*Studenten*handleiding

Programming
TICT-V1PROG-15

Propedeuse HBO-ICT

Cursuseigenaar: Bart.vanEijkelenburg@hu.nl

Auteurs
Bart van Eijkelenburg (SIE)

Rory Sie (SIE)

Rienk van der Ploeg (SNE)

Arno Kamphuis (TI)

Esther van der Stappen (BIM)

Review
BAR: Edwin Bos (Alten PTS)

Student: Carlos van Rooijen (TI), Silas Herlaar

Maikel Martens (Experius)

Datum
29 augustus 2016

Versie: 2.0

Disclaimer: de beschrijvingen in de OER en Studiegids gaan voor de beschrijvingen in deze handleiding!

.

Inhoudsopgave

1 Opzet cursus

1.1 Inleiding

Voor je ligt de studiehandleiding van de cursus Programming (TICT-V1PROG-15) van het Instituut voor ICT. Deze cursus wordt in het eerste jaar van de opleiding HBO-ICT gegeven, en is onderdeel van de gemeenschappelijk kennisbasis die iedere HBO-ICT'er moet hebben.

Studenten krijgen inzicht in de basisconcepten en denkwijze van programmeren. De programmeertaal die hiervoor wordt gebruikt is Python. Op een hands-on manier maak je kennis met de syntax van Python, zoals het gebruik van variabelen, condities, loops, input/output en testing. Tegelijkertijd worden er vaardigheden aangeleerd die ten grondslag liggen aan programmeren en denken en handelen als HBO-ICT'er, zoals gestructureerd problemen oplossen, algoritmes en documentatie.

Hoewel ongeveer de helft van de studenten HBO-ICT al eens heeft geprogrammeerd, is deze cursus zo opgezet dat deze ook te volgen moet zijn zonder enige programmeerervaring. De cursus is daarmee ook toegankelijk voor studenten van andere opleidingen, zoals studenten in de Life Sciences and Chemistry (denk aan bio-informatica), Communicatie en Journalistiek (denk aan multimedia), of Gezondheid (denk aan e-Health).

1.2 Praktijkvoorbeeld

Om de relatie met de beroepspraktijk weer te geven, zullen we een viertal werksituaties schetsen, die respectievelijk staan voor de vier uitstroomprofielen die de opleiding HBO-ICT herbergt, te weten; Business IT & Management (BIM), System & Network Engineering (SNE), Technische Informatica (TI) en Software & Information Engineering (SIE).

Bedrijf HU4U wil graag een oplossing voor haar probleem: het bedrijf loopt, maar de tijd van idee naar product (*product-to-market time)* is erg lang, en tijd is geld. Om extra geld te besparen in deze financieel zware tijden, wil het bedrijf kijken of er toch niet bepaalde processen zijn die geoptimaliseerd en geautomatiseerd kunnen worden, zodat er veel tijd, en dus geld, gewonnen kan worden.

Werksituatie 1 (Business IT & Management (BIM))

De aanpak die hier geldt, is dat je als HBO-ICT'er eerst gaat kijken naar de huidige situatie. Hoe zit deze in elkaar? Hoe werken de processen? Wie voert welke taak uit, en wanneer? Je gaat kijken welke taken tot het primaire proces behoren, en welke taken niet. Vervolgens ga je kijken of er taken zijn die geautomatiseerd kunnen worden, als ze dat nog niet zijn. Je gaat deze taken verder uitsplitsen, zodat je precies weet wat er stap-voor-stap gebeurt in het proces, welke informatie (data) ermee gemoeid is, en wat er nodigen om dit te automatiseren.

Je maakt een reële tijdsinschatting van hoe lang je denkt dat de software engineers bezig zijn met het ontwikkelen van software die de automatisering voor haar rekening neemt. Dit kun je helder communiceren naar de klant, zodat deze weet wat hij of zij kan verwachten qua tijd en kosten die komen kijken bij een dergelijk optimalisatieproces. Je fungeert als een brug tussen de klant en het software-ontwikkelteam: Je weet wat de klant wilt, en vertaalt dit naar concrete acties en taken voor het ontwikkelteam (*requirements*). Hiervoor moet je weten hoe programmeertalen werken en hoe ontwikkelteams problemen doorgaans oplossen (bijv. de *divide-and-conquer*-strategie), zonder dat je de jarenlange ervaring met programmeren, de snelheid van programmeren, of de ins en outs van technologieën hebt die het programmeerproces soepeler laten verlopen.

.

Werksituatie 2 (Software & Information Engineering (SIE))

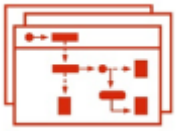
Nadat de bedrijfsprocessen geïdentificeerd zijn en de wensen en eisen van de klant bekend zijn geworden, gaat het ontwikkelteam aan de slag met de functionaliteit van het systeem. Dit houdt in dat de ontwikkelaars (ook wel software engineers genoemd) de requirements, vaak onderverdeeld in taken en subtaken en voorzien van prioriteiten, gaan uitwerken in programmacode.

Er zijn bepaalde gestructureerde manieren waarop de problemen of taken kunnen worden aangepakt, en die zijn de programmeur niet vreemd. De code wordt daar waar mogelijk is hergebruikt en netjes van commentaar voorzien, zodat andere ontwikkelaars in het team de programmacode kunnen begrijpen en desgewenst aanpassen. Als de ontwikkelaar tegen bepaalde problemen aanloopt bij het programmeren, bijvoorbeeld *bugs* of *errors*, dan weet de ontwikkelaar hoe deze horden genomen kunnen worden.

Werksituatie 3 (System & Network Engineering (SNE))

Bedrijf HU4U maakt zoals zoveel bedrijven gebruik van computers, en veel van de informatie van het bedrijf staat op haar interne servers. Deze servers kunnen overdag vanuit het bedrijf bediend worden, maar soms is er 's avonds een storing in de servers waardoor bijvoorbeeld de website niet meer functioneert. Dan kan een *network engineer* naar het bedrijf gaan om de servers bijvoorbeeld te herstarten. Gelukkig bestaan er ook scripts, die in Python geschreven kunnen worden, die vanuit huis naar de servers kunnen worden gestuurd, zodat ze ook op afstand bediend kunnen worden.

Werksituatie 4 (Technische Informatica (TI))



Veel van de producten die bedrijf HU4U verkoopt, moeten eerst in elkaar worden gezet. Grote delen van dat proces kunnen worden overgenomen door robots. Robots kunnen automatisch onderdelen uit het magazijn halen, maar kunnen ook producten in elkaar zetten. Hiervoor moeten de robots wel eerst geprogrammeerd worden. Veel van de vaardigheden die vereist zijn voor de software engineer, zijn ook nodig voor de technische informaticus. Het grote verschil tussen de software engineer en de technische informaticus is dat de één code programmeert op het niveau van de computerprogramma's, terwijl de ander code programmeert op het niveau van de *hardware*: bijvoorbeeld een robotarm.

De werksituaties die hierboven beschreven zijn vereisen een aantal competenties. Deze competenties zijn vastgelegd door de HBO-i stichting in de Domeinbeschrijving Bachelor of ICT (http://www.hbo-i.nl/domeinbeschrijving-bachelor-of-ict/4416-het-model--activiteiten)..

1.3 Plaats cursus binnen het onderwijsprogramma

De cursus Programming wordt in het eerste halfjaar van de propedeuse gegeven tijdens het gemeenschappelijke deel dat iedere HBO-ICT'er moet volgen.

De cursus heeft tot doel de basisconcepten rondom programmeren te leren. Programming hangt samen met veel andere cursussen in het eerste jaar, waaronder Modelleren (zie Figuur 1). Bij **Modelling** worden (bedrijfs)processen en taken onderzocht en gemodelleerd op een manier die aan de ene kant de klant aanspreekt, en aan de andere kant de software-ontwikkelaars aanspreekt. Door goed te modelleren zullen er weinig onduidelijkheden zijn tijdens het programmeerproces.

In de cursus **Computer Systems and Networks** leren studenten hoe computers in de basis opgebouwd zijn, en hoe ze werken. Vooral wanneer complexe systemen gebouwd worden is het van belang om daar de juiste computer, of computeronderdelen (hardware), bij te kiezen.

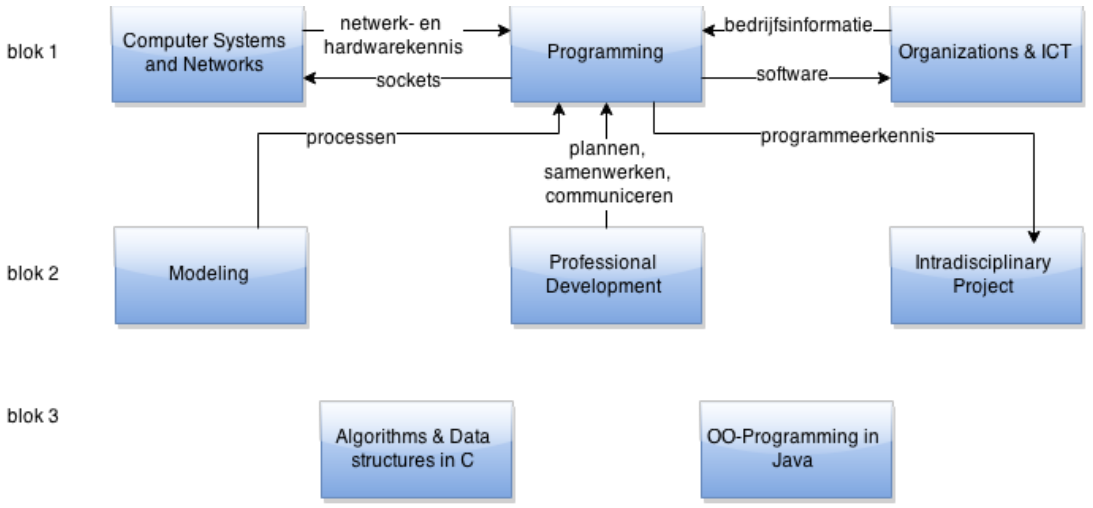
Naast deze kennis leren studenten hoe netwerken werken. Netwerkverbindingen kunnen in Python geprogrammeerd worden, bijvoorbeeld het ophalen of verzenden van email.

In de cursus **ICT and Organizations** leren studenten hoe organisaties werken, en wat hun rol in het geheel zou kunnen zijn. Vaak worden er in organisaties bepaalde processen geautomatiseerd, zoals Enterprise Resource Planning (ERP), waarbij de voorraadsystemen, bedrijfsadministratie en logistiek op elkaar afgestemd worden. Andere voorbeelden zijn Customer Relationship Management (CRM), en Business Intelligence (BI). Bij een ieder van deze voorbeelden moet je weten hoe de bedrijfsprocessen verlopen. Je moet ze kunnen analyseren, om ze vervolgens (deels) te kunnen automatiseren (software-ontwikkeling).

Tijdens de cursus **Professional Development** leren studenten om samen te werken, te communiceren en te plannen. Binnen een ontwikkelteam zijn dit cruciale vaardigheden om de samenwerking tussen teamleden vlot te laten verlopen.

In het **Intra Disciplinary project** brengt de student de geleerde kennis en vaardigheden uit de eerste periode tot uitdrukking in een groot project, waarbij er plaats is voor alle uitstroomprofielen van de opleiding HBO-ICT. Hier komt programmeren ook aan bod, omdat dit een groot onderdeel vormt van de werkzaamheden van de uitstroomprofielen Software and Information Engineering en Technische Informatica.

Bij de SIE-cursus **OO-Programming in Java** (blok 3) wordt er vanuit een object-georiënteerde manier gekeken naar programmeren. Objecten worden centraal gesteld, waardoor het makkelijker wordt om informatie (data) op te slaan in databases, en weer terug te halen. Object-georiënteerd programmeren maakt het makkelijker om programmeercode te hergebruiken. Over het algemeen wordt object-georiënteerd programmeren als redelijk complex en abstract ervaren door eerstejaars studenten. Daarom kiezen we ervoor om je eerst de globale concepten van programmeren in de cursus Programming ut te leggen, zodat je al weet wat programmeren inhoudt, en makkelijk de stap kunt maken naar 1) een andere programmeertaal zoals Java, en 2) object-georiënteerd programmeren. De TI-cursus **Algorithms & Data structures in C** gaat eerst nog dieper in op de algemene concepten van programmeren voordat de overstap naar object oriëntatie gemaakt zal worden.



Figuur 1: Overzicht samenhang met andere cursussen

1.4 Inhoud

1.4.1 Beroepstaken en professional skills

Binnen de Bachelor HBO-ICT werk je aan de vijf competenties Analyseren, Ontwerpen, Beheren, Adviseren en Realiseren binnen verschillende architectuurlagen: Hardware (H), Software (S), Bedrijfsprocessen (B), Gebruikersinteractie (G) en Infrastructuur (I). Daarnaast wordt gewerkt aan acht professional skills die iedere HBO-er moet bezitten: Creatief problemen oplossen, Analyse en Informatieverwerking, Leiderschap, Samenwerken, Communicatie, Plannen en Organisatie, Ethische verantwoording en Leren en persoonlijke ontwikkeling.

Tabel 1 laat zien dat deze cursus vooral gericht is op de architectuurlaag Software, omdat het resultaat van programmeren software is. Daarnaast zijn de leerdoelen per professional skill opgenomen.

1.4.2 Competentiematrix

Gewenst Niveau				ICT Beroepstaken						Professional skills																			
Zelfstandigheid (T,P of S)	Complexiteit inhoud	Complexiteit context		Beheren	Analyseren	Adviseren	Ontwerpen	Realiseren		Creatief problemen oplossen		Analyseren en info verwerking		Leiderschap		Samenwerken		Communicatie		Plannen en Organisatie		Ethische Verantwoording		Leren en persoonlijke ontw.					
+	1	1	G						+	+	Divergeren in flows (alternatieven bedenken)		keuze voor syntax mogelijk maken			+	Constructief feedback geven en ontvangen tbv eindresultaat team	+	Code commentaar	+	Taakverdeling en planning			+	Studievaardigheden voor analyseren en info. verwerking				
			B																										
			I																										
			S	+				1	1																				
			H																										

Tabel 1: Overzicht competenties

De competenties zelf zijn weer uitgewerkt in leerdoelen (zie 1.4.4).

1.4.3 Kennisbasis

[De kennisbasis overnemen uit de cursusbeschrijving en toelichten]

Tijdens deze cursus komen de volgende concepten aan de orde:

- Data
- IDE
- Ontwerp
- Flow control
- Implementatiekeuze
- Documentatie
- I/O
- Algoritme
- Testing
- Feedback
- Object Oriented Programming (OOP)
- Zelfreflectie
- Source control
- Namespaces
- Samenwerken

1.4.4 Leerdoelen

Competenties	Leerdoelen	Wordt getoetst in	
	De student kan..	**Toets 1 50 %**	**Toets 2 50%**
Creatief Problemen Oplossen	een vooraf gegeven taak vertalen in een ontwerp (bijv. Nassi-Shneidermandiagram) van een programma (algoritme), en hierbij correct gebruik maken van selection, iteration en functies.	X	X
	verschillende alternatieven / algoritmen genereren voor een gegeven taak (divergeren in flows).	X	X
	basic input- en output-functies (I/O) toepassen in een script (screen, keyboard en files).	X	X
	veelvoorkomende datatypes (number, string, dictionary, list, tuple) op correcte wijze kunnen gebruiken en daar bewerkingen op kunnen toepassen met behulp van operators.	X	X
	op een correcte manier gebruik maken van de scope (globaal of lokaal) van variabelen en functies (namespaces). Hierbij is een basale kennis van objecten en klassen benodigd.	X	X
	de flow van een programma contoleren door expressies correct te gebruiken in beslispunten (selection) of loops (iteration).	X	X
	weten wat source control is, en dat er verschillende tools zijn waarmee je dit kunt bewerkstelligen.	X	X
	een programma testen op correcte en logische werking (bijv. doctest) en kan omgaan met onverwacht gedrag in de code (debugging).		X
Analyse en Informatieverwerking	een ontworpen programmastructuur vertalen in Python-code (implementatie), en daarbij keuzes in de syntax kunnen onderbouwen.		X
Leiderschap Samenwerken Communicatie	constructief feedback geven én ontvangen ten behoeve van een beter eindresultaat van het team (bijv. code review).		X
	de werking van een programma toelichten door het gebruik van duidelijke en taalkundige juiste Python docstrings.		X
Planning & Organisatie Ethiek	zijn eigen Python ontwikkelomgeving inrichten waarbij gebruik wordt gemaakt van een Integrated Development Environment (IDE).		X
	bij het werken in een team een duidelijke planning en taakverdeling opstellen.		X
Leren & Persoonlijke Ontwikkeling	aangeven welke stappen zij ondernomen heeft om een bepaald programmeerconcept onder de knie te krijgen, en welke stappen hierin succesvol waren en welke minder goed werkten (zelfreflectie).		X

Tabel 2: Overzicht van leerdoelen per compententie en Professional Skill.

1.5 Toetsing

De inhoud, en daarmee de cursus, wordt beoordeeld op basis van

- Toets 1: Tentamen (50%, minimaal 5,5, **zie hoofdstuk 2 voor de tentamenstof**) en
- Toets 2: Miniproject (50%, minimaal 5,5)

Let op: als je een differentiatie-traject doorloopt (extra uitdagend) heeft dat gevolgen voor de looptijd van het miniproject dat hieronder beschreven is! Zie hiervoor §1.7**!!**

1.5.1 Tentamen (Summatieve toetsing)

De cursus wordt aan het einde van lesweek 6 getoetst met een tentamen waarbij we jouw kennis testen van de concepten die ten grondslag liggen aan programmeren. Dit tentamen bestaat uit een serie gesloten **multiple choice** vragen en zal, indien mogelijk, digitaal worden afgenomen.

Dit tentamen zal niet de programmeervaardigheden van de student toetsen, maar eerder het inzicht in het concept van programmeren. Een proeftentamen of oefenopdrachten op tentamenniveau tijdens de les zullen meer duidelijkheid verschaffen over de vorm waarin getoetst zal worden.

1.5.2 Miniproject (Summatieve toetsing)

Afhankelijk van het rooster zal in week 7, 8, of 9 het Miniproject plaatsvinden. Het Miniproject is erop gericht de programmeervaardigheden te toetsen in een project waarbij je samen moet werken in een team. De student krijgt een hele studieweek de tijd om de opdracht te voltooien, dat wil zeggen: 42 uur per teamlid.

De beoordeling van het miniproject wordt gedaan door de docent. Deze geeft in principe een teamcijfer, maar als de bijdrage per teamlid sterk verschilt, dan kan hiervan afgeweken worden. Zie voor de wijze van beoordeling, beschrijvingen, op te leveren producten en overige informatie m.b.t. het miniproject het document “Miniprojecten Programming” op Sharepoint!

Let op: bij het opleveren van het miniproject lever je ook een **portfolio** in met jouw **eigen** uitwerkingen van Practice exercises en Final assignments uit het werkboek dat je tijdens de collegelessen gebruikt! **Zonder dit portfolio is het niet mogelijk om een voldoende voor het miniproject te halen!**

1.5.3 Formatieve toetsing

In week 2 en 4 zullen er formatieve toetsen (niet voor studiepunten) worden afgenomen, om te kijken wat het niveau van de student is. Studenten krijgen inzicht in hun verbeterpunten. Als hier een ruim onvoldoende resultaat wordt gehaald, dan zullen we extra ondersteuning aanbieden om bij de student eruit te halen waarvan wij denken dat het erin zit.

De formatieve toets is ook het moment dat we studenten identificeren die extra uitdaging kunnen gebruiken, en die zullen we dan ook aanbieden in de vorm van uitdagende unitopdrachten, en het lange Miniproject bij een goede score bij de formatieve toets in week 2. Zie hiervoor paragraaf §1.7 m.b.t. **differentiatie!**

1.6 Herkansing

1.6.1 Planning

Bij een **onvoldoende voor het tentamen** is er een herkansingsmogelijkheid in blok B. De precieze datum en tijd van de herkansing is te vinden in Osiris.

Bij een **onvoldoende voor het Miniproject** is er een **herkansingsmogelijkheid** in blok B. Als het cijfer niet lager is dan een 4,5 is er ook de mogelijkheid om te **herstellen**. De deadline van beide mogelijkheden is de vrijdag van de herkansingsweek (lesweek 7) van blok B.

1.6.2 Herstellen of Herkansen?

Bij een onvoldoende voor het Miniproject is er sprake van vier mogelijkheden voor herkansing:

1. **Herkansen als team**, waarbij je een nieuwe opdracht probeert te maken. Je hebt hierbij de mogelijkheid om hoger dan een 5,5 te halen, omdat het een nieuwe opdracht betreft.
2. **Herstellen als team (alleen bij cijfers >= 4,5)**, waarbij je de oorspronkelijke opdracht probeert te verbeteren om zo tot een 5,5 te komen. Een 5,5 is het maximaal haalbare, omdat dit is een verkapte vorm van uitstel betreft.
3. **Herkansen als individu**, waarbij je de individuele herkansingsopdracht maakt.
4. **Herstellen als individu (alleen bij cijfers >= 4,5)**, waarbij je de oorspronkelijke opdracht probeert te verbeteren om zo tot een 5,5 te komen. Een 5,5 is het maximaal haalbare, omdat dit is een verkapte vorm van uitstel betreft.

Als je groep niet meer compleet is, zoek dan nieuwe teamleden (alleen in je eigen klas), of ga toch voor **Herstellen als individu** of **Herkansen als individu**. Overleg dit met je docent.

Denk bij je keuze goed na waarom je als team niet goed gefunctioneerd hebt tijdens de eerste kans van het Miniproject. Komt dat doordat de opdracht niet duidelijk is, of niet aansluit bij jullie kennis? Of komt dat doordat sommige teamleden hun taken niet op tijd af hadden, of zich simpelweg niet aan de afspraken hielden? Kijk in Appendix B van het document “Miniprojecten Programming” hoe je om kunt gaan met deze tegenslagen.

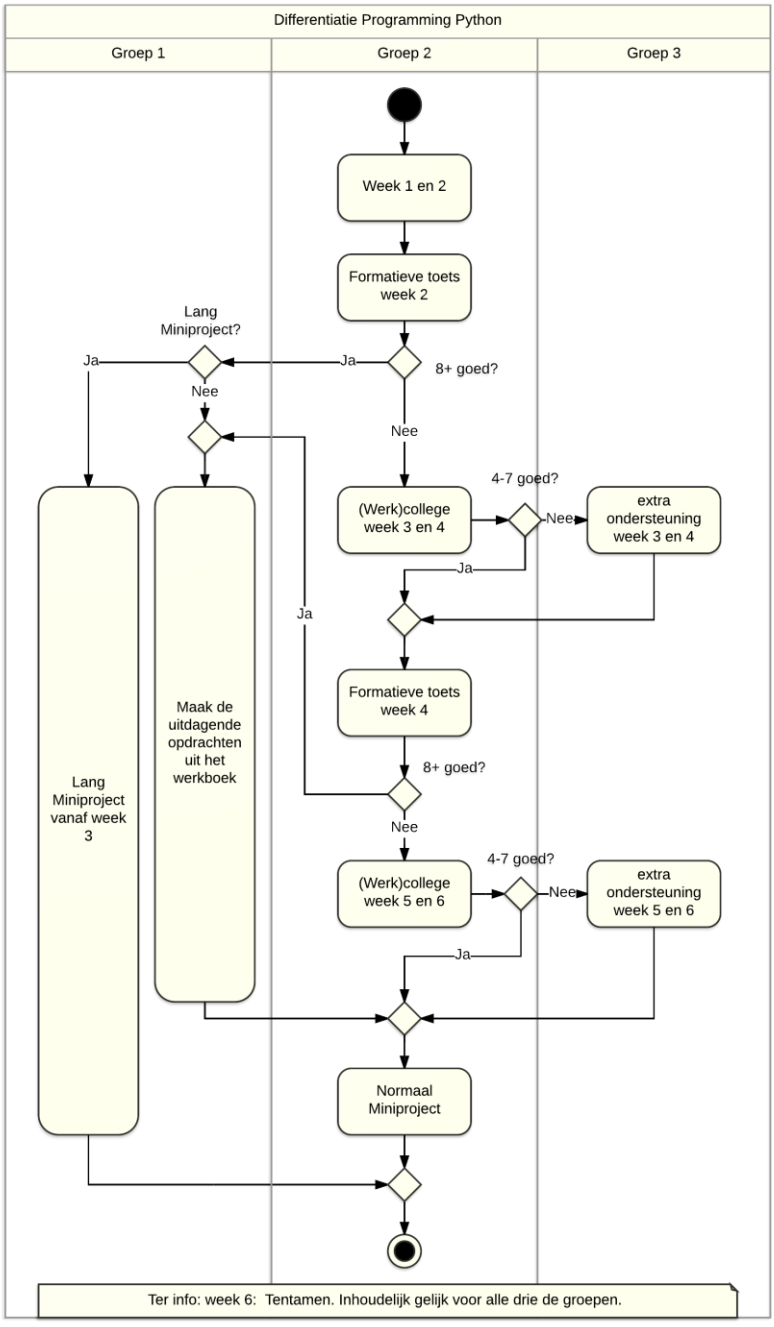
Let op: Als het teamproces niet loopt, dan is de verleiding groot om als individu de herkansings- of herstelopdracht te doen, omdat je denkt dat het beter zal gaan. Naast het feit dat je moet leren samenwerken, omdat de kans groot is dat je over vijf jaar in een team werkt, verlies je veel mogelijkheden die je als team hebt. Zo kun je niet meer overleggen over eventuele problemen/fouten in je code. Doordat je als teamleden verschillende perspectieven hebt, kan je eindproduct efficiënter en beter zijn. En vaak ben je gemotiveerder, omdat je niet alleen steun krijgt van je teamleden, maar ook in ‘hetzelfde schuitje zit’, namelijk de gezamenlijke herkansing. **We raden je dus aan om als team de herkansingsopdracht / herstelopdracht te doen.**

1.7 Differentiatie

Op basis van de formatieve toets zullen studenten een gedifferentieerde uitdaging (passend bij hun niveau) krijgen binnen de cursus. Dat wil zeggen: extra ondersteuning of extra uitdaging. Daarbij maken we onderscheid tussen drie groepen:

- Groep 1 bestaat vooral uit studenten die qua kennis en vaardigheden met Python al ver vooruit zijn op de inhoud van de cursus. Deze groep wordt gevormd na de formatieve toets in week 2. Het is niet verplicht om hieraan mee te doen!
- Groep 2 is de normaal presterende groep. Mochten studenten onvoldoende presteren, dan kunnen ze met wat extra inspanning toch (zelfstandig) een voldoende halen.
- Groep 3 is de groep die extra ondersteuning nodig blijkt te hebben. Deze groep heeft een **verplichte aanwezigheid bij het inloopspreekuur** aan het begin van de week. De docent zal dan aanwezig zijn om deze groep studenten apart te ondersteunen en zoveel mogelijk persoonlijke begeleiding te geven.

Het hele proces wordt weergegeven in de Figuur 2:



Figuur 2. Workflow differentiatie tijdens de cursus Programming

1.8 Leeromgeving

1.8.1 Opzet: werkvormen, type bijeenkomsten, opdrachten, begeleiding

De cursus wordt volgens het blended learning-principe gegeven, wat inhoudt dat van de student verwacht wordt dat hij/zij de leerstof eigen maakt in een combinatie van face-to-face colleges en zelfstudie, in dit geval de voorbereidingstijd en het werken aan de practicumopdrachten. Tijdens de cursus heeft iedere week nagenoeg hetzelfde schema. De weekindeling gedurende de eerste zes weken ziet er als volgt uit:

Lesvorm	Uren	Face-to-face / online / zelfstudie	Ondersteuning	Verplicht aanwezig?
2 x voorbereiding	3	zelfstudie	Nee	Nee
Inloopspreekuur	1	Face-to-face	Ja	Alleen groep 3
Werkcollege 1	2	Face-to-face	Ja	Ja
Werkcollege 2	2	Face-to-face	Ja	Ja
Online spreekuur	1	Online	Ja	Nee
Zelfstudie (huiswerk)	4	zelfstudie	Geen	Nee

Tabel 3. Weekindeling Programming

Voor ieder college is er **voorbereiding** vereist. Deze voorbereiding bevat de leerstof waarvan wij denken dat de student dit zich zelf eigen moet kunnen maken met de reader en het boek die zijn voorgeschreven, of met extra bronnen op internet. De stof voor de voorbereiding zal in het werkboek worden beschreven.

Bij het **inloopspreekuur** is de docent aanwezig om vragen van studenten persoonlijk te behandelen en een extra uitleg te geven. Dit spreekuur is verplicht voor groep 3. De plaats en tijd van het inloopspreekuur staan aangegeven op het rooster.

Hierna volgt het **werkcollege** waarin we voortbouwen op de voorbereide kennis: we gaan op de stof in door het maken van opdrachten, maar we leren ook nieuwe vaardigheden aan, zoals het gestructureerd en creatief problemen oplossen. Ook behandelen we tijdens het college onderwerpen waarvan we denken dat de studenten dit zichzelf niet (makkelijk) eigen kunnen maken, zoals het programmeren van bepaalde algoritmes.

Bij het **online spreekuur** is de docent online beschikbaar om vragen van individuele studenten over de stof te behandelen en een extra uitleg te geven. De tijd van het online spreekuur en het type medium zijn ter keuze van de docent, maar tijdens de opleiding wordt veelal gebruik gemaakt van Telegram, Whatsapp, Skype, Google Hangouts en Lync.

Begeleiding

Studenten krijgen begeleiding van de docent en mogelijk een student-assistent. De docent is aanwezig bij het inloopspreekuur, de (werk)colleges en het online spreekuur. De student-assistent is aanwezig bij de werkcolleges.

Aanwezigheid

Aanwezigheid bij het inloopspreekuur is verplicht voor de studenten die ruim onvoldoende hebben gescoord bij de formatieve toets. Wij zijn ervan overtuigd dat iedereen die in het HBO instroomt het vak Programming moet kunnen halen en helpen je daar graag bij.

Aanwezigheid tijdens contacturen is niet verplicht, maar de werkcolleges (4 uur per week) zijn zo opgebouwd dat ze voortbouwen op de zelfstudietijd (7 uur). Voortbouwen op de zelfstudietijd betekent dan ook dat er dieper in wordt gegaan op de kennis en vaardigheden die tijdens zelfstudietijd eigen zijn gemaakt. Ook zullen complexe onderwerpen tijdens het college behandeld worden in plaats van tijdens zelfstudietijd.

Gewenst studiegedrag van de deelnemers

De student wordt geacht vóór aanvang van het college de vereiste leerstof zelfstandig te hebben bestudeerd. Tijdens het college zal er voortgebouwd worden op de zelfstandig bestudeerde leerstof, en wordt er een actieve leerhouding van de student verwacht. Onder een actieve leerhouding wordt verstaan:

- op tijd aanwezig zijn
- geen huiswerk van andere cursussen maken tijdens het college
- geen games, messaging apps of andere dingen die jou en je medestudenten kunnen afleiden
- opdrachten tijdens college maken als dat gevraagd wordt
- vragen stellen als je die hebt
- bij samenwerking zorgen dat je een proactieve houding toont, afspraken nakomt, en constructieve feedback geeft

1.8.2 Bronnen

Binnen deze cursus maken we gebruik van:

- Literatuur **(verplicht)**: Introduction to Computing Using Python, an application development focus, Ljubomir Perkovic, Wiley, ISBN: 978-0-470-61846-2, **Second Edition**

- Cursussite (Sharepoint) waar meer bronnen op staan, zoals:
    - Slides
    - Werkboek met opdrachten, links naar tutorials en video's
    - Miniprojecten
    - Studenthandleiding
    - Ondersteunende literatuur

2 Overzicht cursusweken

Wk	Les	Onderwerp	Perkovic	Bijzonderheden
1	1	Starten met Python	§1.1 - 2.3	expressies, strings, lists
	2	Basisconcepten	§2.4 - 3.2	input, if/else, for-loop
2	3	Functions	§3.3 - 3.5	
	4	Text data+files / Exceptions	§4.1 - 4.4	**Formatieve toets**
3	5	Control Structures	§5.1 - 5.6	while-loop, list in list
	6	Containers (dict/set) / chars	§6.1 - 6.4	
4	7	Catching Exceptions / CSV	§7,3	
	8	Namespaces / XML	§7.1,2,4,5	**Formatieve toets**
5	9	GUI / API / Git(hub)	§9.1 - 9.3	
	10	GUI / API / Git(hub)	§9.1 - 9.3	
6	11	Proeftentamen		Bespreken
		Tentamen		Digitaal (multiple-choice)
7/8/9		Miniproject		Teamopdracht (1 week)

Tabel 4: Overzicht cursusweken (gele arcering == tentamenstof)

2.1 Opdrachten per week

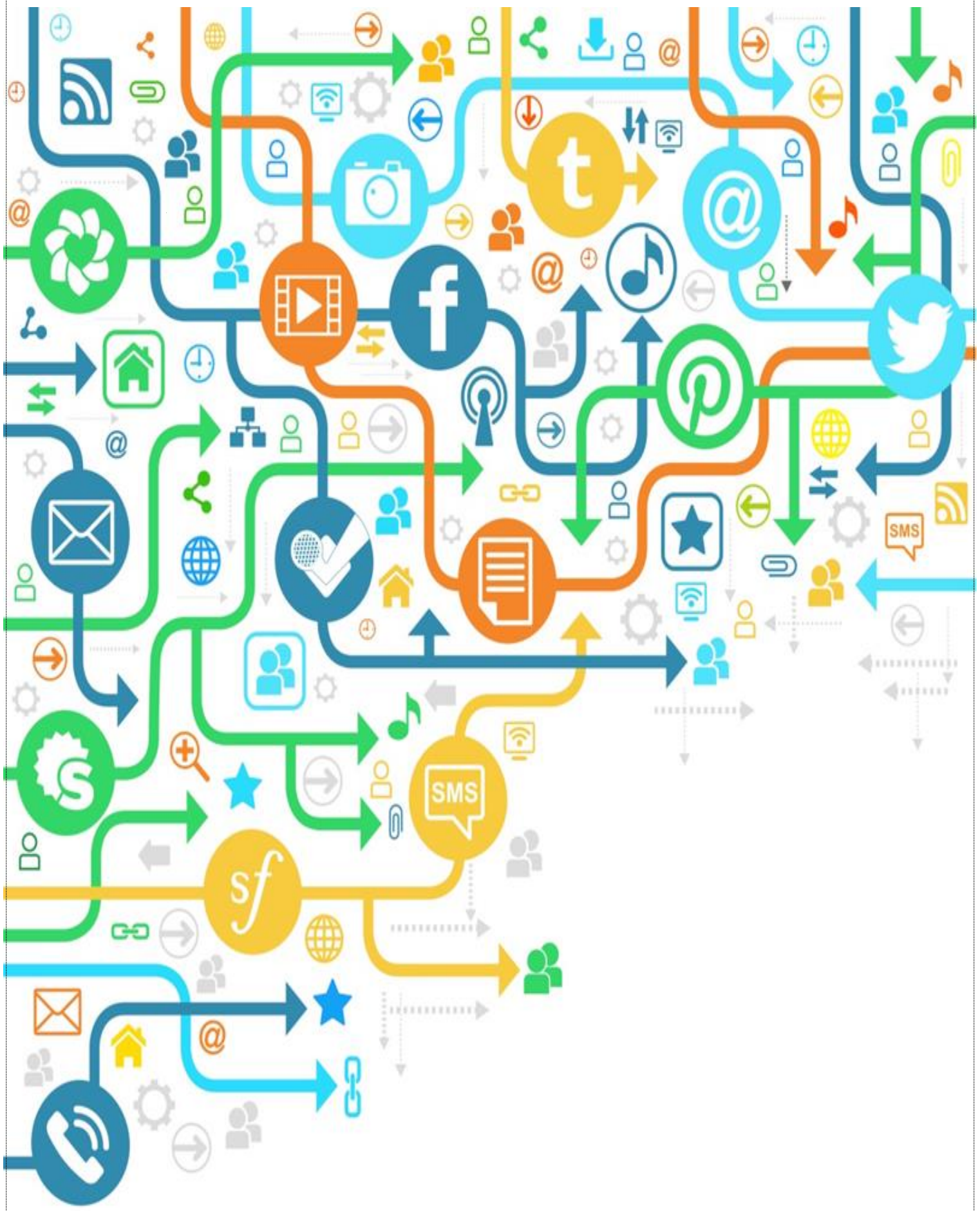
Iedere week worden er opdrachten gemaakt. De uitwerking van de **Final Assignment** lever je uiterlijk in tijdens het eerste (werk)college van de week die daarop volgt. Met andere woorden: de opdrachten van les 1 lever je uiterlijk tijdens werkcollege 1 van lesweek 2 in. Laat de uitwerking aan je docent of de studentassistent zien!

Waarom lever je elke week opdrachten in?

1) Alleen dan krijg je feedback van de docent of student-assistent
2) Tijdens de les bespreek je, net zoals later in je werk, de opdracht met een medestudent/collega.
3) De uitwerkingen van de **Practice Exercises** en **Final Assignments** moeten ingeleverd worden om het miniproject te kunnen behalen!

De beschrijving van de zelfstudiestof en opdrachten staan in het Werkboek op Sharepoint. Bij opdrachten voor het miniproject en in het portfolio moet duidelijk vermeld worden met welke studenten er is samengewerkt (indien van toepassing). Ook dienen de geraadpleegde bronnen erkend te worden.

Alle code en schriftelijke antwoorden moeten origineel zijn. We vertrouwen erop dat alleen eigen werk ingeleverd wordt, maar om de integriteit van de opleiding te waarborgen, zullen we actief controleren op **plagiaat**. Bij twijfel kan de examencommissie worden geïnformeerd.


SMS
SMS