

Programmeren in Python I

Een introductie tot programmeren en Python

Studiejaar	2015-2016
Periode	Blok 2
Doelgroep	Bio-Informatica propedeuse
Studiebelasting	7.5 ects
Auteurs	R. Piek, M.H.H. van der Bruggen, E. Kok
Versie	4.1

Versiebeheer

Versie 08-09 maakt gebruik van een nieuw boek: “Starting Out with Python”. Dit vindt zijn weerslag in de opbouw van de course. De onderdelen blijven hetzelfde maar worden in de volgorde van het boek besproken.

Versie 09-10 heeft bijgewerkte afvinkopdrachten en hyperlinks. Ook zijn er kennistoetsvragen opgenomen die overeenkomen met de nieuwe vorm van toetsen.

Versie 10-11 bevat een nieuwe verantwoording van gebruikt beeldmateriaal en tekstbronnen. Ook is de versie bijgewerkt naar de actuele versie van Python (2.6).

Versie 11-12

- Overstap naar Python 3.2 (van Python 2.6).
- Nieuwe versie van het boek “Starting out with Python: second edition”
- Studentevaluaties van 10-11 wezen op te weinig uitdaging. Op basis hiervan is besloten iedere week aan te geven welke opgaven van het boek gemaakt kunnen worden ter voorbereiding op de afvinkopdracht.
- Studentevaluaties wezen op onduidelijkheden in de afvinkopdrachten. Deze zijn bijgewerkt.
- Python Code met kleuring opgenomen
- Verwijzingen naar het toetsprogramma opgenomen.
- Hyperlinks en versienummering bijgewerkt.

Versie 11-12

- Afvinkopdrachten bijgewerkt en uit deze handleiding gehaald
- Afvinkopdrachten op online.han.nl gezet
- Naar aanleiding van opmerkingen van studenten de afvinkopdrachten verder verduidelijkt
- Datum van screenshots bijgevoegd op verzoek van Stichting Pro
- Hyperlinks bijgewerkt

Versie 12-13

- Afvinkopdrachten bijgewerkt en uit deze handleiding gehaald
- Afvinkopdrachten op online.han.nl gezet

- Naar aanleiding van opmerkingen van studenten de afvinkopdrachten verder verduidelijkt
- Datum van screenshots bijgevoegd op verzoek van Stichting Pro
- Hyperlinks bijgewerkt

Versie 13-14

- Afvinkopdrachten bijgewerkt
- Datum van screenshots bijgevoegd op verzoek van Stichting Pro
- Hyperlinks bijgewerkt
- Online Python bijgewerkt: codecademy en ideone bijgevoegd

Versie 14-15

- Teksten bijgewerkt
- Afvinkopdrachten bijgewerkt
- Nieuwe versie van het boek “Starting Out with Python: third edition”
- Aanpassing lesprogramma aan de hand van nieuwe boek

Versie 15-16

- Teksten bijgewerkt
- Handleiding geupdate aan de hand van OS
- Afvinkopdrachten bijgewerkt en toegevoegd aan handleiding

Contents

1	Inleiding	6
1.1	Algemene inleiding	6
1.2	Doelstelling	6
1.3	Competenties en algemene doelstellingen	6
1.4	Beginvereisten	6
1.5	Flexibilisering en instroom	7
1.6	Werkwijze, organisatie en planning	7
1.7	Leermiddelen	7
1.8	Try this at home	8
1.9	Beoordeling	8
2	Courseoverzicht	9
3	Studiewijzer	10
3.1	Werkwijze	10
3.2	Toetsing	11
3.3	Beoordeling	11
4	Python en Bio-Informatica	12
5	Python als Versatile Programmeertaal	13
6	Afvinkopdrachten	14
6.1	Afvinkopdracht 1	15
6.1.1	Input/Output en bewerkingen	15
6.1.2	Bonus opdracht	16
6.2	Afvinkopdracht 2	17
6.2.1	Restrictie enzymen	17
6.2.2	Bonus opdracht	17
6.2.3	Toelichting Flowcharts	18
6.3	Afvinkopdracht 3	19
6.3.1	Sikkelcel-anemie	19
6.3.2	Bonus opdracht	20
6.4	Afvinkopdracht 4	21
6.4.1	Functies	21
6.4.2	Bonus opdracht	21
6.5	Afvinkopdracht 5	22
6.5.1	Muizengenen	22
6.5.2	Bonus opdracht	22
6.6	Afvinkopdracht 6	23
6.6.1	Galgje	23
6.6.2	Alternatieve opdracht	23
6.6.3	Bonus opdracht	24
6.6.4	Bonus opdracht 2	24
6.7	Afvinkopdracht 7	25
6.7.1	DNA vertalen	25
6.7.2	Bonus opdracht	26
	Appendices	27
A	Voorbeeld Kennistoets Informatica	27

B	Voorbeeld Thematoets Eindopdracht	30
B.1	Casus	30
B.2	Opdracht	31
B.3	Beoordeling	32
B.4	Voorbeelduitwerking	33

1 Inleiding

1.1 Algemene inleiding

Het belangrijkste hulpmiddel voor bio-informatici is de computer. Vaak zal software om onderzoeksvragen te beantwoorden al aanwezig zijn. Meestal is dat echter niet op maat. Daardoor zal er soms nieuwe software geschreven moeten worden. Ook wordt er erg veel software geschreven om software te integreren. In dit project gaan we een begin maken met het ontwikkelen van programmeervaardigheden. De daarbij gebruikte programmeertaal heet Python, een binnen de bio-informatica heel erg populaire taal. Als bio-informaticus zul je zeker in aanraking komen met Python, door het zelf te leren schrijven kan je meerdere programmeertalen beter begrijpen.

Python is de programmeertaal waarmee we scripts schrijven om biologische problemen op te lossen.

1.2 Doelstelling

Het vak Python voor Bio-Informatica heeft als doelstelling de cursist te leren:

- een eenvoudig Python programma (ca. 100 regels code) te ontwerpen;
- een ontwerp te vertalen naar gedocumenteerde Python-code

1.3 Competenties en algemene doelstellingen

De **competenties**, die in dit thema centraal staan, zijn (allen op niveau 1):

- C2 Ontwikkelen software
- C4 Data analyseren (in mindere mate)
- C12 Sturen van professionele ontwikkeling

De **doelstellingen** van dit thema zijn:

- Programma's ontwerpen het ontwerp vertalen naar gedocumenteerde Python-code en de code te testen.

1.4 Beginvereisten

- T.a.v. Informatica: kunnen omgaan met Linux
- T.a.v. Wiskunde: niveau HAVO/MLO-Wiskunde (Zie opleidingsstatuut)
- T.a.v. Engels: engels lezen op HAVO-niveau (Zie opleidingsstatuut)

1.5 Flexibilisering en instroom

Dit thema kan in principe gevolgd worden door studenten van de HLO- en ICA-opleidingen en andere studenten met een N&T- of N&G-profiel; bij twijfel aan de geschiktheid van de student zal een intake-gesprek plaats vinden.

1.6 Werkwijze, organisatie en planning

Het vak wordt gegeven in de vorm van 7 wekelijkse lessen van elk 2 uur, die gedeeltelijk bestaan uit theorie, gedeeltelijk de vorm van een werkcollege (begeleid practicum) zullen hebben. Daarnaast is er 4 uur begeleid practicum per week, waar de programmeeropdrachten kunnen worden afgevinkt, en 4 uur onbegeleid practicum. Dit zal meestal niet genoeg zijn, reken op nog 1/2 tot 1 dag per week zelfwerkzaamheid. Daarnaast zal in het thema 2b veel van de programmeervaardigheid direct worden aangesproken en geoefend. Ook dit telt mee voor het verkrijgen van voldoende programmeervaardigheid. Verder is er tijd ingeruimd voor het bestuderen van en oefenen op de wiskunde die nodig is voor het uitvoeren van de beroepstaak.

De afvinkopdrachten in deze handleiding dienen **individueel** voltooid te worden. Dat betekent niet dat je geen hulp mag vragen aan medestudenten of aan de docent. Integendeel: **vraag veel aan de docent** en medestudenten. Professionele programmeurs vragen dagelijks elkaar om raad en gebruiken dagelijks fora om hun code te verbeteren.

1.7 Leermiddelen

De te gebruiken leermiddelen:

- Software
 - Python Interpretator/IDE <http://www.python.org/>
 - Flowchart Builder DIA <http://www.gnome.org/projects/dia/>
 - Flowchart Builder Online <http://www.gliffy.com/>
- Schriftelijk studiemateriaal
 - Dictaat: “Programmeren in Python I”
 - Boek: “Starting Out with Python” Third Edition Tony Gaddis Pearson Education ISBN 9780132576376
- Aanvullende documentatie
 - <https://docs.python.org/3/tutorial/index.html>
- Online Python

- http://www.compileonline.com/execute_python_online.php
- Python reference
 - http://www.brunningonline.net/simon/python/quick-ref2_0.html

1.8 Try this at home

1. Om Python uit te proberen kun je het makkelijkste Python 3.2 (<http://www.python.org/download/#pubkeys>) installeren. Er is ook Python 3.2. Echter de modules voor bio-informatica waar wij gebruik van maken zijn meestal gebaseerd op Python versie 2.7.
2. Je kunt eenvoudige opdrachten in Python ook online uitproberen op een website: <http://ideone.com/>.
3. Gebruik een Virtual Box Image waarmee je ook Python Programma's kunt uitvoeren.
4. Het is ook mogelijk Portable Python op Windows vanaf een USB stick te draaien: <http://portablepython.com/wiki/PortablePython3.2.1.1>.

1.9 Beoordeling

De beoordeling zal plaatsvinden conform het opleidingsstatuut. Zie hiervoor het toetsprogramma.

2 Courseoverzicht

De student schrijft in Python eenvoudige programma's om de analyse van biologische sequenties te automatiseren. Voornamelijk worden programma's geschreven om in biologische sequenties te zoeken, sequenties te vergelijken en te vertalen. De student pakt dit methodisch aan. Hij documenteert zijn programmacode en test de functionaliteit van zijn scripts.

Omvang 210 SBU

<table><tr><th colspan="3">Beroepstaak 2</th></tr><tr><td colspan="3">Het ontwerpen en ontwikkelen van software (Python) t.b.v. het automatiseren van het bio-informatica onderzoek</td></tr></table>			Beroepstaak 2			Het ontwerpen en ontwikkelen van software (Python) t.b.v. het automatiseren van het bio-informatica onderzoek		
Beroepstaak 2								
Het ontwerpen en ontwikkelen van software (Python) t.b.v. het automatiseren van het bio-informatica onderzoek								
<table><tr><th>Analyse</th></tr><tr><td>Maatstaven voor ligging en spreiding Populatie en steekproef Het belang van toetsen Binomiale verdeling en chi-kwadraat verdeling</td></tr></table>	Analyse	Maatstaven voor ligging en spreiding Populatie en steekproef Het belang van toetsen Binomiale verdeling en chi-kwadraat verdeling	<table><tr><th>Competenties</th></tr><tr><td>Ontwikkelen Software Methodisch werken Presenteren Samenwerken Professionele ontwikkeling</td></tr></table>	Competenties	Ontwikkelen Software Methodisch werken Presenteren Samenwerken Professionele ontwikkeling	<table><tr><th>Informatica</th></tr><tr><td>Basis van python Functies Exceptions Flowcharts</td></tr></table>	Informatica	Basis van python Functies Exceptions Flowcharts
Analyse								
Maatstaven voor ligging en spreiding Populatie en steekproef Het belang van toetsen Binomiale verdeling en chi-kwadraat verdeling								
Competenties								
Ontwikkelen Software Methodisch werken Presenteren Samenwerken Professionele ontwikkeling								
Informatica								
Basis van python Functies Exceptions Flowcharts								
<table><tr><th>Communicatie</th></tr><tr><td>SCV</td></tr></table>	Communicatie	SCV	<table><tr><th>Beroepsproducten</th></tr><tr><td>Werkende scripts voor analyseren van biologische sequenties (python) Code ontwerp Gedocumenteerde eigen code</td></tr></table>	Beroepsproducten	Werkende scripts voor analyseren van biologische sequenties (python) Code ontwerp Gedocumenteerde eigen code			
Communicatie								
SCV								
Beroepsproducten								
Werkende scripts voor analyseren van biologische sequenties (python) Code ontwerp Gedocumenteerde eigen code								

3 Studiewijzer

De theorie van Programmeren in Python is opgedeeld in 7 lessen. In het onderstaande overzicht wordt per les aangegeven welke onderwerpen aan bod komen, welke delen uit het Python Leerboek en deze syllabus bestudeerd dienen te worden en welke opgaven daarbij horen. De afvinkopdrachten waarnaar verwezen wordt zijn te vinden in bijlage B van deze syllabus.

Les	Onderwerp	Opgaven
1	Hoofdstuk 1: <i>Introduction to Computers and Programming</i> Hoofdstuk 2: <i>Input, Processing and Output</i>	Programming Exercises: Opdracht 1, 3 en 9 (Hoofdstuk 2, pagina 95 en verder) Afvinkopdracht 1 (6.1)
2	Hoofdstuk 3: <i>Decision Structures and Boolean Logic</i> Flowcharts	Programming Exercises: Opdracht 4, 6, 15 (Hoofdstuk 3, pagina 133 en verder) Afvinkopdracht 2 (6.2)
3	Hoofdstuk 4: <i>Repetition Structures</i>	Programming Exercises: Opdracht 6, 8, 14 (Hoofdstuk 4, pagina 179 en verder) Afvinkopdracht 3 (6.3)
4	Hoofdstuk 5: <i>Functions</i>	Programming Exercises: Opdracht 4, 11, 20 (Hoofdstuk 5, pagina 247 en verder) Afvinkopdracht 4 (6.4)
5	Hoofdstuk 6: <i>Files and Exceptions</i>	Programming Exercises: Opdracht 6, 9 (Hoofdstuk 6, pagina 306 en verder) Afvinkopdracht 5 (6.5)
6	Hoofdstuk 7: <i>Lists and Tuples</i>	Programming Exercises: Opdracht 8, 9 (Hoofdstuk 7, pagina 352 en verder) Afvinkopdracht 6 (6.6)
7	Hoofdstuk 8: <i>More About Strings</i>	Programming Exercises: Opdracht 2, 4, 10 (Hoofdstuk 8, pagina 384 en verder) Afvinkopdracht 7 (6.7)

3.1 Werkwijze

Het boek heeft aan het eind van ieder hoofdstuk een onderdeel review questions. Deze zijn representatief voor de kennistoets. Door deze te oefenen kun je je voorbereiden op de kennistoets. Daarna is er een onder programming exercises, hierin staan programmeeropdrachten. Hiermee kun je je voorbereiden op de thematoets (praktijkopdracht achter de computer). Iedere week zijn daar opdrachten uit geselecteerd om te maken ter voorbereiding van de afvinkopdracht.

3.2 Toetsing

Er is een kennistoets aan het eind van dit vak en een thematoets (eindopdracht) die beide met een cijfer beoordeeld worden. Er is voor de kennistoets een tussentijdse bonustoets. Indien hiervoor een 5.5 of hoger wordt behaald dan wordt de eind kennistoets met 1 punt opgehoogd. Deeltentamens voor deze course zijn:

- BI2a-Af Afvinkopdrachten (Af)
- BI2a-T Thematoets (T) (praktische opdracht)
- BI2a-K Kennistoets Informatica en Statistiek (K, Ktt (tussentijds tentamen) en Ket (eindtentamen))

Afvinkopdrachten moeten zijn afgevinkt met voldaan als voorwaarde voor deelname aan de thematoets.

3.3 Beoordeling

De beoordeling wordt als volgt bepaald:

- Kennistoets K (Informatica 50%, Statistiek 50%)
- Thematoets T (Informatica)
- Eindcijfer: $(K + T)/2$

Cijfers toetsen moeten tenminste 5.5 zijn. $K = Ket$, maar als $Ktt \geq 5,5$, dan is $K = Ket + 1$. Ktt is 1 jaar geldig. Voor detaillering van de beoordeling bekijk het opleidingsstatuut, onderdeel toetsprogramma.

4 Python en Bio-Informatica

Python is een programmeertaal die door bio-informatici erg veel gebruikt wordt. Het is een bijzonder krachtige taal die erg dicht bij gewone spreektaal staat. Dat is de reden dat we het een hoge programmeertaal noemen. Het is een zogenaamde derde generatietaal (3GL) waarin het mogelijk is Object-georiënteerd te programmeren. Vaak noemen we dit programmeren ook scripten. Er is niet een helder onderscheid tussen scripten en programmeren. Wanneer echter een aantal logische commando's wordt gecombineerd in een bestand spreken we meestal van een script. Wanneer er duidelijk een interactie met gebruikers en een grafische user interface spreken we van een programma.

Biologische vraagstellingen kunnen we vaak verhelderen met een scripttaal. Dat betekent dat we bijvoorbeeld data uit verschillende databases halen en vergelijken en daarmee tot een verheldering van het antwoord komen.

Zouden we bijvoorbeeld willen weten of een gen dat we gevonden hebben gerelateerd is aan een bekende ziekte, dan kunnen we een script schrijven wat alle databases die we kennen doorzoekt op het gen en zoekt naar een relatie met een ziekte.

5 Python als Versatile Programmeertaal

Python is een erg populaire programmeertaal voor veel populaire toepassingen. Vrijwel iedere dag maak je indirect gebruik van de programmeertaal Python. Bijvoorbeeld Youtube en Google zijn voor een belangrijk deel gemaakt met Python. Het is dus niet alleen voor ons als bio-informatici geschikt om te gebruiken maar ook voor allerlei andere serieuze toepassingen.

Zo zouden we met Python een programma kunnen schrijven waarmee de kaas-database toegankelijk wordt gemaakt voor mensen die geen SQL beheersen. We kunnen er ook programma's mee schrijven om bijvoorbeeld moleculen te bekijken (PyMol <http://pymol.sourceforge.net/>) of games mee programmeren (PyGame <http://www.pygame.org/>) of een bittorrent client mee schrijven. De eerste bittorrent client was geschreven in Python. (http://en.wikipedia.org/wiki/Bram_Cohen)

6 Afvinkopdrachten

In dit hoofdstuk, op de volgende pagina's, vind je alle afvinkopdrachten voor de komende weken. Maak voordat je een afvinkopdracht gaat maken eerste de opgaven uit het boek, ter voorbereiding en ter oefening. Werk hieraan volgens de studiewijzer (hoofdstuk 3).

6.1 Afvinkopdracht 1

6.1.1 Input/Output en bewerkingen

Een dipeptide is een molecuul dat bestaat uit twee aminozuren. Dipeptides kennen niet de kenmerkende structuren van een groot eiwit. Toch hebben veel dipeptides een belangrijke biologische betekenis. Zo is van een aantal dipeptides bekend dat ze betrokken zijn bij de signaaloverdracht in de hersenen. Een van de dipeptides komen we zeer veelvuldig tegen en wel in frisdranken als suikervervanger: Aspartaam.

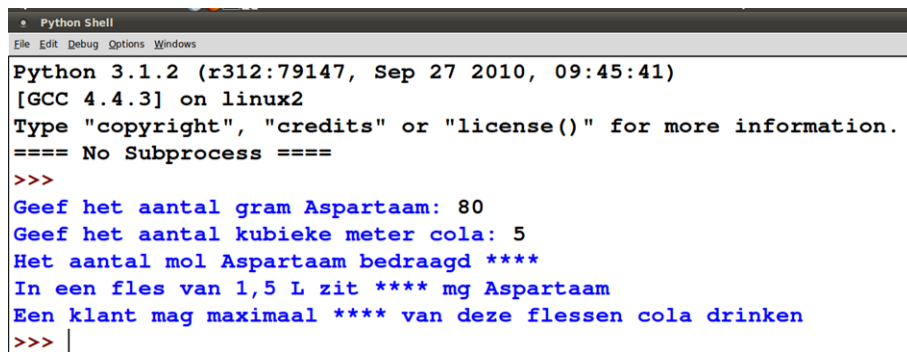
Aspartaam is een zoetstof die qua smaak zeer dicht in de buurt van suiker komt en is per gram 160 maal zo zoet als suiker. Het bestaat uit de aminozuren asparaginezuur en fenylalanine. Bij de afbraak van aspartaam ontstaan de aminozuren asparaginezuur en fenylalanine maar ook methanol. Methanol wordt door het lichaam geoxideerd tot formaldehyde en dat weer tot mierenzuur. Beide stoffen kunnen schadelijk zijn voor de mens. Vandaar dat de ADI (Aanvaarde Dagelijkse Inname) van Aspartaam door de Europese Unie op $40mg * kg^{-1}$ lichaamsgewicht is bepaald.

Bron: <http://en.wikipedia.org/wiki/Aspartame>

Aspartaam	
Molecuulformule	$C_{14}H_{18}N_2O_5$
Energie	0 kcal per gram
Molmassa	$294,30g * mol^{-1}$

Je loopt stage bij een frisdrank fabriek in Lieshout en krijgt de opdracht een programma te ontwikkelen dat na invoer van de hoeveelheid gram Aspartaam en daarna het aantal m^3 de volgende gegevens weergeeft.

- Het aantal toegevoegde mol Aspartaam
- De hoeveelheid Aspartaam in 1,5 L in mg
- Het aantal flessen dat een klant maximaal mag drinken per dag zonder de ADI te overschrijden. Ga er van uit dat de gemiddelde klant een massa van 70 kg heeft.



```
Python 3.1.2 (r312:79147, Sep 27 2010, 09:45:41)
[GCC 4.4.3] on linux2
Type "copyright", "credits" or "license()" for more information.
==== No Subprocess ====
>>>
Geef het aantal gram Aspartaam: 80
Geef het aantal kubieke meter cola: 5
Het aantal mol Aspartaam bedraagt ****
In een fles van 1,5 L zit **** mg Aspartaam
Een klant mag maximaal **** van deze flessen cola drinken
>>> |
```

Fig. 1: Voorbeeld van een run van het script (screenshot eigen werk auteurs)

6.1.2 Bonus opdracht

Vraag naar een naam en naar het gewicht van de persoon en geef diegene een gepersonaliseerd advies. Als iemand meer weegt mag die persoon dus meer cola drinken voordat hij/zij aan de ADI zit.

6.2 Afvinkopdracht 2

6.2.1 Restrictie enzymen

Er bestaan erg veel verschillende restrictie enzymen. Hiermee kunnen we een methode opzetten om DNA te analyseren. Schrijf een script dat op basis van de invoer van de naam van een restrictie enzym bepaalt of er een knip gezet wordt in de DNA reeks gegeven in het bestand `startOpgave2.py`. Werk dus het script in `startOpgave2.py` verder uit. Doe dit voor alle restrictie enzymen gegeven in het bestand.

1. Teken de flowchart om de biologische vraagstelling op te lossen.
2. Vul (op basis van je flowchart) het script aan van `startOpgave2.py` met code die aangeeft welke enzymen knippen in de sequentie opgehaald door de functie `getSequentie()`.
3. Voorzie je bestand van commentaar en schrijf deze weg als `opgave2enzymen.py`
4. Draai het script in dezelfde map als `startOpgave2.txt`, anders kan het script het bestand niet vinden

Voor het tekenen van je flowchart kun je gebruik maken van:

- Pen en papier!
- Dia (voorgeïnstalleerde versie op Linux of <http://dia-installer.de/download/index.html.en>)
- Online flowchart builder <http://www.gliffy.com/>
- <https://www.draw.io/>
- Open Office / Microsoft Office Visio

6.2.2 Bonus opdracht

Geef per restrictie enzym op welke positie(s) van het DNA hij geknipt heeft. Dit is een moeilijke opgave!

6.2.3 Toelichting Flowcharts

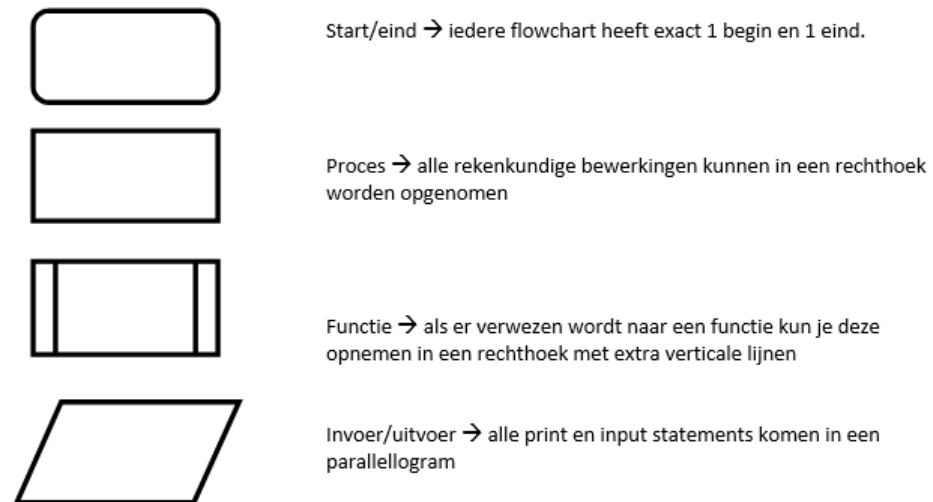


Fig. 2: Toelichting bij de verschillende onderdelen van de flowchart

6.3 Afvinkopdracht 3

6.3.1 Sikkelcel-anemie

In het DNA is een base veranderd. Een A is omgezet in een T. Hierdoor wordt er op positie 6 van de β -keten een valine ingebouwd in plaats van een glutamaat. Het resultaat is dat zonder zuurstof het eiwit polymeriseert.

Voor de diagnostiek van sikkelcel-anemie is een gedeelte van het hemoglobine-gen m.b.v. PCR vermeerderd. Het gaat om het gebied rond de start van het eerste intron (startcodon in sequentie dikgedrukt). Het is bekend dat dit gebied een van de mutaties kan bevatten die aanleiding geven tot sikkelcel-anemie. Hieronder vind je de sequenties van een sikkelcel-gen met die mutatie en van het normale gen zonder de mutatie.

Als blijkt dat die mutatie resulteert in het verdwijnen of juist verschijnen van een restrictiesite, kan voor toekomstige prenatale diagnostiek hiervan gebruik worden gemaakt. Men hoeft dan alleen nog maar het betreffende gebied te vermeerderen m.b.v. PCR, gevolgd door een knip met dit restrictie-enzym en analyse op een agarose-gel.

Kortom: je opdracht is om een aantal restrictie enzymen te testen tegen onderstaande sequenties. Zoek uit of een van de restrictie enzymen geschikt is om een onderscheid te maken tussen beide sequenties.



Fig. 3: Gewone en sikkelcelvormige rode bloedcellen (Bron wikipedia: http://en.wikipedia.org/wiki/Sickle-cell_disease)

Sikkelcel gen:

```
GAGCCATCTATTGCTTACATTTGCTTCTGACACAACTGTGTTCACTAGCAACCTCAAACA  
GACACCATTGGTGACCTGACTCCTGTGGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAG  
GTGAACGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTGGTATCAAGGTTACAAGAC  
AGGTTTAAGGAGACCAATAGAAACTGGGCATGTGGAGACAGAGAAGACTCTTGGGTTTCT
```

Normaal gen:

```
GAGCCATCTATTGCTTACATTTGCTTCTGACACAACTGTGTTCACTAGCAACCTCAAACA  
GACACCATTGGTGACCTGACTCCTGAGGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAG  
GTGAACGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTGGTATCAAGGTTACAAGAC  
AGGTTTAAGGAGACCAATAGAAACTGGGCATGTGGAGACAGAGAAGACTCTTGGGTTTCT
```

1. Teken een flowchart voor je code.
2. Open het script startOpgave3.py
3. Creëer een script dat door alle enzymen loopt uit het bestand: enzy-
men.txt
4. Vergelijk de restrictiepatronen van beide sequenties. Welk restrictie-enzym
is geschikt voor bovengenoemde diagnostiek?
5. Voorzie je bestand van commentaar.

6.3.2 Bonus opdracht

Geef van beide sequenties het stuk coderend DNA op (ofwel, het stuk dat begint met ATG en eindigt bij een stopcodon). Zorg dat dit in-frame is.

6.4 Afvinkopdracht 4

6.4.1 Functies

Schrijf een python programma met de volgende functies:

1. `gcPercentage()` voor het bepalen van het GC percentage. Deze functie retourneert een getal dat het percentage GC in de meegegeven DNA sequentie representeert.
2. `isDNA()`, functie die controleert of alle nucleotiden werkelijk in DNA voorkomen. Deze functie retourneert een boolean die indiceert of alle letters bestaan uit ATGC.
3. `getComplement()`, functie die de complementaire streng oplevert. Deze functie retourneert een String die de complementaire streng representeert.

De aanroep van deze functies mag je doen door in je programma een klein stukje code te schrijven die dit regelt (bijvoorbeeld met een `main()` functie), of door ze via de IDLE aan te roepen.

6.4.2 Bonus opdracht

Schrijf nog een extra functie, `getCodons()`, die vanaf het eerste startcodon (ATG) de DNA streng ophapt in codon stukjes en dit als uitvoer geeft.

Optioneel: stop ook weer op het moment dat je een stopcodon tegenkomt (TAA/TAG/TGA).

Je uitvoer komt er dan ongeveer zo uit te zien:

```
>>> DNA streng in codons:  
ATG AAA TAC GCG TAT GAA AAC GTT TTG CGC GAG AGA GCG TGT GTA GGC CCA TAG
```

6.5 Afvinkopdracht 5

6.5.1 Muizengenen

Op de site van het Hugelab kunnen we alle informatie terugvinden van het genoom van de muis. Op http://hugelab.ccbr.utoronto.ca/supplementary-data/IRC/IRC_representative_cdna.fa.gz staat bijvoorbeeld het bestand waarin alle bekende genen van de muis staan.

Deze bestanden zijn in FASTA (http://en.wikipedia.org/wiki/FASTA_format) formaat, dat houdt in dat alle gevonden coderende stukken DNA voorafgegaan worden door een regel met het groter dan teken: >.

Schrijf een programma dat vraagt om een woord en als resultaat alle regels die vooraf gegaan worden door > en waarin dat woord voorkomt toont.

Uiteraard zorg je voor exception handling. Indien het bestand niet bestaat geeft het programma een foutmelding en ook alle onverwachte fouten worden afgevangen.

6.5.2 Bonus opdracht

Bereken voor alle gevonden stukken DNA het GC percentage.

6.6 Afvinkopdracht 6

6.6.1 Galgje

Schrijf een Python programma dat een tekst bestand met woorden in kan lezen (woorden.txt). Als het bestand niet bestaat wordt deze exception afgevangen en weergegeven dat het bestand niet bestaat. De woorden lees je in en plaats je in een lijst (list). Uit deze lijst kiest je programma een willekeurig woord.

De gebruiker moet nu proberen het woord te raden. Het programma vraagt herhaaldelijk om een letter, wanneer de letter in het woord voorkomt geeft je programma aan op welke positie de letter voorkomt. Als de letter er niet in zit krijgt de gebruiker een strafpunt en toont het programma de status van de speler in een plaatje, bijvoorbeeld zoals in figuur 1. Bij 10 strafpunten ben je af. Bouw je programma op met behulp van functies en zorg voor een goede exception afhandeling. Denk ook aan goede documentatie!

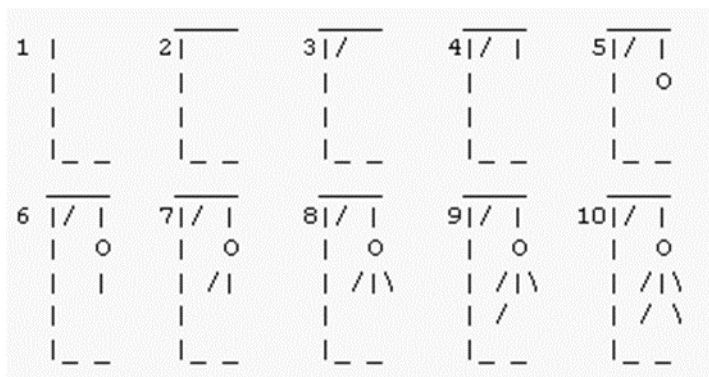


Fig. 4: Voorbeeld van ASCII code weergave van de status van de speler (Bron Wikipedia: <http://nl.wikipedia.org/wiki/Galgje> 10-nov-2012)

6.6.2 Alternatieve opdracht

Verzin zelf een spelletje waarin de volgende componenten zitten:

- Inlezen van een bestand
- Gerandomiseerd kiezen van elementen uit een lijst
 - List elementen
 - Tuple elementen
- Invoer door gebruiker
- Output naar de commandline, optioneel met een leuke ASCII art.
- Error handling

- Niet bestaand bestand
- Invalide invoer

Je kan hierbij denken aan:

- Een (kleine) tekst RPG
- Een poker/blackjack spelletje
- Een virtuele soorteerhoed quiz of soortgelijks
- Etc. etc....

De mogelijkheden zijn eindeloos en je mag creatief zijn. Hierbij is het zeker belangrijk om je code goed te documenteren, zodat ik kan zien wat de bedoeling van je spelletje is.

Deze opdracht mag je in tweetallen maken. De les na de vakantie gaan we een kleine live-demo houden waarbij je kan laten zien wat jullie gemaakt hebben.

6.6.3 Bonus opdracht

Lees een extra tekstbestand in met prijzen die mensen kunnen winnen als ze je spelletje gewonnen hebben. Geef de winnaar feedback middels prints naar het scherm en bedenk leuke prijzen.

6.6.4 Bonus opdracht 2

Laat mensen inloggen met een naam en het programma houdt de score voor die persoon bij. Tip: Je kan een bestand maken waarin je kan lezen en schrijven die dit bijhoudt.

6.7 Afvinkopdracht 7

6.7.1 DNA vertalen

Het centrale dogma van de microbiologie, zoals opgesteld door Francis Crick in 1958, stelt dat informatie overgedragen kan worden van nucleïnezuren (DNA en RNA) naar eiwitten, maar niet andersom. Binnen de microbiologie en bio-informatica wordt veelvuldig gekeken hoe de vertaling van DNA naar eiwitten verloopt. Een streng DNA kan vertaald worden naar meerdere eiwitten, die dus een andere functie in de cel vervullen. De tussenstap van deze vertaling is het RNA. Als bio-informaticus gaan jullie aan de slag met het vertalen van een stuk DNA naar RNA en zoeken naar patronen in dit RNA om te controleren of de vertaling goed gelukt is.

Jullie opdracht is om een programma te schrijven dat de sequentie inleest uit een bestand (sequentie.txt) en de volgende elementen bevat:

1. Controle of het bestand wel uit een DNA sequentie bestaat, inclusief exception handling als het bestand niet bestaat of als er een onverwachte fout optreedt.
2. Zet de DNA sequentie om naar een RNA sequentie (in hoofdletters). LET OP: De DNA sequentie in het bestand is de coding strand van het DNA en loopt van 5' naar 3'. Jullie moeten dus een RNA strand krijgen die ook van 5' naar 3' loopt, gebaseert op deze coding strand. Zie figuur 1 voor een visualisering van dit proces.
3. Zoekt naar de start en stop codons in de verkregen RNA sequentie en zoekt in-frame, dus na het vinden van het start codon mogen out-of-frame stop codons niet meer geaccepteerd worden.
4. Geeft op basis van de gevonden codons aan op welke posities deze liggen. Hou verschillen tussen (string)indexen en sequentie posities in de gaten. Noem als positie de plek van de eerste letter van het codon.
5. Geef als resultaat het gevonden transcript op.

Het resultaat zal er ongeveer zo uitzien:

```
>>> RNA sequentie:
UAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGC
UAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGC
UAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGC
Start codon gevonden op positie:
Stop codon gevonden op positie:
Gevonden transcript:
AUGUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAGCUAA
>>>
```

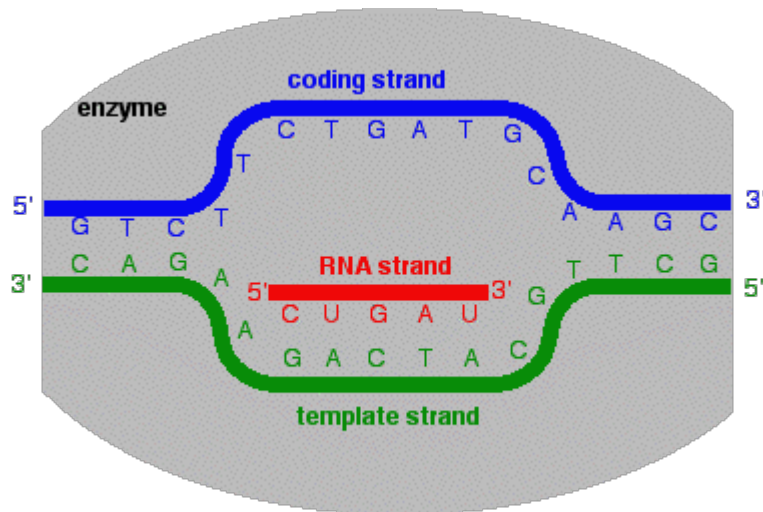


Fig. 5: Transcriptie van DNA naar RNA via coding en template strand. Bron: <http://www.chemguide.co.uk/organicprops/aminoacids/dna3.html> (20 november 2014)

6.7.2 Bonus opdracht

Je hebt de DNA streng gekregen die van 5' naar 3' loopt (forward ofwel coding). Nu is de opdracht om de RNA codering van de reverse strand (ofwel template strand) te krijgen. In figuur 1 staat beschreven hoe de RNA sequentie er uit komt te zien als je de coding strand gebruikt, maar jullie opdracht is nu om dus de DNA template strand te vertalen naar RNA.

Appendices

A Voorbeeld Kennistoets Informatica

1. Wat zal de waarde van a zijn?

```
a = 4 + 2 = 8
```

- (a) 8
- (b) true
- (c) false
- (d) Niets, dit levert een fout op

2. Wat zal het volgende statement printen?

```
print(int(9.0/3.0))
```

- (a) 3.0
- (b) 3
- (c) 9/3
- (d) Niets, dit is een foutief statement

3. Wat zal het volgende statement printen?

```
print(float(9.0/3.0))
```

- (a) 3.0
- (b) 3
- (c) 9/3
- (d) Niets, dit is een foutief statement

4. `def wissel(a,b):`

```
    c = a  
    x = b  
    y = c  
    print(x)  
    print(y)
```

Bij de functie wissel zijn:

- (a) a,b en c lokale variabelen die niet buiten de functie te raadplegen zijn
- (b) x en y zijn parameters die buiten de functie te raadplegen zijn
- (c) x, y en c zijn lokale variabelen en zijn niet te raadplegen buiten de functie
- (d) a, b, c, x en y zijn allen globale variabelen

5. `def gc(dna):`
 `print((dna.count('c') + dna.count('g'))/float(len(dna)) * 100.0)`
De functie zal bij een aanroep van `gc("AATTAGAG")` het volgende printen.
- (a) 25
 - (b) 0.0
 - (c) 25.0
 - (d) 0
6. `for naam in ["apen", "noten", "miezen"]:`
 `if "a" in naam:`
 `print(naam)`
Wat print dit statement?
- (a) apen, noten en miezen
 - (b) apen
 - (c) noten
 - (d) miezen
 - (e) Geen van bovenstaande
7. `while 1:`
 `print("HelloWorld")`
Bovenstaande statement is een voorbeeld van:
- (a) Een syntactisch onjuiste loop
 - (b) Een oneindige loop
 - (c) Een niet startende loop
8. `i = 0`
`for x in range(10):`
 `for y in range(10):`
 `i = i + 1`
 `print(i)`
Wat is de waarde van `i` aan het eind?
- (a) minder dan 10
 - (b) gelijk aan 10
 - (c) meer dan 10
 - (d) dit statement geeft een error
9. `for a in [0]:`
 `print(a)`
Wat print deze loop?

- (a) 0
- (b) 1
- (c) 2
- (d) Niets

10. Welke functie vereist geen import van een library?

- (a) `cos(10)`
- (b) `randrange(100)`
- (c) `len("bio – informatica")`
- (d) `ceil(100.10)`

Antwoorden:

1	2	3	4	5	6	7	8	9	10
d	b	a	c	b	b	b	d	a	c

B Voorbeeld Thematoets Eindopdracht

Thematoets Python I – blok 2 2008/2009

Bio-Informatica 1: dinsdag 20 januari 13h00-17h00

B.1 Casus

Uit onderzoek is een lijst opgesteld met alle tot nu toe bekende genen.

(http://www.broad.mit.edu/genome_bio/trc/trc.geneId.human.apr07.lst)

De lijst kent een zogenaamde tab-delimited opmaak met onderstaand fragment als voorbeeld.

sourceId	symbol	description
NM_130786	A1BG	alpha-1-B glycoprotein
NM_000022	ADA	adenosine deaminase
NM_001792	CDH2	cadherin 2, type 1, N-cadherin (neuronal)
NM_014249	NR2E3	nuclear receptor subfamily 2, group E, member 3
NM_005467	NAALAD2	N-acetylated alpha-linked acidic dipeptidase 2
NM_005469	ACOT8	peroxisomal acyl-CoA thioesterase
NM_005470	ABI1	abl-interactor 1

Figuur 1: fragment van de 16112 rijen uit genes.txt. Kolommen zijn gescheiden door tabs (tab-delimited)

Onderzoekers hebben nog een groot aantal vragen over deze genen. Deze lijst is ontzettend groot (16112 rijen) en wordt iedere maand weer groter. Nb. Dit is een real-life bestand. Momenteel wordt deze lijst met de hand gecontroleerd en bewerkt. Dat is iedere maand veel werk.

Het onderzoeksinstituut wil deze werkzaamheden automatiseren. Graag zouden ze een mogelijkheid hebben om genen te zoeken een symbool. Ook willen ze de data controleren op inconsistenties. Nu kan het voorkomen dat een sourceId meer dan een keer voorkomt, maar handmatig valt dit niet te controleren.

Als aanvullende opdracht willen ze de lijst naar een nieuw bestand wegschrijven gesorteerd op symbool. Het onderzoeksinstituut veronderstelt dat er onvoldoende tijd voor beschikbaar is en beschouwt dat als een bonus opdracht.

B.2 Opdracht

Ga uit van het bestand *genes.txt*.

Het op te leveren script dient aan de volgende **functionele eisen** te voldoen:

1. Leest het bestand *genes.txt*
2. Bij ontbreken van het bestand of een leesfout genereert het een nette foutmelding waarin staat dat het script het bestand niet kan vinden.
3. Het script vraagt om een symbool en toont alle genen die als symbool het opgegeven symbool hebben.
4. *Genes.txt* wordt gecontroleerd op de aanwezigheid van dubbele *sourceId*'s. Wanneer dubbele *sourceId*'s worden geconstateerd wordt dit weergegeven.
5. Er wordt een nieuw bestand gemaakt: *nieuw.txt*. Hierin staan alle genen gesorteerd op symbool.

Het op te leveren script dient aan de volgende **niet functionele (technische eisen)** te voldoen.

1. Ontwerp een flowchart voor de opdrachten en schrijf bijbehorende code.
2. Het script is opgedeeld in functies.
 - Er is een functie *zoekSymbool* die een symbool als parameter accepteert en vervolgens alle genen toont met dat symbool.
 - Er is een functie *zoekDubbeleIDs* die controleert of ieder id slechts een keer voorkomt.
 - Er is een functie *maakNieuw* die een nieuw bestand (*nieuw.txt*) maakt waarin de rijen gesorteerd staan op symbool.
 - Alle functies worden vanuit een *main* functie aangeroepen.
3. Alle te verwachten exceptions worden afgevangen.
4. De snelheid van het script is van geen belang.
5. Het script is geschreven in Python.

B.3 Beoordeling

Naam student:

Bestandsnaam van de in te leveren code: bil<studentnaam>.py

Bestandsnaam van de in te leveren flowchart: bil<studentnaam>.py

HBI-Cou2a Beoordelingsformulier Thematoets

	Punten	Beoordeling	
		Student	Docent
I. Code			
a. Maakt gebruik van variabelen en gebruikt de juiste datatypes	10		
b. Past de juiste controlestructuren toe	10		
c. Bouwt programma op volgens standaard met een onderverdeling in functies	10		
d. Schrijft functies met parameterisering	10		
e. Schrijft functies die waarden retourneren	10		
f. Documentatie conform de standaard	10		
II. Juiste Werking			
a. Bestand lezen	10		
b. Exception Handling	20		
c. Voert de gevraagde bewerking juist uit	30		
III. Flowchart			
a. Flowchart is volgens standaard	10		
b. Flowchart stemt overeen met de code	20		
Eindoordeel	150		

Inleveren

Geschreven code en een eigen beoordeling van je geschreven code aan de hand van bovenstaande beoordelingscriteria. Vul daartoe de kolom student in. N.B. Het eindoordeel van de docent is hetgeen geadministreerd wordt. Het cijfer is het aantal behaalde punten delen door het maximaal aantal te behalen punten maal 10. Afgerond op een decimaal.

B.4 Voorbeelduitwerking

```
# Auteur: M van der Bruggen, E. Kok
# Creatie: 7 jan 2011
# Updated: 12 okt 2015 - verbeterde zoek functie
# Functie:
# Known bugs:

def openBestand(inBestand):
    try:
        bestand = open(inBestand)
        return bestand
    except IOError:
        print ("Bestand_niet_gevonden")

# Deze functie zoekt per symbool een description
def zoekSymbool(b4, sym):
    lijst = []
    for regel in b4:
        lijst = regel.split("\t")
        if sym in lijst[1]:
            print ("Gevonden", lijst[1], "met_desc:", lijst[2])

def zoekDubbel(x):
    lijst = []
    srid1 = []
    for regel in x:
        lijst = regel.split("\t")
        srid1.append(lijst[0])
    for id1 in srid1:
        if srid1.count(id1)>1:
            print ("SourceID:", id1, "komt", srid1.count(id1), "_keer_voor")

def nieuw(x):
    newFile = open ("Nieuw.txt", "w")
    lijst = []
    srid1 = []
    for regel in x:
        lijst = regel.split("\t")
        if lijst[0] not in srid1:
            newFile.write(regel)
            print ("Regel_toevoegen:", regel)
        else:
            print ("Dubbele_", lijst[0], "_niet_weggeschreven.")
            srid1.append(lijst[0])
    newFile.close()
```

```
x.close()
print ("Bestand_weggeschreven")

def main():
    b5 = openBestand("genes.txt")
    nieuw(b5)
    b5.close()

main()
```