jaren zijn geanalyseerd door Barendsen et al. (2015). De vragen van de Beverwedstrijd van 2010 tot en met 2014 zijn geclassificeerd op concept van Computational Thinking (CT). Door Computer Science Teachers Association en de International Society for Technology (ISTE) is een uitleg gemaakt van CT waarbij CT wordt onderverdeeld in negen concepten (Barendsen et al., 2015): data verzameling, data-analyse, data representatie, probleem decompositie, abstractie, algoritmes en procedures, automatisering, parallellisatie en simulatie. Belangrijk voor dit onderzoek zijn de concepten Probleem Decompositie (PD) en Algoritmes & Procedures (AL). Om deze reden zijn er alleen maar vragen van deze twee categorieën gebruikt in dit onderzoek.
Door de makers van de bevervragen is een inschatting gemaakt van de moeilijkheidsgraad van de opdrachten voor bepaalde leeftijdscategorieën. Die leeftijdscategorieën zijn op volgende manier ingedeeld (Barendsen et al.):

(0) Primary, 8-9 years (grade 3-4)
(I) Benjamin, 10-12 years (grade 5-6)
(II) Cadets, 13-14 years (grade 7-8)
(III) Junior, 15-16 years (grade 9-10)
(IV) Senior, 17-19 years (grade 11-13)

Codekwaliteit

Geschreven code kan op verschillende manieren worden beoordeeld. Als je code uitvoert, dan werkt het of dan werkt het niet. Maar de kwaliteit van de code zelf kun je ook beoordelen. Stegeman, Barendsen & Smetsers (2016) hebben een rubric ontworpen aan de hand waarvan feedback op codekwaliteit kan worden gegeven.

LEVEL	1	2	3	4
names	names appear unreadable, meaningless or misleading	names accurately describe the intent of the code, but can be incomplete, lengthy, misspelled or inconsistent use of casing	names accurately describe the intent of the code, and are complete, distinctive, concise, correctly spelled and consistent use of casing	all names in the program use a consistent vocabulary
headers	headers are generally missing or descriptions are redundant or obsolete; use mixed languages or are misspelled	header comments are generally present; summarize the goal of parts of the program and how to use those; but may be somewhat inaccurate or incomplete	header comments are generally present; accurately summarize the role of parts of the program and how to use those; but may still be wordy	header comments are generally present; contain only essential explanations, information and references
comments	comments are generally missing, redundant or obsolete; use mixed languages or are misspelled	comments explain code and potential problems, but may be wordy	comments explain code and potential problems, are concise	comments are only present where strictly needed
layout	old commented out code is present or lines are generally too long to read	positioning of elements within source files is not optimized for readability	positioning of elements within source files is optimized for readability	positioning of elements is consistent between files and in line with platform conventions
formatting	formatting is missing or misleading	indentation, line breaks, spacing and brackets highlight the intended structure but erratically	indentation, line breaks, spacing and brackets consistently highlight the intended structure	formatting makes similar parts of code clearly identifiable
flow	there is deep nesting; code performs more than one task per line; unreachable code is present	flow is complex or contains many exceptions or jumps; parts of code are duplicate	flow is simple and contains few exceptions or jumps; duplication is very limited	in the case of exceptions or jumps, the most common path through the code is clearly visible
idiom	control structures are customized in a misleading way	choice of control structures is inapproriate	choice of control structures is appropriate; reuse of library functionality may be limited	reuse of library functionality and generic data structures where possible
expressions	expressions are repeated or contain unnamed constants	expressions are complex or long; data types are inappropriate	expressions are simple; data types are appropriate	expressions are all essential for control flow
decomposition	most code is in one or a few big routines; variables are reused for different purposes	most routines are limited in length but mix tasks; routines share many variables instead of having parameters	routines perform a limited set of tasks divided into parts; use of shared variables is limited	routines perform a very limited set of tasks and the number of parameters and shared variables is limited
modularization	most code is in one or a few large modules; or modules are artificially separated	modules have mixed responsibilities, contain many variables or contain many routines	modules have clearly defined responsibilities, contain few variables and a somewhat limited amount of routines	modules are defined such that communication between them is limited

- for each criterion, circle the level that is most representative of the features that are present
- no need to circle a level that is not relevant to the assignment
- level 2 implies that the features in level 1 are not present, level 4 implies that the features in level 3 are also present

Welk level een programmeur scoort op de criteria in de rubric zegt iets over de kwaliteit van de code. Codekwaliteit gaat over de structuur en stijl van geschreven code. Als een leerling op alle criteria level 4 scoort, dan schrijft deze leerling dus kwalitatief goede code. Voor dit onderzoek gaan we er

dan van uit dat kwalitatief slechte code erop duidt dat degene die de code geschreven heeft nog vaardigheden mist die nodig zijn om goed te kunnen programmeren. De kwaliteit van de code is hier dus een indicatie van hoe goed een leerling kan programmeren. De rubric is in dit onderzoek gebruikt om te kijken naar hoe de kwaliteit van de code in verband staat met de individuele ontwikkeling van de leerlingen tijdens een beginnerscursus programmeren in Python.

Doel van de studie

Om de hoofdvraag te kunnen beantwoorden, zijn er voor dit onderzoek ook een paar concrete deelvragen geformuleerd.

Hoofdvraag
Hoe effectief is de cursus programmeren voor beginners in Python?

Deelvragen

- Welke vorderingen maken de leerlingen tijdens deze cursus?
- Wat is de relatie tussen de achtergrond van de leerlingen en de vorderingen die ze maken?

Context van de studie

Dit is een casestudy waarin gekeken wordt naar verschillen tussen studenten en verschillende manieren van monitoren van voortgang tijdens een cursus. Het gaat in dit onderzoek om een relatief nieuwe cursus. 28 september 2016 is een cursus gestart voor middelbare scholieren om te leren programmeren in Python (http://cs.ru.nl/pythoncursus/). Er hebben zich 40 leerlingen van het voortgezet onderwijs in Nederland aangemeld. Het zijn bijna allemaal leerlingen uit de 5^{e} klas van het vwo (15-18 jaar). De cursus bestaat uit vijf bijeenkomsten op woensdagmiddag van 14.00 uur tot 17.30 uur in een computerlokaal op de Radboud Universiteit te Nijmegen. Voorafgaand aan de bijeenkomsten worden de leerlingen geacht theorie te hebben geleerd via een online cursus van de Universiteit van Waterloo (http://cscircles.cemc.uwaterloo.ca/nl/). Tijdens de bijeenkomsten zijn er drie student-assistenten aanwezig die aan het begin van de bijeenkomst een korte presentatie houden over de stof van die week. Daarna kunnen de leerlingen aan de slag met opdrachten die door de student-assistenten worden verstrekt en kunnen zij natuurlijk vragen stellen aan de student-assistenten.

Methode

In dit onderzoek zijn een aantal indicatoren gebruikt om de hoofd- en deelvragen te kunnen beantwoorden. Deze indicatoren zijn gemeten aan de hand van verschillende methoden met verschillende middelen. Hieronder een schematische weergave van wat onderzocht is en daarbij op welke manier en wanneer:

Deelonderwerp	Indicatoren:	Methode/middelen	Wanneer
Gemaakte vorderingen	Verloop van de algoritmische kennis	Testjes in de vorm van oude vragen van de Beverwedstrijd, vragen over codefragmenten en het regenprobleem	Bijeenkomst 2 – 5
	Resultaat van het programmeren	Werking van de code en codekwaliteit beoordelen	Na de bijeenkomsten aan de hand van ingeleverde opdrachten
	Het proces van het programmeren	Analyseren van opnames van een aantal teams	Bijeenkomst 4
	Indruk van de leerlingen zelf	Vragen aan de leerlingen of ze bepaalde onderwerpen uit de theorie denken te begrijpen	Iedere bijeenkomst een aantal ‘exitvragen’
Achtergrond factoren	Ervaring met programmeren	Vragen over ervaring met programmeren en programmeertalen	Eerste bijeenkomst
	Voorkennis	Vragen over voorkennis	Eerste bijeenkomst
	Interesse in schoolvakken	Vragen over schoolvakken	Eerste bijeenkomst
	Profielkeuze	Vragen naar profielkeuze	Eerste bijeenkomst
	Score van Beverwedstrijd	Vragen naar de score van de Beverwedstrijd	Eerste bijeenkomst

Tabel 1: schematische weergave methode

In de rest van deze methodesectie wordt eerst beschreven per databron hoe de data verzameld en geanalyseerd is. Daarna wordt beschreven hoe de verzamelde gegevens gebruikt zijn om te kijken naar de vorderingen die leerlingen hebben gemaakt en om de invloed van achtergrondfactoren te bespreken.

Instaptest

Als je al voorkennis hebt van bijvoorbeeld Python voordat je een cursus Python gaat doen lijkt het logisch dat deze cursus dan minder moeite kost dan wanneer er geen relevante voorkennis is. Dit werd getest door middel van een vragenlijst aan het begin van de cursus over de achtergrond van de leerlingen. In die vragenlijst kwamen alle onderdelen aan bod die met achtergrond te maken hebben zoals in Tabel 1 aangegeven is. Het is moeilijk om aan de hand van een korte vragenlijst vast te stellen wat de voorkennis van de leerlingen precies inhoudt. Om toch een globaal idee te krijgen zijn er vragen gesteld die gerelateerd zijn aan programmeren. De vragenlijst bestond uit een aantal vragen over interesses en vakken op school en daarna vragen over programmeerervaring. Er werd

gevraagd naar ervaring in termen van veel/een beetje/geen programmeerervaring en 'In welke programmeertalen heb je ooit geprogrammeerd?'. Vervolgens werd er gevraagd in hoeverre de leerlingen bepaalde concepten uit het programmeren al kenden en/of hadden toegepast. Zo kon een globaal idee gevormd worden van de voorkennis van de leerlingen. Zie de bijlage voor de hele vragenlijst.
In combinatie met de analyse van de individuele ontwikkeling van de leerlingen is de relatie tussen de achtergrond van een leerlingen en hoe goed hij/zij het heeft gedaan tijdens deze cursus beoordeeld.

Bevervragen

In eerste instantie zijn leerlingen voor deze Pythoncursus gevraagd die een hoge score hadden voor de Beverwedstrijd. Voor het volgen van de algoritmische vaardigheden is gebruik gemaakt van oude opdrachten van de Beverwedstrijd. Deze opdrachten zijn gekozen met behulp van de classificatie van bevervragen van Barendsen et al. (2015). Belangrijke concepten voor dit onderzoek zijn Probleem Decompositie (PD) en Algoritmes & Procedures (AL). Dit zijn namelijk de concepten die aan bod komen in de cursus. Zie Tabel 2 voor de vragen en categorieën. In de tabel staat bij het niveau van de vraag de leeftijdscategorie en de moeilijkheidsgraad die daarbij hoort. De indeling van de leeftijdscategorieën staat gegeven in het theoretisch kader. De leerlingen van de cursus waren allemaal junior/senior, aangezien ze tussen de 15 en 18 waren.
Aan het begin van bijeenkomst 2, 3 en 4 is de leerlingen gevraagd twee oude bevervragen te maken. Dat hebben zij online ingevuld op een Google form. Voor gestelde vragen zie bijlage.

Naam opdracht	**Categorie opdracht**	**Niveau vraag**	**Uitleg opdracht**
Hoekjes Knippen	AL	III easy IV easy	Waar kan een gat in het papier zitten als je een hoekje afknipt van een gevouwen blaadje?
Tangram	PD	III easy IV easy	Welke figuur kan niet gemaakt worden met de blokjes?
Dobbelen	AL	III medium IV medium	If then else vraag. Wat moet er achtereenvolgens gegooid worden voor een bepaalde uitkomst?
School-feestje	PD	III easy IV easy	Database vraag. Welke gegevens heb je nodig om te kunnen bepalen of iemand een 18+ bandje krijgt op het feest?
MP3	AL	III medium IV easy	Hash vraag. Welke naam krijgt een nieuw liedjes aan de hand van de regels voor het bedenken van een naam?
Strafwerk	AL	III medium IV easy	Hoe is het strafwerk het snelst af met keuze uit verschillende combinaties van typen, kopiëren en plakken.

Tabel 2: Bevervragen

Zowel bij deze bevervragen als bij de vragen over codefragmenten van Lister (2004, 2016) die hieronder beschreven worden, is gebruikt gemaakt van meerkeuze vragen. Meerkeuze vragen kunnen objectief beoordeeld worden, want er is maar één antwoord goed. Er is geen ruimte voor discussie of interpretatie. Zoals Lister ook vermeld, is er natuurlijk wel een gok kans van 25% (bij vier mogelijke antwoorden) of 20% (bij vijf mogelijke antwoorden).

Regenprobleem

De leerlingen hebben tijdens de laatste bijeenkomst de vraag gekregen een oplossing op te schrijven (in een Google form) voor het regenprobleem:

> Schrijf een Python programma dat integers blijft inlezen (invoer door de gebruiker) totdat het getal 999 wordt ingelezen. Print vervolgens het gemiddelde van de ingelezen getallen. In dit gemiddelde is het getal 999 niet meegenomen. Het is de bedoeling dat je het programma hieronder typt en niet eerst in PyCharm (of ergens anders) gaat testen.

Hoewel onderzoek anders uitwijst (Soloway & Bonar, 1982; Mulholland & Eisenstadt, 1998), zou dit een relatief makkelijk probleem moeten zijn voor leerlingen aan het einde van een beginnerscursus programmeren. Er is getest hoe de leerlingen van deze cursus het regenprobleem oplossen. Alle antwoorden van de leerlingen zijn bekeken en in PyCharm uitgevoerd. Per leerlingen is genoteerd of het antwoord goed of fout was en in dat laatste geval is ook genoteerd wat er dan fout ging.

Exitvragen

Theorievragen

Ook de ervaringen van de leerlingen zelf zijn meegenomen in dit onderzoek. Hiervoor is aan de leerlingen gevraagd een inschatting te maken van hun eigen kennis. De vragen gingen over de theorie van de online Pythoncursus Computer Science Circles (https://cscircles.cemc.uwaterloo.ca/nl/). De leerlingen hebben zelf aangegeven in hoeverre zij dachten bepaalde onderdelen van de theorie te begrijpen en/of toe te kunnen passen. Ze hebben per bijeenkomst een aantal stellingen gekregen waarbij ze aan konden geven welk antwoord volgens hen het meest van toepassing was op henzelf. Een voorbeeld:

> Ik weet wat een array is en waarvoor ik deze kan gebruiken.
> ⬭ Ja
> ⬭ Ik heb een idee
> ⬭ Nee

Zie bijlage voor alle vragen.

Evaluatie opdracht van de week

Iedere week hebben de leerlingen na de theorievragen ook de volgende vragen gekregen:

1. Ik heb de opdrachten af: Ja / Bijna / Nee
2. Ik vond de opdrachten: Makkelijk / Goed te doen met hulp van student-assistenten / Moeilijk

Hierna is bekeken in hoeverre de ervaring van de leerlingen overeenkwam met de resultaten van de andere verzamelde data. Zie hiervoor de sectie over individuele ontwikkelingen.

Vragen codefragmenten

Lister (2004, 2016) heeft verschillende onderzoeken gedaan naar het niveau van beginners die leren programmeren. Een aantal van zijn vragen is ook gesteld aan de leerlingen van de Pythoncursus. Het resultaat hiervan is vergeleken met de resultaten die Lister zag in zijn onderzoek. Op basis van de ideeën van Lister, zoals beschreven in de fases van Piaget in het theoretisch kader, kan ook een indicatie gegeven worden van het niveau van leren waarop de leerlingen zich bevinden.

In de exitvragen van bijeenkomst 3 zat één vraag uit onderzoek van Lister (2016). Dit was de vraag die als voorbeeld is besproken in het Theoretische kader (zie p. 8) over het doel van variabele c.
In de exitvragen van bijeenkomst 4 zitten drie vragen uit onderzoek van Lister (2004). Oorspronkelijk waren deze vragen niet in Python geschreven. De originele vragen zijn door een student-assistent van de Pythoncursus vertaald naar Python.

Bekijk het volgende stukje code:

```
x = [2, 1, 4, 5, 7]
limit = 3
i = 0
sum = 0

while sum < limit and I < len(x) :
   i += 1
   sum += x[i]
```

Welke waarde heeft variabele i nadat deze code is uitgevoerd?

a) 0
b) 1
c) 2
d) 3

Vraag 1 gaat over het loopen door een array tot een limiet is bereikt. Het juiste antwoord hier is C.

Bekijk het volgende stukje code:

```
x = [1, 2, 3, 3, 3]
b = [False, False, False, False, False]

for i in range (len(b)):
   b[i] = False

for i in range (len(x)):
   b[x[i]] = True

count = o

for i in range (len(b)):
   if b[i]:
      count += 1
```

Welke waarde heeft “count” nadat deze code is uitgevoerd?

a) 1
b) 2
c) 3
d) 4
e) 5

Vraag 2 gaat ook over arrays. Het juiste antwoord is C.

Bekijk het volgende stukje code:

```
x = [0, 1, 2, 3]
i = 0
j = len(x)-1

while i < j:
   temp = x[i]
   x[i] = x[j]
   x[j] = 2*temp
   i += 1
   j -=  1
```

Welke waarde heeft “x” nadat deze code is uitgevoerd?

a) [3, 2, 2, 0]
b) [0, 1, 2, 3]
c) [3, 2, 1, 0]
d) [0, 2, 4, 6]
e) [6, 4, 2, 0]

Vraag 3 gaat over het omdraaien en/of verdubbelen van de waardes in een array. Het juiste antwoord is A.

Analyse codekwaliteit

Er is aangenomen dat voor het proces van het leren programmeren ook de kwaliteit van geschreven code een indicator is van het niveau van de programmeur. De leerlingen hebben vanaf bijeenkomst 2 hun uitwerkingen van de opdrachten iedere keer via de mail ingeleverd. Deze uitwerkingen zijn geanalyseerd op werking en codekwaliteit met behulp van software van Atlas-ti. De opdrachten zijn in Atlas-ti gecodeerd aan de hand van criteria uit de rubric van Stegeman et al. (2016). Omdat sommige criteria niet van belang zijn voor dit specifieke onderzoek is de rubric aangepast. Headers, idiom, expressions en modularization zijn weggelaten. Daarnaast is ervoor gekozen om drie niveaus te identificeren in plaats van vier. Dit past beter bij de uitwerkingen van de leerlingen in dit onderzoek. Naast de criteria in de rubric zijn de uitwerkingen ook beoordeeld op hun werking. Hierbij is het criteria ‘werkt’ ook in drie niveaus verdeeld, namelijk 1 = werkt niet, 2 = werkt bijna/gedeeltelijk en 3 = werkt goed. In Tabel 3 staat de aangepaste versie van de rubric die gebruikt is om opdrachten te coderen.

Niveau Criteria	1	2	3
Names	Niet gebruikt, niet aanwezig	Niet voor alles een duidelijk naam, nauwelijks gebruikt	Duidelijke namen voor variabelen en functies
Comments	Niet aanwezig of heel stuk oude code	Aanwezig maar niet overal nuttig	Aanwezig en verduidelijken programma
Layout	Oude code staat nog als commentaar, te lange regels	De positie van de code verbetert leesbaarheid niet/nauwelijks, onnodig omdat kort programma	De positie van de code verbetert leesbaarheid
Format	Niet aanwezig	Inspringen en haakjes zorgen hier en daar voor structuur	Inspringen, haakjes en witregels zorgen voor een duidelijke structuur
Flow	Diepe nesting, onbereikbare code, heel veel codeduplicatie, verkeerde volgorde code	Flow kan efficiënter, codeduplicatie, jumps en/of excepties	Goede flow, hoofdroute is duidelijk
Decompositie	Alle code in één groot programma (bij veel kleine opdrachtjes niet erg)	Gebruik van hulpfuncties	Goede decompositie

Tabel 3: Aangepaste rubric voor analyse codekwaliteit

In eerste instantie zijn alleen de uitwerkingen gecodeerd van de 6 leerlingen waarvan ook opnames gemaakt zijn.

Opnames

Tijdens bijeenkomst 4 zijn opnames gemaakt van het scherm van een aantal leerlingen en ook het geluid is daarbij opgenomen. Dit is gebeurd met behulp van de software van screencast-o-matic (https://screencast-o-matic.com/screen_recorder) en een camera met microfoon. Zes leerlingen is gevraagd om tijdens de opnames in tweetallen te werken zodat de kans groter was dat ze hardop zouden overleggen en daarmee dus zouden laten horen hoe ze de opdrachten hebben aangepakt en waar ze tegenaan liepen.

De opnames zijn deels letterlijk, deels globaal getranscribeerd en bekeken. Dat betekent dat de opnames zijn bekeken en tegelijkertijd is genoteerd wat de leerlingen deden en wat ze hardop zeiden. Daarna zijn de transcripties en de opnames geanalyseerd op basis van een aantal indicatoren om de praktijk te kunnen vergelijken met de theorie. De volgende indicatoren/vragen zijn beoordeeld tijdens de analyse:

- Gebruiken de leerlingen de vijf stappen van McCracken et al. (2001) (abstraheren, sub-problemen, sub-oplossingen, samenvoegen, evalueren en itereren)?
- In hoeverre maken de leerlingen gebruik van traceren?
- In hoeverre maken de leerlingen gebruik van de voorbeelden uit de lesstof?

De zes leerlingen waarvan opnames zijn gemaakt, zijn geselecteerd op hun verschil in resultaten tot aan bijeenkomst 4.
De opdracht die ze hebben gemaakt is uitgekozen op het soort opdracht. Het ging bij deze opdracht om sorteeralgoritmes waarbij algoritmische vaardigheid aan bod komt.

Individuele ontwikkeling

Om de vorderingen van de leerlingen te meten tijdens deze cursus is gebruikt gemaakt van de volgende onderdelen van de methode:

- Bevervragen
- Exitvragen
- Vragen codefragmenten
- Regenprobleem
- Analyse codekwaliteit

Al deze onderdelen zijn hierboven verder uitgelegd. Aan de hand van de antwoorden op al deze vragen en aan de hand van de ingeleverde uitwerkingen van de opdrachten is een analyse gemaakt van hoe de leerlingen het doen tijdens de cursus. En daarnaast zijn de opnames nog bekeken om eventuele verklaringen voor de vorderingen van enkele leerlingen te kunnen vinden. In een tabel zijn alle onderdelen tegen de zes geselecteerde leerlingen van de opnames uitgezet en per onderdeel is aangegeven hoe de leerling scoort (goed/fout of een korte uitspraak).

Achtergrondfactoren

De data die verzameld is met de instaptest is gebruikt om iets te zeggen over de eventuele relatie tussen de achtergrond van de leerlingen en de behaalde resultaten in de cursus. Hiervoor zijn de resultaten van de instaptest vergeleken met de resultaten van de vorderingen die gemaakt zijn door de leerlingen.

Resultaten

Instaptest

De antwoorden op de instaptest geven de achtergrond van de leerlingen weer. Deze informatie is gebruikt om de relatie tussen de achtergrond en de behaalde resultaten tijdens de cursus te beschrijven. Een aantal observaties over de instaptest:

- De meeste leerlingen hebben een Natuur & Techniek of Natuur & Gezondheid profiel gekozen op school. Van de 34 leerlingen hebben er 4 een Economie & Maatschappij of Cultuur & Maatschappij profiel gekozen.
- De lievelingsvakken van de leerlingen zijn allemaal bètavakken, af en toe gecombineerd met gym of één alfa vak.
- Er zijn maar 3 leerlingen met veel ervaring met programmeren. Er zijn er 7 zonder ervaring. De overige leerlingen, de meeste, hebben dus een beetje ervaring.
- De leerlingen gaven aan de meeste concepten al te kennen of een idee van de betekenis ervan te hebben. Concepten als loops, while-loops, for-loops, arrays en recursie zijn bij veel leerlingen helemaal niet bekend of ze hebben alleen maar een idee van wat het is.
- De helft van de leerlingen (17) heeft één of meerdere keren meegedaan aan de Beverwedstrijd.

Bevervragen

Aan het begin van bijeenkomst 2, 3 en 4 hebben de leerlingen twee bevervragen beantwoord. In onderstaande tabel is het resultaat hiervan weergegeven. PD staat hier voor Probleem Decompositie en AL voor Algoritmische vaardigheid. De antwoorden op deze vragen zijn gebruikt om de vorderingen per leerlingen te bekijken. De resultaten in onderstaande tabel zeggen alleen maar iets over hoe goed iedere vraag gemaakt is in het algemeen. Hieruit blijkt dat Schoolfeestje en vooral Strafwerk de moeilijkere opdrachten waren voor de leerlingen. De inschatting van de makers van de vragen was echter dat Dobbelen moeilijker is dan de rest gevolgd door MP3 en Strafwerk. De rest van de vragen zou ‘easy’ moeten zijn voor de leerlingen.

Bijeenkomst	Vraag	Categorie	Aantal goed	Aantal fout	Totaal
2	Hoekjes knippen	AL	34 D	1 C	35
	Tangram	PD	34 C	1 A	35
3	Dobbelen	AL	24 C	4 B	28
	Schoolfeestje	PD	18 A	10 (1B, 9C)	28
4	MP3	AL	29 C	2 (A, D)	31
	Strafwerk	AL	12 C	19 (9A, 2B, 8D)	31

Tabel 1: Resultaat bevervragen

Regenprobleem

De helft van de leerlingen die deze opdracht heeft gemaakt, heeft een goede oplossing bedacht. 13 van de 26 leerlingen hebben het juiste antwoord gegeven. Code die fouten bevatte die geen betrekking hadden op de uitkomst zijn niet fout gerekend. Dit zijn fouten die bijvoorbeeld syntax errors veroorzaken.
De volgende fouten zijn gemaakt door de mensen die het niet goed hadden:

Aantal leerlingen	Soort fout
5	In het geval dat je meteen 999 zou invullen is er een /0 fout, door verkeerde initialisatie
2	Het eerst ingevoerde getal wordt niet meegenomen voor de berekening van het gemiddelde omdat er één keer te veel naar de input wordt gekeken
2	999 is meegenomen in de berekening van het gemiddelde
2	Er wordt gekeken naar getallen <999
1	Het totaal wordt door 1 te veel gedeeld
1	Totaal fout

Tabel 5: Fouten in oplossing regenprobleem

Een voorbeeld van een uitwerking waarin de meest gemaakte fout naar voren komt:

```
getal  = 0
teller = 0
while True:
     n = int(input())
     if n != 999:
         getal = getal + n
         teller = teller + 1
     else:
         print(getal/teller)
         break
```

Als de gebruiker van dit programma meteen 999 invoert, dan ontstaat er een /0 error. De waardes getal en teller zijn namelijk geïnitialiseerd op 0, dus er wordt door de code 0/0 geprobeerd in het print statement.

Exitvragen

Theorievragen

Over het algemeen gaven de leerlingen iedere week met 'Ja' aan de begrippen te snappen en toe te kunnen passen die ze geleerd hadden voor die week. Bij de thema's arrays en loops gaf een deel van de leerlingen aan dat ze 'een idee hebben'. Het was, zoals eerder benoemd, ook terug te zien in de uitwerkingen van de opdrachten dat niet iedereen even goed weet hoe en waarvoor loops en arrays gebruikt worden.

Evaluatie opdracht van de week

In de methodesectie staat beschreven dat de leerlingen iedere bijeenkomst hebben aangegeven wat ze van de opdracht vonden en of ze de opdrachten af hadden. De opdrachten van bijeenkomst 1 en 5 bleken makkelijk/goed te doen en daarnaast ook af te zijn. De data laat zien dat de leerlingen gedurende de cursus de opdrachten steeds moeilijker vonden en ook minder vaak af hadden. Hierbij lopen de ervaringen van de leerlingen behoorlijk uiteen. De ene heeft het niet af en vond het moeilijk en de ander vond het makkelijk en heeft het af, waarbij het over dezelfde opdracht gaat.

Analyse

De exitvragen lieten zien dat de meeste leerlingen de theorie van de cursus hebben geleerd en voor een groot deel ook hebben begrepen. Echter komt dit niet altijd overeen met de praktijk. De opnames, foutenanalyse en de vragen over codefragmenten laten zien dat het coderen zelf nog niet goed gaat.

Vragen codefragmenten

In de exitvragen van bijeenkomst 3 werd een vraag uit onderzoek van Lister (2016) gesteld. Hierbij was het de vraag wat het doel is van variabele c in het gegeven stukje code.

Op één iemand na (die een vraagteken heeft ingevuld) heeft iedereen bij bijeenkomst 3 de exitvragen ingevuld (24 leerlingen). 10 leerlingen leggen uit wat er met waarde c gebeurd, namelijk 'in c wordt de waarde van a opgeslagen'. De overige leerlingen hebben het doel van variabele c in het stukje code gegeven. Er waren onder die 13 leerlingen twee verschillende soorten antwoorden:

- De beginwaarde wordt opgeslagen zodat b a kan worden.
- c is een temporary value/ tijdelijke waarde/temp.

Het goed beantwoorden van deze vraag zou kunnen wijzen op een concreet operationeel niveau van programmeren volgens de terminologie van Lister (2016).

In de exitvragen van bijeenkomst 4 zaten drie vragen van Lister (2004), zie de methodesectie en de bijlage voor de vragen. De resultaten hiervan zijn weergegeven in de tabellen hieronder. Vraag 1 is het best gemaakt, daarna vraag 2 en daarna vraag 3. De meesten hadden vraag 1 en 2 of 1 en 3 goed.

Alles goed	Vraag 1 goed	Vraag 2 goed	Vraag 3 goed	1 & 2 goed	1 & 3 goed	2 & 3 goed	Alles fout	Totaal
4	4	2	1	7	6	0	5	29

Tabel 6: Resultaat vragen over codefragmenten

Vraag	Correct	Totaal aantal antwoorden
1	72%	29
2	46%	29
3	38%	29

Tabel 7: Percentage goed vragen over codefragmenten

Codekwaliteit

De ingeleverde opdrachten van zes studenten zijn gecodeerd aan de hand van criteria uit de rubric van Stegeman et al. (2016). In de analyse van de opdrachten komen een aantal dingen duidelijk naar voren:

Er wordt gedurende de cursus steeds vaker gebruik gemaakt van betekenisvolle namen. Dat betekent dat in plaats van de naam 'a' of 'opdracht 1' een variabele of methode een naam krijgt die de functie of het doel van de variabele of methode beschrijft. Ook wordt er gebruikt gemaakt van decompositie zodra de opdrachten langer worden. De flow is iets wat bij langere opdrachten niet goed loopt. Er wordt naarmate de programma's ingewikkelder worden en de code langer wordt, meer aandacht besteed aan lay-out en format. Comments worden nauwelijks gebruikt. De meeste fouten die gemaakt worden gaan over algoritmische vaardigheid. Loops en arrays worden niet of niet op de juiste manier ingezet waar nodig om een goede uitwerkingen van de opdrachten te kunnen realiseren.

Opnames

Bij het bekijken van de opnames is, zoals in de methode uitgelegd, gelet op verschillende indicatoren. De vijf stappen van McCracken et al. (2001) kwamen niet allemaal naar voren in de opnames. Vooral stap 3 tot en met 5 werden uitgevoerd door de leerlingen bij deze opdracht. De leerlingen bedachten (sub-) oplossingen voor de problemen die in de opdracht staan beschreven. Evalueren en itereren

werd continue toegepast door alle tweetallen. Het evalueren gebeurde door middel van overleg, code uitvoeren in PyCharm en traceren.
Het doorlopen van de laatste drie stappen van het stappenplan van McCracken et al. laat zien dat de leerlingen over probleemoplossend vermogen beschikken.
Voor het traceren gebruikte één stel meerdere keren de online visualisatie applicatie die bij de online Pythoncursus hoort. Deze applicatie maakt het mogelijk om stap voor stap door een stukje code heen te lopen. De andere stellen traceerden hun eigen code ook, maar vooral met de hand op het scherm.
Om een oplossing te vinden voor de vragen in de opdracht die ze moesten maken, werd er ook door alle tweetallen naar de theorie online gekeken. Het gevorderde tweetal dacht zelfs terug aan een opdracht die ze eerder hadden gemaakt bij een andere programmeercursus. Ze zetten hun kennis van de oplossing van een ander probleem in en probeerden van die oude oplossingen een nieuwe te maken die toepasbaar moest zijn op het nieuwe probleem.

In de opnames was te zien dat er veel verschil bestond tussen het niveau van de leerlingen. Het gebruik maken van een enkele keer traceren, steeds met dezelfde input, zou volgens de terminologie van Lister (2016) duiden op een pre-operationele fase van het programmeren. Echter lieten andere stellen zien dat ze ook naar een klasse van input konden kijken en daarmee konden variëren bij het traceren. Dit zou er dan op duiden dat deze leerlingen zich in de concreet operationele fase van het programmeren bevonden op het moment van opname. Het laatste stel was een uitzondering op de rest van alle leerlingen. Ze hadden, anders dan alle andere leerlingen, geen moeite met syntax, wisten precies welke loops ze waarvoor konden gebruiken en wisten de betekenis van verschillende errors. Dat zou duiden op een formeel operationeel niveau.

Individuele ontwikkeling

In de tabellen hieronder is weergegeven hoe de 6 geselecteerde leerlingen de opdrachten tijdens de cursus volbracht hebben. De eerste tabel is een weergave van de bevervragen, vragen over codefragmenten en het regenprobleem. De resultaten van de analyse van de codekwaliteit aan de hand van de ingeleverde uitwerkingen wordt daarna besproken. De tweede tabel laat zien hoe de leerlingen de exitvragen hebben ingevuld.

Opdrachten

Leerling	Bevervragen	Vraag over het doel van c	Vragen codefragmenten	Regenprobleem
A	Alles goed	Juist	Vraag 2 fout	Opgelost
B	Alles goed	Juist	Vraag 2 fout	Opgelost

C	Schoolfeestje fout	Juist	Vraag 2 en 3 fout	Helemaal fout:
				input(int(a))
				while a => 999:
				b + a= b
				else b/998 = c:
				print(c)

D	Schoolfeestje fout	Onjuist	Vraag 2 en 3 fout	Opgelost
E	Schoolfeestje en Strafwerk fout	Juist	Vraag 1 en 2 fout	Opgelost
F	Schoolfeestje en Strafwerk fout	Onjuist	Vraag 2 fout	Opgelost

Tabel 2: Resultaat bevervragen, vragen codefragmenten, regenprobleem per leerling

Analyse codekwaliteit

Leerling A

Alles is ingeleverd van bijeenkomst 2 tot en met 5. Deze leerling zat al sinds de eerste opdrachten bij alle criteria volgens de rubric op niveau 3. Alleen de gegeven namen waren vaak abstract en daarom was het niveau van namen geven 2 bij deze leerling. Wel had A alle opdrachten af en alle opdrachten werkten ook.

Leerling B

Bij bijeenkomst 2 zat leerling B bij alle criteria op niveau 2, ook de werking van het programma kreeg een 2. De keer daarna had B de opdrachten die ingeleverd zijn allemaal helemaal goed (niveau 3). De uitwerking van de laatste opdracht van bijeenkomst 3 bevatte te weinig code om te kunnen beoordelen. Ook bij bijeenkomst 4 was alles goed behalve de laatste opdracht. Die laatste opdracht werkte nog niet helemaal (niveau 2). De laatste opdracht voor het diploma was volledig op niveau 2 en werkte.

Leerling C

Bij leerling C zijn alleen de uitwerkingen van bijeenkomst 3 ontvangen. Van deze bijeenkomst is alleen de eerste opdracht gemaakt. De uitwerkingen werken wel allemaal, maar alles is verder niveau 1.

Leerling D

De foutenanalyse kon alleen maar gemaakt worden na bijeenkomst 3 en 5. De rest van de uitwerkingen zijn niet verkregen. De uitwerkingen van bijeenkomst 3 bevatten nauwelijks code en dat was te weinig om te beoordelen. Dit zou betekenen dat leerling D er helemaal niet uitgekomen is met deze opdrachten. Na bijeenkomst 5 is wel een stuk code ingeleverd. Dit was een werkend programma. Er is in die laatste opdracht gebruikt gemaakt van duidelijke namen, goede lay-out en format en duidelijke scheiding van code in methodes.

Leerling E

De uitwerkingen van bijeenkomst 2 werkten bijna of helemaal. Echter scoorde de leerling op alle andere criteria niveau 1 en 2. De opdrachten vanaf bijeenkomst 3 waren allemaal goed gemaakt, alle criteria zaten op niveau 3. Echter viel op dat bij een sorteeropdracht van bijeenkomst 4 onnodig en foutief gebruikt gemaakt is van while-loops achter elkaar.

Leerling F

Net als leerling E, zat leerling F bij alle criteria op niveau 1 en soms op 2 bij de eerste opdrachten. Ook viel hier het onnodig/foutief vaak gebruik van while-loops op. Bij bijeenkomst 3 was er voornamelijk sprake van niveau 3 en alle uitwerkingen werkten. De uitwerkingen van de laatste twee bijeenkomst zorgden ervoor dat leerling F eindigde op niveau 3 van alle criteria.

Exitvragen

Leerling	Bijeenkomst 1	Bijeenkomst 2	Bijeenkomst 3	Bijeenkomst 4	Bijeenkomst 5
A	Opdracht af Makkelijk Alles begrepen	Opdracht af Makkelijk Alles begrepen	Opdracht af Makkelijk Alles begrepen	Bijna af Makkelijk Alles begrepen	Opdracht af Makkelijk Alles begrepen
B	Opdracht af Makkelijk Alles begrepen	Bijna af Goed te maken Math functies: idee	Bijna af Goed te maken Alles begrepen	Bijna af Goed te maken Alles begrepen	Opdracht af Goed te maken Alles begrepen
C	Opdrachten af Goed te maken Alles begrepen	Bijna af Goed te maken Break/continue: idee	Bijna af Goed te maken 'A or B' en 'not A': idee	Bijna af Moeilijk Arrays en methode: idee	Opdracht af Goed te maken Alles begrepen
D	Opdrachten af Goed te maken Alles begrepen	Bijna af Goed te maken While/for: idee Break/continue: nee	Bijna af Goed te maken Alles begrepen	Bijna af Moeilijk Arrays en methode: idee	Bijna af Goed te maken Recursie en 'is operator': idee
E	Opdrachten af Goed te maken Alles begrepen	Bijna af Moeilijk Alles begrepen	Niet af Goed te maken Alles begrepen	Niet af Goed te maken Methodes: idee	Bijna af Makkelijke Alles begrepen
F	Opdrachten af Goed te maken Alles begrepen	Opdracht af Goed te maken Gebruikersinput en break/continue: idee	Niet af Goed te maken Alles begrepen	Niet af Moeilijk Alles begrepen	Bijna af Makkelijke Efficiënte programma's: idee

Tabel 3: Resultaat ervaring per leerling per bijeenkomst

Aan de hand van de data kan globaal gezegd worden dat de leerlingen wel vorderingen maken tijdens deze cursus, maar niet zo veel dat ze aan het einde alle opdrachten zonder hulp zouden kunnen maken.

Achtergrondfactoren

Uit de analyse van de instaptest en de individuele ontwikkelingen van de leerlingen komt naar voren dat er een verband zou kunnen zijn tussen de achtergrond van de leerling en de ontwikkeling die de leerling laat zien. De leerlingen die hebben aangegeven dat ze al veel ervaring hadden, hebben betere resultaten behaald dan de leerlingen zonder ervaring.
Ook lijkt het uit de data naar voren te komen dat de keuze voor een Natuur & Techniek of Natuur & Gezondheid profiel, ervaring met programmeren en plezier in het volgen van bètavakken een voordeel is als je wil leren programmeren. De leerlingen met een Cultuur & Maatschappij of Economie & Maatschappij profiel hebben meer moeite met de opdrachten dan de anderen.

Conclusie

De volgende hoofd- en deelvragen zullen beantwoord worden in deze conclusie:

Deelvragen

- Welke vorderingen maken de leerlingen tijdens deze cursus?
- Wat is de relatie tussen de achtergrond van de leerlingen en de vorderingen die ze maken?

Hoofdvraag

Hoe effectief is de cursus programmeren voor beginners in Python?

- *Welke vorderingen maken de leerlingen tijdens deze cursus?*

Uit de analyse van de codekwaliteit van de uitwerkingen van de leerlingen is duidelijk gebleken dat de leerlingen kwalitatief betere code konden schrijven aan het einde van de cursus dan aan het begin van de cursus. Alle andere tests en opnames hebben vooral inzicht gegeven in het niveau van de vaardigheden van de leerlingen op het moment van onderzoeken. De meeste leerlingen zaten aan het einde van de cursus volgens de indeling van Lister (2016) in de concreet operationele fase. Dit is de fase vanaf wanneer de leerling in staat zou moeten kunnen zijn om te gaan beginnen met programmeren. Echter zat een aantal leerlingen ook nog in de pre-operationele fase en dat zou er dus op duiden dat zij nog niet genoeg basisvaardigheden hebben om te kunnen gaan programmeren. Uitzondering op de rest zijn leerlingen A en B, die volgens de data al in de formeel operationele fase zaten tijdens de cursus.

- *Wat is de relatie tussen de achtergrond van de leerlingen en de vorderingen die ze maken?*

Over een relatie tussen de achtergrond van de leerlingen en de resultaten die ze behaald hebben kan alleen maar gespeculeerd worden op basis van de data. Het verschil in niveau en in vorderingen tussen de leerlingen zou te maken kunnen hebben met verschillende factoren. Het verschil in profielkeuze, interesses en ervaring zijn daar voorbeelden van. Sommige leerlingen hebben meer moeite met de opdrachten dan andere leerlingen. In dit onderzoek is naar voren gekomen dat leerlingen met een Natuur & Gezondheid of Natuur & Techniek profiel betere resultaten behaalden dan leerlingen die een Cultuur & Maatschappij of Economie & Maatschappij profiel hebben gekozen. Ook lijkt uit de data naar voren te komen dan programmeerervaring (in verschillende talen) ook een voordeel is bij het volgen van deze cursus.

- *Hoe effectief is de cursus programmeren voor beginners in Python?*

De cursus geeft de leerlingen de mogelijkheid om de basis te leren van programmeren in Python. Hoeveel de leerlingen precies geleerd hebben kan op basis van de data niet gezegd worden. Wel kan gezegd worden dat alle leerlingen een bepaald eindniveau hebben behaald wat aansloot bij de cursus. Ze hebben namelijk allemaal de laatste opdracht goed genoeg kunnen maken om een certificaat in ontvangst te mogen nemen. Echter bleek tijdens het onderzoek dat er veel leerlingen waren die onderdelen van de theorie nog niet genoeg begrepen om goed functionerende code te kunnen schrijven. Een voorbeeld hiervan is het foutieve gebruik van arrays en loops.
Hoe effectief deze cursus programmeren voor beginners in Python exact is, kan met de verzamelde data niet worden gezegd. Uit het onderzoek komt wel naar voren dat de leerlingen die de cursus hebben gevolgd beter zijn geworden in programmeren.

Discussie

De vragen uit onderzoek van Lister (2004, 2016) zijn in dit onderzoek anders beantwoord dan in eerder onderzoek. Vraag 1 werd het best gemaakt, waar uit onderzoek van Lister blijkt dat vraag 3 juist het makkelijkst zou zijn. Vraag 3 werd juist het minst goed gemaakt door de deelnemers van de Pythoncursus.
Het regenprobleem werd, niet geheel in lijn met de theorie (38% goed na één beginnerscursus programmeren), door de helft van de leerlingen opgelost. De oplossing van dit makkelijk lijkende probleem bevat toch een aantal punten die leerlingen hebben verward zoals beschreven in de resultaten.

De analyse van de criteria voor codekwaliteit liet zien dat de codekwaliteit van de uitwerkingen van de leerlingen omhoog ging tijdens de cursus. Er werd naar het einde van de cursus toe steeds meer gebruik gemaakt van zinvolle namen en decompositie. Een mogelijke verklaring hiervoor zou kunnen zijn dat er ook in de opdrachten die de leerlingen krijgen steeds meer aanbevelingen stonden voor bepaalde design opties en voor het gebruik van zinvolle namen. Maar ook de lay-out en het format van de uitwerkingen van de leerlingen werd beter. Dit zou kunnen volgen uit de complexiteit van de opdrachten. Hoe meer complex de oplossing van een opdracht is, hoe meer de leerlingen misschien al automatisch letten op lay-out, format en decompositie. Een verklaring hiervoor zou kunnen zijn dat de leerlingen door het gebruik van deze principes een beter overzicht kunnen houden over de zelfgeschreven code.

In de opnames waren duidelijk de laatste drie stappen van McCracken et al. (2001) te zien. Naast het oplossen van sub-problemen, het samenvoegen daarvan en het evalueren en itereren behoren ook de stappen abstraheren en sub-problemen definiëren tot het stappenplan. Echter werden deze twee stappen vaak al gedaan voor de leerlingen door de manier waarop de vragen werden gesteld. Dit zou kunnen verklaren waarom deze stappen niet terug te zien zijn in de opnames.

De opdrachten die uit onderzoek van Lister (2004) kwamen, waren oorspronkelijk in een andere taal. De omzetting naar Python heeft misschien gezorgd voor een andere uitkomst dan in het oorspronkelijke onderzoek.

Een aantal aspecten van dit onderzoek zorgt mogelijk voor een ander uitkomst dan in andere onderzoeken. Zo is in dit onderzoek de belasting van de leerlingen met extra taken geprobeerd beperkt te houden. Hierdoor is de manier waarop gemonitord is, niet erg uitgebreid. Verder bevatte het onderzoek relatief weinig proefpersonen. De leerlingen kregen daarnaast geen feedback op hun gemaakte opdrachten. Ze hebben dus niet van eventuele feedback kunnen leren. Hierdoor bleven leerlingen mogelijk dezelfde fouten maken in volgende bijeenkomsten wat de uitwerkingen van de opdrachten niet bevorderde.

De omstandigheden in dit onderzoek waren niet hetzelfde als in andere onderzoeken. Door de mogelijkheid tot samenwerken en het missen van een beoordeling kan het bijvoorbeeld zijn dat motivatie anders is dan bij anderen die leren programmeren. Dit zou kunnen zorgen voor andere resultaten. Een voorbeeld hiervan is het regenprobleem. De leerlingen in dit onderzoek hebben de vraag beter beantwoord dan in het onderzoek van Soloway & Bonar (1982). Dit zou een gevolg kunnen zijn van het verschil in beschikbare middelen. In dit onderzoek werd het probleem opgelost op de computer waarop de deelnemers ook PyCharm konden openen om hun code te testen. Ook al was expliciet gevraagd dit niet te doen.

Omdat dit onderzoek een casestudy is, is het lastig om aan de hand van dit onderzoek te generaliseren. In de twee alinea's hierboven staat ook dat door bepaalde aspecten van de case, de resultaten niet altijd overeenkomen met andere, grotere, onderzoeken. Desondanks zijn de

bevindingen in dit onderzoek zijn interessant voor de Pythoncursus. Op basis van dit onderzoek kunnen namelijk aanbevelingen worden gedaan voor de cursus:

- Voor veel leerlingen was de functie van verschillende soorten loops niet volledig duidelijk. Daarom zou er de volgende keer meer aandacht besteed kunnen worden aan dit onderwerp in de presentaties tijdens de bijeenkomsten.
- Er zouden de volgende keer ook meer kleine opdrachten toegevoegd kunnen worden om het begrip van de werking van verschillende loops te vergroten bij de leerlingen.
- Een vorm van feedback in de cursus zou leerlingen kunnen helpen om sneller te leren wat wel en niet werkt bij het oplossen van een probleem en specifiek in Python.

Naast een aantal aanbevelingen voor de Pythoncursus, zijn er ook nog suggesties voor verder onderzoek:

- Er zou meer onderzoek gedaan kunnen worden aan de hand van sessie waarin leerlingen worden gevraagd hardop te denken tijdens het programmeren. Met meer wordt bedoeld dat er meerdere sessies met dezelfde persoon en ook met meerdere personen onderzoek gedaan kan worden. Dan krijg je (hopelijk) meer inzicht in de gedachte achter bepaalde keuzes van leerlingen tijdens het programmeren.
- Onderzoek de fases van Piaget/Lister (2016) met de vragen van Lister die beschreven zijn in het artikel Toward a Developmental Epistemology of Computer Programming. De uitkomst daarvan kan een interessante aanvulling zijn op de theorie. Er kan dan bepaald worden dat sommige onderwerpen of het leren van bepaalde vaardigheden meer aandacht behoeft dan andere.

Literatuur

Barendsen, E., Mannila, L., Demo, B., Grgurina, N., Izu, C., Mirolo, C., ... & Stupurienė, G. (2015, July). Concepts in K-9 Computer Science Education. In *Proceedings of the 2015 ITiCSE on Working Reports* (pp. 85-116). ACM.

Beverwedstrijd. (z.d.). Geraadpleegd op 3 januari 2017, van http://www.beverwedstrijd.nl

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of educational psychology*, *76*(6), 1051-1058.

Kahn, K. (1999). A Computer Game To Teach Programming.

Lister, R. (2016, October). Toward a Developmental Epistemology of Computer Programming. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education* (pp. 5-16). ACM.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., ... & Simon, B. (2004, June). A multi-national study of reading and tracing skills in novice programmers. In *ACM SIGCSE Bulletin* (Vol. 36, No. 4, pp. 119-150). ACM.

Matthíasdóttir, Á. (2006, June). How to teach programming languages to novice students? Lecturing or not. In *International Conference on Computer Systems and Technologies-CompSysTech* (Vol. 6, pp. 15-16).

McCracken et al., M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B. D., ... & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, *33*(4), 125-180.

Mulholland, P., & Eisenstadt, M. (1998). Using software to teach computer programming: Past, present and future. Software Visualization: Programming as a Multimedia Experience, 399-408.

Soloway, E. & Bonar, G. (1982). What do novices know about programming?
, 27-54.

Soloway, E., Bonar, J., & Ehrlich, K. (1983). Cognitive strategies and looping constructs: An empirical study. *Communications of the ACM*, *26*(11), 853-860.

Stegeman, M., Barendsen, E., & Smetsers, S. (2014, November). Towards an empirically validated model for assessment of code quality. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research* (pp. 99-108). ACM.

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2016). Computer science in K-12 school curricula of the 2lst century: Why, what and when?. *Education and Information Technologies*, 1-24.

Appendix

Hier zijn alle vragen te vinden die gesteld zijn aan de leerlingen tijdens dit onderzoek.
Een overzicht van de verschillende vragenlijsten die hierna volgen:

- Instaptest Pythoncursus
- Vragen bijeenkomst 2
- Vragen bijeenkomst 3
- Vragen bijeenkomst 4
- Vragen bijeenkomst 5
- Exitvragen bijeenkomst 1
- Exitvragen bijeenkomst 2
- Exitvragen bijeenkomst 3
- Exitvragen bijeenkomst 4
- Exitvragen bijeenkomst 5

Instaptest Pythoncursus

Hoi, wat leuk dat je deelneemt aan deze cursus! Voordat je aan de cursus begint, wil ik je graag wat beter leren kennen. Ik ben benieuwd naar de kennis en ervaring die je nu hebt met betrekking tot programmeren. Omdat ik je gegevens wil gebruiken voor onderzoek, wil ik je vragen of je zo eerlijk mogelijk antwoord wil geven op onderstaande vragen. Je zult niet beoordeeld worden op je antwoorden en je antwoorden zullen ook niet openbaar gemaakt worden. Het is geen lange vragenlijst. Het is heel fijn dat je de vragenlijst in wil vullen.

1. Wat is je naam?
2. Wat is je profielkeuze?

Markeer slechts één ovaal.

⬭ Natuur en Techniek
⬭ Natuur en Gezondheid
⬭ Economie en Maatschappij
Cultuur en Maatschappij

3. Welke 3 vakken vind je het leukst op school?
4. Volg je nu het vak Informatica op school?

Markeer slechts één ovaal.

⬭ Ja
⬭ Nee

5. Hoeveel ervaring heb je met programmeren?

Markeer slechts één ovaal.

⬭ Geen ervaring
⬭ Een beetje ervaring
⬭ Veel ervaring

6. Met welke programmeertaal heb je ervaring?
7. Welke andere programmeertalen ken je?

Dan nu een paar theorievragen over programmeren. Het verschilt per persoon wat je weet. Het is dus helemaal niet erg als je de antwoorden niet weet.

8. Weet je wat er gebeurt als je print(x) uitvoert?

Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee

9. Weet je wat een variabele is?

Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee

10. Weet je wat errors zijn en hoe ze kunnen ontstaan?

Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee

11. Weet je wat een functie is?

Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee

12. Weet je het verschil tussen een string en een integer?

Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee

13. Weet je wat het verschil is tussen if en while?

Markeer slechts één ovaal.

⬭ Ja

⬭ Ik heb wel een idee
⬭ Nee
14. Weet je wat een boolean en/of booleaanse vergelijking is?
Markeer slechts één ovaal.
⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee
15. Weet je wat een loop is (bijvoorbeeld een while loop of een for loop)?
Markeer slechts één ovaal.
⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee
16. Weet je wat het verschil is tussen while en for?
Markeer slechts één ovaal.
⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee
17. Weet je wat bedoeld wordt met een if-then-else constructie?
Markeer slechts één ovaal.
⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee
18. Weet je wat een array is?
Markeer slechts één ovaal.
⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee
19. Weet je wat recursie is?
Markeer slechts één ovaal.
⬭ Ja
⬭ Ik heb wel een idee
⬭ Nee
20. Heb je zelf wel een recursie gebruikt?
Markeer slechts één ovaal.
⬭ Ja
⬭ Misschien
⬭ Nee
21. Heb je deelgenomen aan de Beverwedstrijd?
Markeer slechts één ovaal.
⬭ Ja
⬭ Nee
22. Wanneer was dat?
Markeer slechts één ovaal.
⬭ 2016
⬭ 2015
⬭ 2014
⬭ 2013
⬭ 2012
⬭ 2011
⬭ 2010
⬭ Niet van toepassing
23. Wat was je score bij de Bevertest?

Dat was de laatste vraag. Bedankt voor het invullen van de vragen! Veel plezier en succes met de cursus!

Vragen bijeenkomst 2

Om te testen hoe jullie vaardigheden vorderen, zou ik jullie willen vragen onderstaande opgaven te maken. Het zijn twee oude vragen van de Beverwedstrijd. Succes!

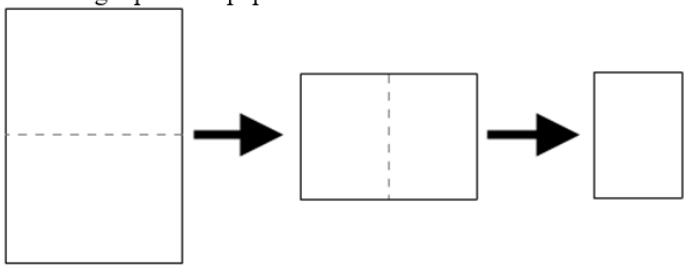
1. Welk van de onderstaande uitgevouwen blaadjes kan niet gemaakt worden als het papier 2 keer wordt dubbelgevouwen en er vervolgens een hoekje schuin wordt afgeknipt?

Markeer slechts één ovaal.

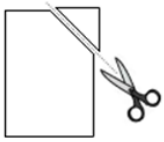
- ⬭ A
- ⬭ B
- ⬭ C
- ⬭ D

Cutting the corner

A rectangle piece of paper is folded twice so that the result is a rectangle again.



In this rectangular one of the corner is cut off.
Then the paper is folded out.



Which sample cannot be obtained in this way?

2. Welke van de onderstaande figuren kan niet gemaakt worden met de blokjes die je hebt zoals weergegeven in de afbeelding?

Markeer slechts één ovaal.

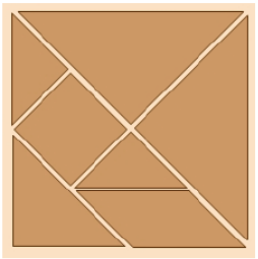
- ⬭ A
- ⬭ B
- ⬭ C
- ⬭ D

Title:

TANGRAM

Body:

Beaver has learned in the workshop how to construct tangrams. The square is first divided into seven pieces (as shown in the figure).



Then he has started to compile a variety of forms. Which of the forms in the figures Beaver was not able to construct?

a) 

b) 

c) 

d) 

Vragen bijeenkomst 3

Om te testen hoe jullie vaardigheden vorderen, zou ik jullie willen vragen onderstaande opgaven te maken. Het zijn twee oude vragen van de Beverwedstrijd. Vul eerst nog even je naam in. Succes!
1. Naam

Schoolfeestje

De Beverschool geeft een schoolfeest. Leerlingen die ouder zijn dan 18 mogen alcohol drinken en leerlingen die jonger zijn dan 18 mogen dat niet. Om het makkelijk te maken voor de barmannen, krijgen de leerlingen die ouder zijn dan 18 een polsbandje bij de ingang.
De school maakt gebruik van een database voor het uitdelen van de bandjes. Alle leerlingen hebben een school-ID-kaart en een leerling nummer. Op de ID-kaart staat een foto van de leerling en een barcode die aan het unieke leerling nummer van bijbehorende leerling is gekoppeld.
Een docent controleert de foto's en scant de barcodes van de pasjes bij de ingang. De database beslist dan over een leerling wel of geen bandje krijgt.
Welke informatie is niet noodzakelijk voor een goed werkende database?
Antwoorden:

a) De achternaam van de leerlingen
b) Het leerling nummer van de leerlingen
c) Een vakje per student dat kan worden aangevinkt als een leerling al een bandje heeft gekregen
d) De geboortedatum van de leerlingen

2. Lees bovenstaande vraag en selecteer een antwoord.
Markeer slechts één ovaal.

⬭ A
⬭ B
⬭ C
⬭ D

Dobbelen

Na school spelen de leerlingen buiten. Om ruzie te voorkomen over waar ze gaan spelen, gooien ze met een dobbelsteen met nummer 1 tot en met 6 erop.
De beslissing voor de plek waar ze gaan spelen wordt als volgt genomen:

```
IF de eerste worp groter is dan de tweede worp,
THEN ga in het bos spelen,
ELSE    IF de derde worp minder is dan de eerste worp,
            THEN ga bij de rivier spelen,
            ELSE ga op het sportveld spelen.
```

Welke volgorde van worpen heeft als uitkomst dat de leerlingen op het sportveld gaan spelen?
Antwoorden:

a) Eerste worp 6, tweede worp 6, derde worp 2
b) Eerste worp 5, tweede worp 3, derde worp 6
c) Eerste worp 3, tweede worp 4, derde worp 3
d) Eerste worp 2, tweede worp 4, derde worp 1

3. Lees bovenstaande vraag en selecteer een antwoord.
Markeer slechts één ovaal.

⬭ A
⬭ B
⬭ C
⬭ D

Vragen bijeenkomst 4

Om te testen hoe jullie vaardigheden vorderen, zou ik jullie willen vragen onderstaande opgaven te maken. Het zijn twee oude vragen van de Beverwedstrijd. Vul eerst nog even je naam in. Succes!
1. Naam

MP3 collectie

Billy Bever heeft een grote collectie MP3s. Namen van MP3s mogen maximaal 8 karakters lang zijn op zijn nieuwe MP3-speler. Daarom heeft Billy de volgende regels voor het kiezen van een gepaste bestandsnaam voor zijn MP3s:

- De eerste drie karakters van de bestandsnaam zijn de eerste drie karakters van de band.
- De volgende drie karakters van de bestandsnaam zijn de eerste drie karakters van het album.
- De laatste twee karakters zijn het nummer van het liedje op de CD.
- Als twee bestanden dezelfde naam zouden krijgen, wordt het zesde karakter één letter opgehoogd (A→B, B→C,…,Z→A) totdat er een naam ontstaat die nog vrij is.

De collectie bestaat op dit moment uit de volgende MP3s:

BEABOX01	BEABOX02	BEABOX03	BEABOX04	BEABOX05
BEABOY01	BEABOY02	BEABOZ01	BEABOZ02	BEABOA01
BEABOA02	BEABOA03	BEABOA04	BEABOA05	BEABOB01
BEABOC01	BEABOC02	BEABOC03	BEABOD01	BEABOD02

Het volgende liedje wil hij toevoegen aan de collectie:
Band: Beatles
Album: Boxed Set, CD 5
Track nr: 03

Wat wordt de bestandsnaam van dit liedje?

Antwoorden:

a) BEABOD03
b) BEABOD06
c) BEABOY03
d) BEABOB03

2. Lees bovenstaand vraag en selecteer een antwoord.
Markeer slechts één ovaal.

⬭ A
⬭ B
⬭ C
⬭ D

Strafwerk

Bever heeft strafwerk gekregen en moet de zin “Ik zal mijn staart niet meer aan de meester laten zien” 100 keer opschrijven. Hij is aan het schrijven in een tekstverwerker en heeft al één zin geschreven. Hij wil gebruik maken van kopiëren, klembord en plakken om het werk zo snel mogelijk gedaan te krijgen.

Stel het symbool A betekent dat Bever de toetsen Ctrl A heeft ingedrukt en alle geschreven tekst heeft gemarkeerd. Symbool C betekent dat hij Ctrl C heeft ingedrukt en het gemarkeerde stuk tekst

heeft gekopieerd naar het klembord. Symbool V betekent dat hij Ctrl V heeft ingedrukt en dat hij de tekst van het klembord heeft geplakt aan het einde van de tekst (zonder de tekst die er al stond te verwijderen).

ACV betekent dat Bever een zin markeert, kopieert en dan plakt achter de al geschreven zin. Nu heeft hij al twee zinnen in plaats van één.

Welke van de volgende reeksen zorgt ervoor dat er zo snel mogelijk minstens 100 zinnen geschreven staan?

Antwoorden:

a) ACVACVACVACVACVACV…
b) ACVVVVVVVVVVVVVVVV…
c) ACVVVACVVVACVVVACVVV…
d) ACVVACVVACVVACVVACVV…

3. Lees bovenstaande vraag en selecteer een antwoord.
Markeer slechts één ovaal.

⬭ A
⬭ B
⬭ C
⬭ D

Vragen bijeenkomst 5

Om te testen hoe jullie vaardigheden vorderen, zou ik jullie willen vragen onderstaande opgave te maken. Dit keer is het maar één opdracht. Bedankt voor jullie medewerking de afgelopen weken! Vul eerst nog even je naam in. Succes!

1. Naam
2. Schrijf een Python programma dat integers blijft inlezen (invoer door de gebruiker) totdat het getal 999 wordt ingelezen, print vervolgens het gemiddelde van de ingelezen getallen. In dit gemiddelde is het getal 999 niet meegenomen. Het is de bedoeling dat je het programma hieronder typt en niet eerst in PyCharm (of ergens anders) gaat testen.

Exitvragen bijeenkomst 1

Vul eerst je naam in en geef daarna per stelling aan welke antwoord het best bij jou past.

1. Naam
2. Ik weet nu wat print("Hello, World!") doet.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

3. Ik weet nu hoe ik een variabele maak en er een waarde in op kan slaan.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

4. Ik weet wat een syntax fout is.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

5. Ik weet wat een runtime error is.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

6. Ik weet hoe ik een functie schrijf.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

7. Ik weet hoe ik commentaar moet schrijven in de code.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

8. Ik heb de opdrachten van vandaag af.
Markeer slechts één ovaal.

⬭ Ja
⬭ Bijna
⬭ Nee

9. Ik vond de opdrachten:
Markeer slechts één ovaal.

⬭ Makkelijk
⬭ Goed te maken met hulp van de studentassistenten en de theorie
⬭ Moeilijk

Exitvragen bijeenkomst 2

Vul eerst je naam in en geef daarna per stelling aan welke antwoord het best bij jou past.

1. Naam
2. Ik weet nu wat een String is en waarvoor ik die kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

3. Ik weet nu wat een Integer is en waarvoor ik die kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

4. Ik weet nu wat een float is en waarvoor ik die kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

5. Ik weet hoe ik input vraag van een gebruiker en hoe ik dat kan gebruiken in een programma.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

6. Ik weet hoe ik een ifstatement schrijf en waarvoor ik het kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

7. Ik weet wat een boolean is en hoe ik een booleaanse vergelijking gebruik.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

8. Ik weet hoe ik math functies gebruik in een programma.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

9. Ik weet het verschil tussen een while- en een for-loop.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

10. Ik weet waarvoor ik een break en een continue statement kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

11. Ik heb de opdrachten van vandaag af.
Markeer slechts één ovaal.

⬭ Ja
⬭ Bijna
⬭ Nee

12. Ik vond de opdrachten:
Markeer slechts één ovaal.

⬭ Makkelijk
⬭ Goed te maken met hulp van de studentassistenten en de theorie
⬭ Moeilijk

Exitvragen bijeenkomst 3

Vul eerst je naam in en geef daarna per stelling aan welke antwoord het best bij jou past.

1. Naam
2. Ik weet hoe ik een if-else statement schrijf en waarvoor ik het kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

3. Ik weet wat de booleaanse expressie 'A and B' betekent en waarvoor ik die kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

4. Ik weet wat de booleaanse expressie 'A or B' betekent en waarvoor ik die kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

5. Ik weet wat de booleaanse expressie 'not A' betekent en waarvoor ik die kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

6. Ik weet hoe ik een functie schrijf en waarvoor ik die kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

7. Ik weet hoe ik een functie moet aanroepen en waar ik dat kan doen.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

Het doel van de volgende drie regels code is om de waardes in variabele a en b om te wisselen:

```
c = a
a = b
b = c
```

8. Wat is het doel van variabele c in de code hierboven?
9. Ik heb de opdrachten van vandaag af.
Markeer slechts één ovaal.

⬭ Ja
⬭ Bijna
⬭ Nee

10. Ik vond de opdrachten:
Markeer slechts één ovaal.

⬭ Makkelijk
⬭ Goed te maken met hulp van de studentassistenten en de theorie
⬭ Moeilijk

Exitvragen bijeenkomst 4

Vul eerst je naam in en geef daarna per stelling aan welke antwoord het best bij jou past.

1. Naam
2. Ik weet wat een array is en waarvoor ik deze kan gebruiken.

Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

3. Ik weet hoe ik een lijst kan doorlopen met een loop.

Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

4. Ik weet wat een methode is en waarvoor ik die kan gebruiken.

Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

5.

Bekijk het volgende stukje code:

```
x = [2, 1, 4, 5, 7]
limit = 3
i = 0
sum = 0

while sum < limit and I < len(x) :
       i += 1
       sum += x[i]
```

Welke waarde heeft variabele i nadat deze code is uitgevoerd?

e) 0
f) 1
g) 2
h) 3

Markeer slechts één ovaal.

⬭ A
⬭ B
⬭ C
⬭ D

6.

Bekijk het volgende stukje code:

```
x = [1, 2, 3, 3, 3]
b = [False, False, False, False, False]

for i in range (len(b)):
   b[i] = False

for i in range (len(x)):
   b[x[i]] = True

count = o

for i in range (len(b)):
   if b[i]:
      count += 1
```

Welke waarde heeft “count” nadat deze code is uitgevoerd?

f) 1
g) 2
h) 3
i) 4
j) 5

Markeer slechts één ovaal.

⬭ A
⬭ B
⬭ C
⬭ D
⬭ E

7.
Bekijk het volgende stukje code:

```
x = [0, 1, 2, 3]
i = 0
j = len(x)-1

while i < j:
   temp = x[i]
   x[i] = x[j]
   x[j] = 2*temp
   i += 1
   j -=  1
```

Welke waarde heeft “x” nadat deze code is uitgevoerd?

```
f) [3, 2, 2, 0]
g) [0, 1, 2, 3]
h) [3, 2, 1, 0]
i) [0, 2, 4, 6]
j) [6, 4, 2, 0]
```

Markeer slechts één ovaal.

⬭ A
⬭ B
⬭ C
⬭ D
⬭ E

8. Ik heb de opdrachten van vandaag af.

Markeer slechts één ovaal.

⬭ Ja
⬭ Bijna
⬭ Nee

9. Ik vond de opdrachten:

Markeer slechts één ovaal.

⬭ Makkelijk
⬭ Goed te maken met hulp van de studentassistenten en de theorie
⬭ Moeilijk

Exitvragen bijeenkomst 5

Vul eerst je naam in en geef daarna per stelling aan welke antwoord het best bij jou past. Er zitten ook een aantal evaluatievragen van Pre-University College (vanuit de Radboud Universiteit) bij over de gehele cursus dit keer.

1. Naam
2. Ik weet wat recursie is.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

3. Ik weet hoe ik recursie toe moet passen.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

4. Ik weet waarvoor en hoe ik de 'is' operator kan gebruiken.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

5. Ik denk dat ik efficiënte programma's kan schrijven in Python.
Markeer slechts één ovaal.

⬭ Ja
⬭ Ik heb een idee
⬭ Nee

6. Ik heb de opdrachten van vandaag af.
Markeer slechts één ovaal.

⬭ Ja
⬭ Bijna
⬭ Nee

7. Ik vond de opdrachten:
Markeer slechts één ovaal.

⬭ Makkelijk
⬭ Goed te maken met hulp van de studentassistenten en de theorie
⬭ Moeilijk

8. Wat vond je van de eindopdracht?
9. Hoe beoordeel je de inhoud van de cursus? (1 = zeer slecht, 2 = onvoldoende, 3 = neutraal, 4 = voldoende, 5 = goed)
Markeer slechts één ovaal.

⬭1 ⬭2 ⬭3 ⬭4 ⬭5

10. Hoe beoordeel je de begeleiding door de studentassistenten? (1 = zeer slecht, 2 = onvoldoende, 3 = neutraal, 4 = voldoende, 5 = goed)
Markeer slechts één ovaal.

⬭1 ⬭2 ⬭3 ⬭4 ⬭5

11. In hoeverre ben je het eens met de stelling "Ik vond de cursus interessant." (1 = helemaal oneens, 5 = helemaal mee eens)
Markeer slechts één ovaal.

⬭1 ⬭2 ⬭3 ⬭4 ⬭5

12. In hoeverre ben je het eens met de stelling "Ik vond de stof moeilijk." (1 = helemaal oneens, 5 = helemaal mee eens)
Markeer slechts één ovaal.

⬭1 ⬭2 ⬭3 ⬭4 ⬭5