Universiteit Gent
Faculteit Wetenschappen

Programmeren in Python

durf probleemoplossend denken

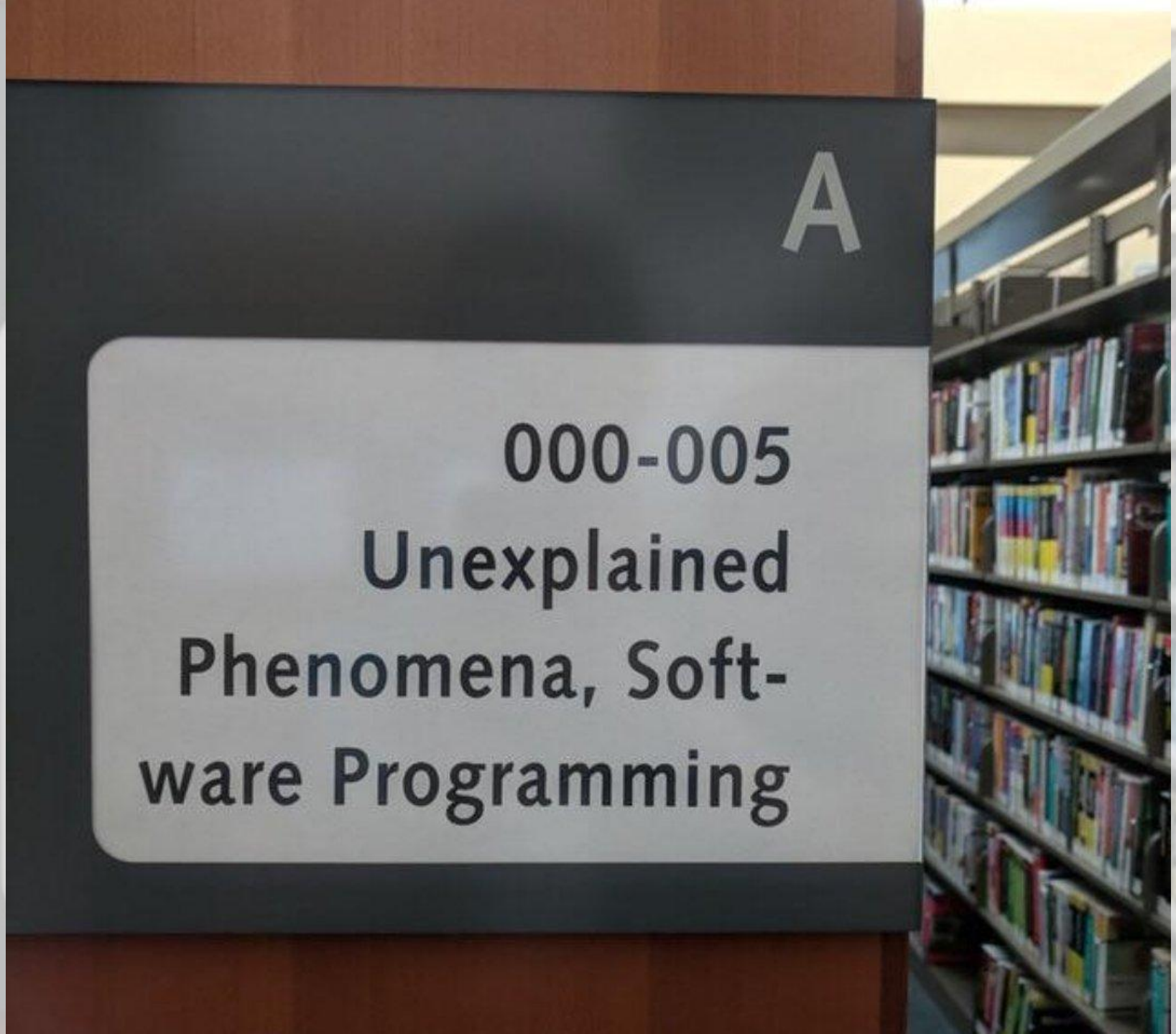
prof. dr. Peter Dawyndt

peter.dawyndt@ugent.be

@dawyndt

UNIVERSITEIT GENT

Les programmeren start binnenkort ...


A
000-005
Unexplained
Phenomena, Soft-
ware Programming

Programmeerervaring?

In welke taal programmeerde je al het vaakst?

A. Scratch
B. Java
C. Python
D. HTML5 (HTML, CSS & JavaScript)
E. nog nooit geprogrammeerd

UNIVERSITEIT GENT

socrative

socrative.com

room: 407122

Doelstelling

leren denken als een computerwetenschapper

Doelstelling

deze manier van denken combineert ideeën uit

- wiskunde
  - gebruik van formele talen om ideeën uit te drukken
- ingenieurswetenschappen
  - dingen ontwerpen
  - componenten samenvoegen tot grotere systemen
  - voor- en nadelen afwegen tussen alternatieven
- natuurwetenschappen
  - problemen formuleren
  - hypothesen uitdenken
  - voorspellingen testen

leren denken als een computerwetenschapper

Doelstelling

probleemoplossend denken is de belangrijkste vaardigheid voor een computerwetenschapper

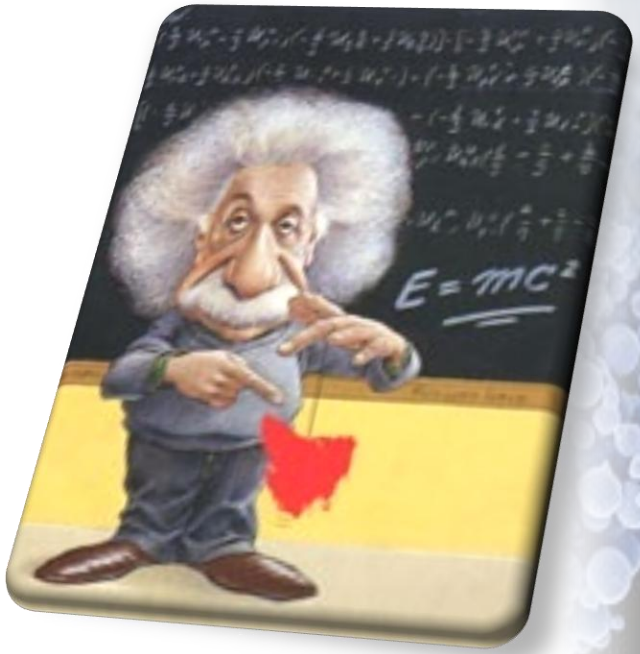
- problemen formuleren
- creatief nadenken over oplossingen
- oplossingen helder en nauwkeurig omschrijven

leren programmeren gaat over
het leren oplossen van problemen

Doelstellingen

kunnen > kennen

learning by doing



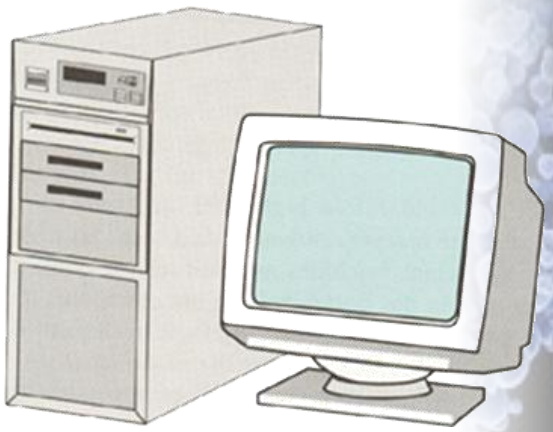
UNIVERSITEIT GENT

Ontwerpcyclus

natuurlijke taal
(Nederlands, Engels, Frans, …)

programmeertaal
(Python, Java, C++, …)

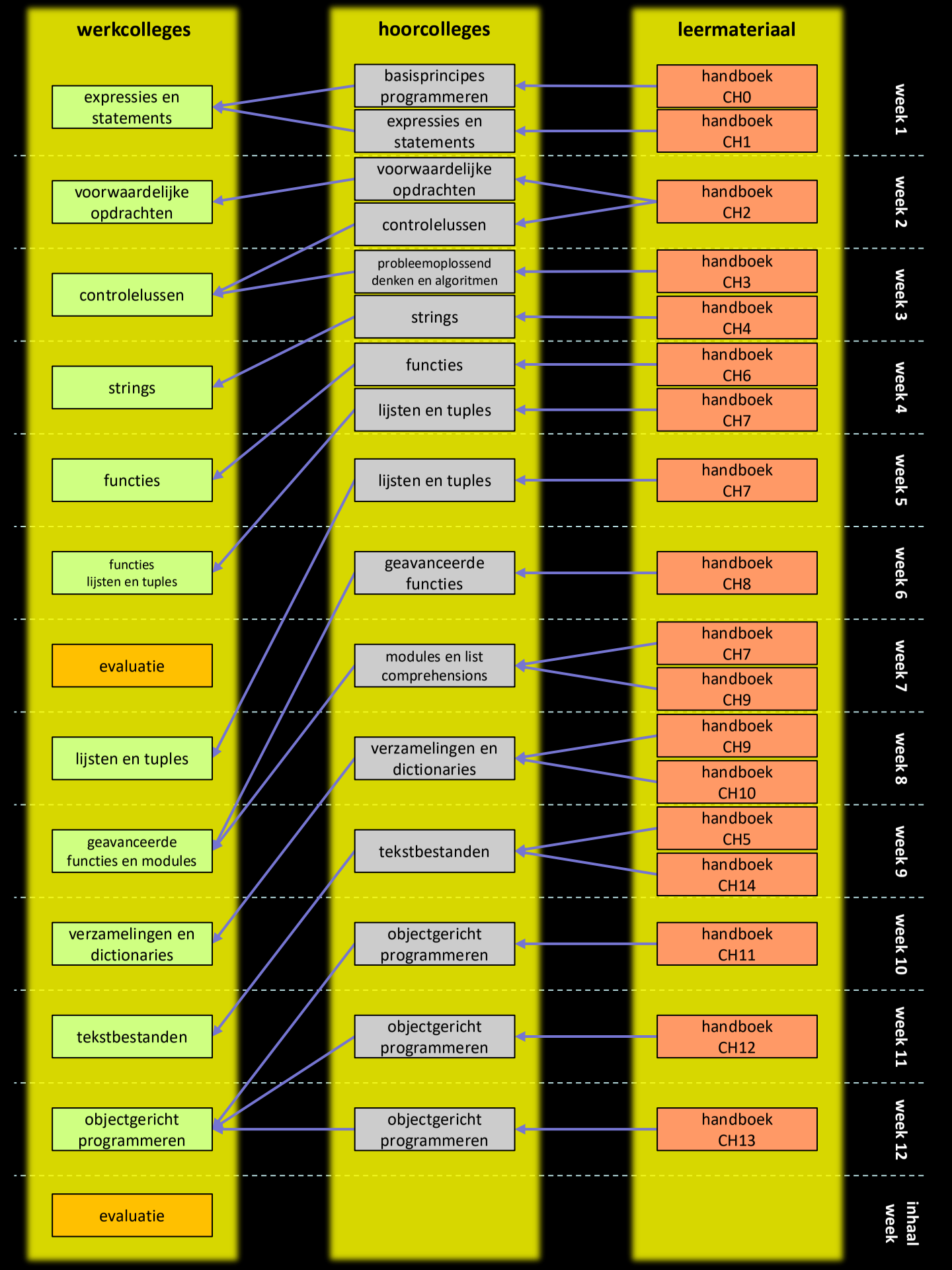
machinetaal
(Pentium, Xeon, AMD, …)



probleemoplossend denken

kennis programmeertaal

UNIVERSITEIT GENT


werkcolleges
hoorcolleges
leermateriaal
expressies en statements
voorwaardelijke opdrachten
controlelussen
strings
functies
functies lijsten en tuples
evaluatie
lijsten en tuples
geavanceerde functies en modules
verzamelingen en dictionaries
tekstbestanden
objectgericht programmeren
evaluatie
basisprincipes programmeren
expressies en statements
voorwaardelijke opdrachten
controlelussen
probleemoplossend denken en algoritmen
strings
functies
lijsten en tuples
lijsten en tuples
geavanceerde functies
modules en list comprehensions
verzamelingen en dictionaries
tekstbestanden
objectgericht programmeren
objectgericht programmeren
objectgericht programmeren
handboek CH0
handboek CH1
handboek CH2
handboek CH3
handboek CH4
handboek CH6
handboek CH7
handboek CH7
handboek CH8
handboek CH7
handboek CH9
handboek CH9
handboek CH10
handboek CH5
handboek CH14
handboek CH11
handboek CH12
handboek CH13
week 1
week 2
week 3
week 4
week 5
week 6
week 7
week 8
week 9
week 10
week 11
week 12
inhaal week

Ontwerpcyclus

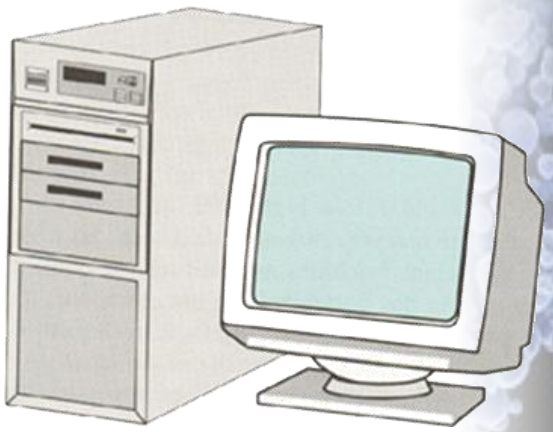
probleemoplossend denken

natuurlijke taal
(Nederlands, Engels, Frans, …)

kennis programmeertaal

programmeertaal
(Python, Java, C++, …)

machinetaal
(Pentium, Xeon, AMD, …)



UNIVERSITEIT GENT

Ontwerpcyclus

A beschrijf en analyseer het probleem

- *wat moet het programma doen?*
- *wat is de invoer, en wat de gewenste uitvoer?*

B ontwerp een algoritme (*pseudocode*)

- *hoe bereik je het vooropgestelde doel?*

C zet algoritme om in broncode (*source code*)

- volgens syntaxisregels van gekozen programmeertaal

D compileer broncode tot machinetaal

D voer het programma uit

E spoor eventuele fouten op (*debuggen*)

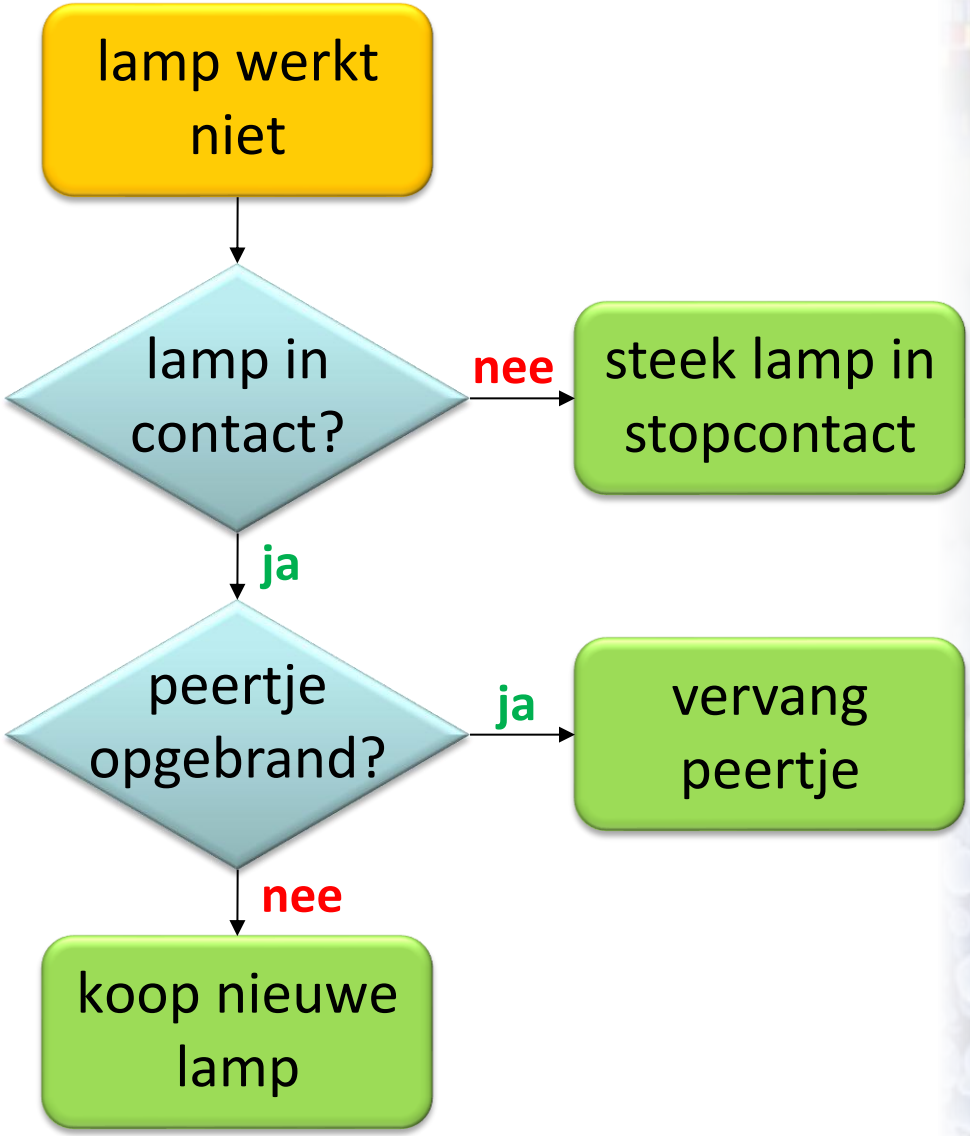
UNIVERSITEIT GENT

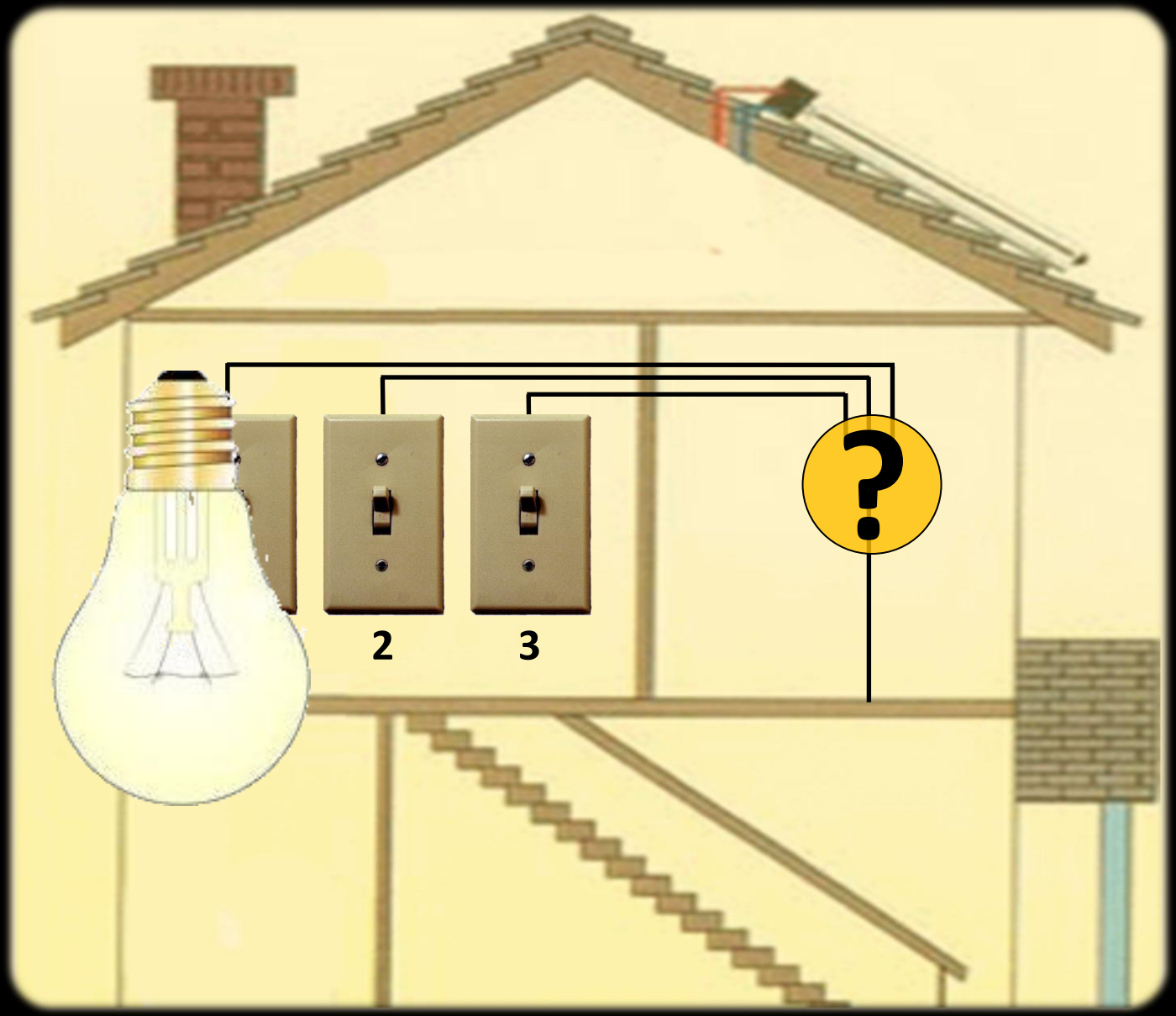
socrative

socrative.com

room: 407122

Algoritme


lamp werkt niet
lamp in contact?
nee
steek lamp in stopcontact
ja
peertje opgebrand?
ja
vervang peertje
nee
koop nieuwe lamp


2
3
?

Russische postzegel uitgegeven op 6 september 1983 ter ere van de (ongeveer) 1200e verjaardag van Muhammad ibn Mūsā al-Khwārizmī. Perzisch wiskundige, sterrenkundige, astroloog en geograaf. Het woord algoritme is afgeleid van Algoritmi, de verlatijnsing van zijn naam.

FOTO WIKIMEDIA

Algoritme

UNIVERSITEIT GENT

Programmeren

Programmeren

Programmeren is het **automatiseren** van bepaalde taken of toepassingen met behulp van de computer.

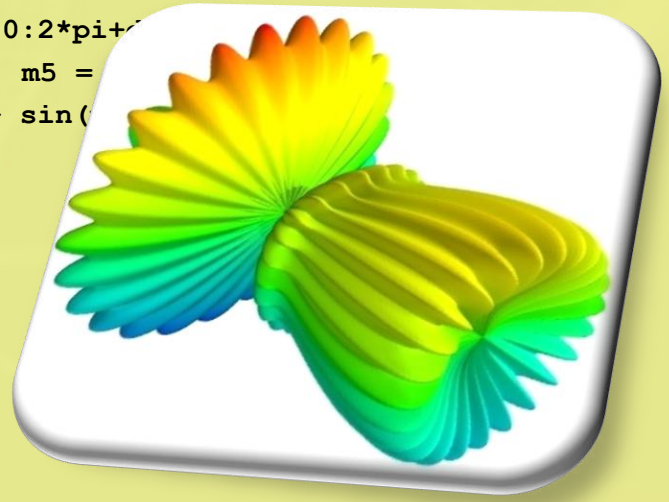
Dit gebeurt door middel van een reeks **instructies** neergeschreven in een **programmeertaal**.

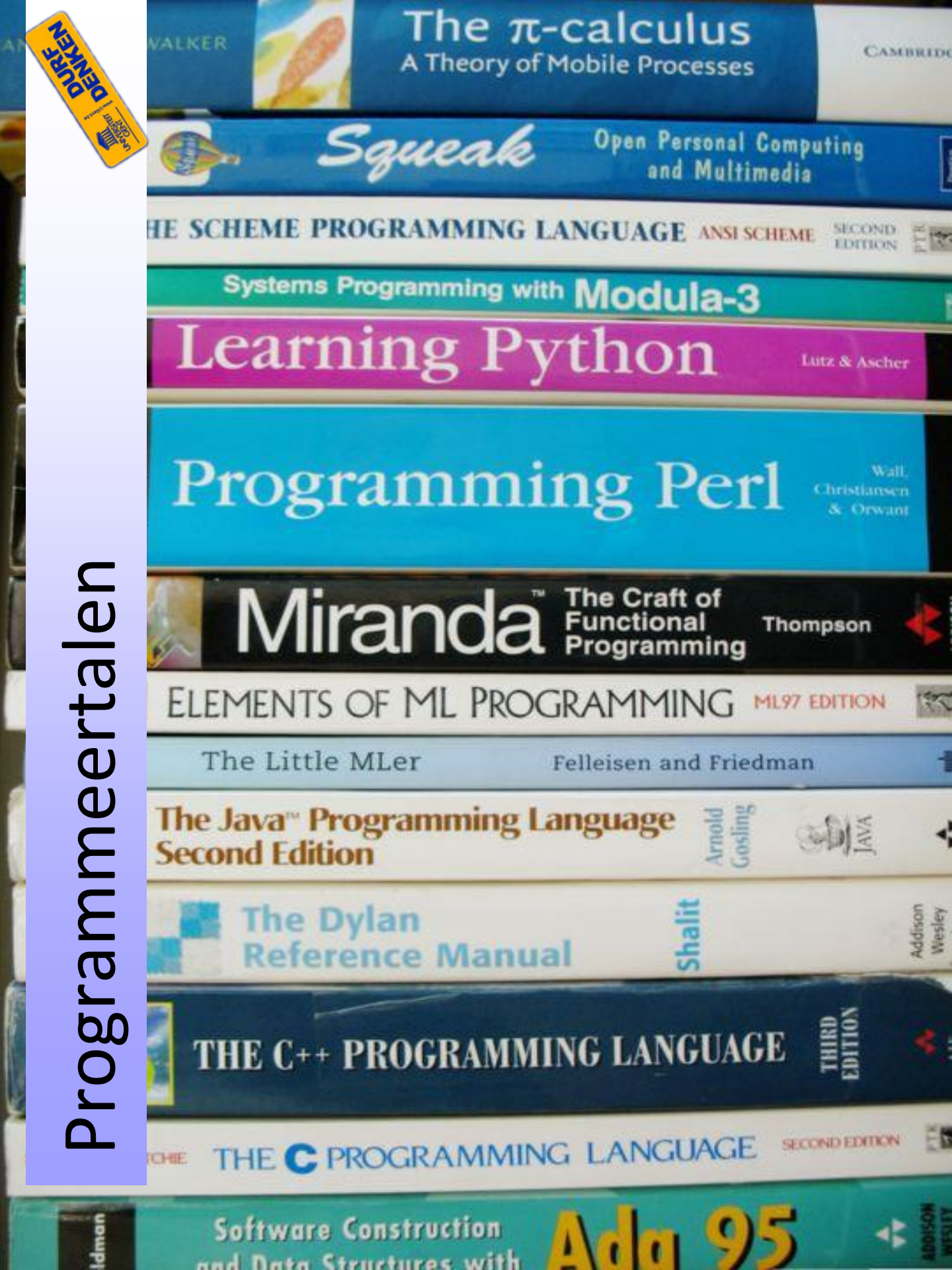
Programmeertalen

Een **programmeertaal** is een taal waarin de opdrachten worden geschreven die door een computer moeten uitgevoerd worden. Deze talen hebben een andere **syntaxis** en **grammatica** dan natuurlijke talen, aangezien de talen die door mensen wordt gesproken hiervoor veel te complex en veel te ambigu zijn. Code die in een programmeertaal geschreven is (**programmacode**, **broncode**), mag immers maar op één enkele manier door een computer kunnen "begrepen" worden.


```
# harmonische functie opstellen
from numpy import pi, sin, cos, mgrid
dphi, dtheta = pi/250.0, pi/250.0
[phi,theta] = mgrid[0:pi+dphi*1.5:dphi,0:2*pi+
m0 = 4; m1 = 3; m2 = 2; m3 = 3; m4 = 6; m5 =
r = sin(m0*phi)**m1 + cos(m2*phi)**m3 + sin(
      cos(m6*theta)**m7
x = r*sin(phi)*cos(theta)
y = r*cos(phi)
z = r*sin(phi)*sin(theta)

# harmonische functie weergeven
from mayavi import mlab
s = mlab.mesh(x, y, z)
mlab.show()
```




DURF DENKEN
Programmeertalen
The π-calculus
A Theory of Mobile Processes
Squeak
Open Personal Computing and Multimedia
THE SCHEME PROGRAMMING LANGUAGE
ANSI SCHEME
SECOND EDITION
Systems Programming with Modula-3
Learning Python
Lutz & Ascher
Programming Perl
Wall, Christiansen & Orwant
Miranda™
The Craft of Functional Programming
Thompson
ELEMENTS OF ML PROGRAMMING
ML97 EDITION
The Little MLer
Felleisen and Friedman
The Java™ Programming Language
Second Edition
Arnold Gosling
JAVA
The Dylan Reference Manual
Shalit
Addison Wesley
THE C++ PROGRAMMING LANGUAGE
THIRD EDITION
THE C PROGRAMMING LANGUAGE
SECOND EDITION
Software Construction and Data Structures with
Ada 95
ADDISON WESLEY

Programmeertalen

- twee soorten programmeertalen
  - hoog-niveau programmeertalen
    - Python, Java, C/C++/C#, Perl, Ruby
  - laag-niveau programmeertalen
    - ook assembleertalen of machinetalen genoemd
    - hardware-specifiek
      - Intel x86 (Pentium) – IBM PC compatibel
      - Motorola/IBM PowerPC – Apple Macintosh; IBM servers
      - Sun SPARC – Sun servers
- computers kunnen enkel machinetaal uitvoeren
  - programma geschreven in hoog-niveau programmeertaal wordt vertaald naar machinetaal voor specifieke computer

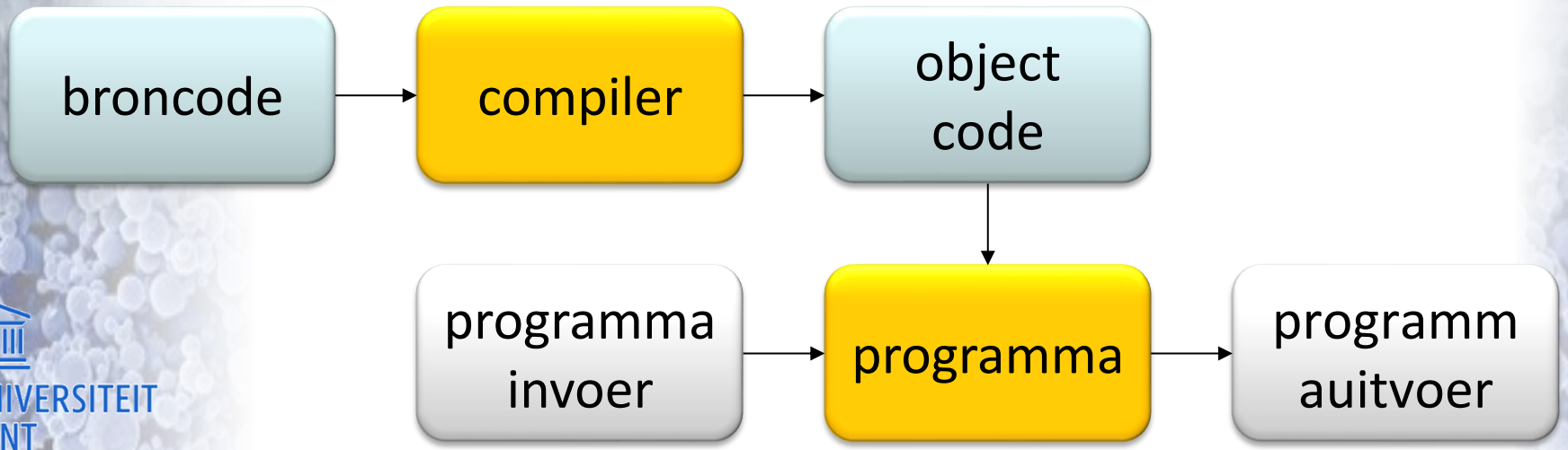
Programmeertalen

- voordeel van machinetaal
  - programma kan getuned worden voor specifieke hardware
    - minimale uitvoeringstijd
    - minimaal geheugengebruik
- voordelen van hogere programmeertaal
  - makkelijker te programmeren
    - minder tijd nodig om te schrijven
    - korter en makkelijker te lezen
    - meer kans om correct te zijn (minder bugs)
  - overdraagbaar naar verschillende hardware-platformen

vandaag de dag worden bijna alle programma's uitsluitend in hogere programmeertalen geschreven

Programmacode vertalen

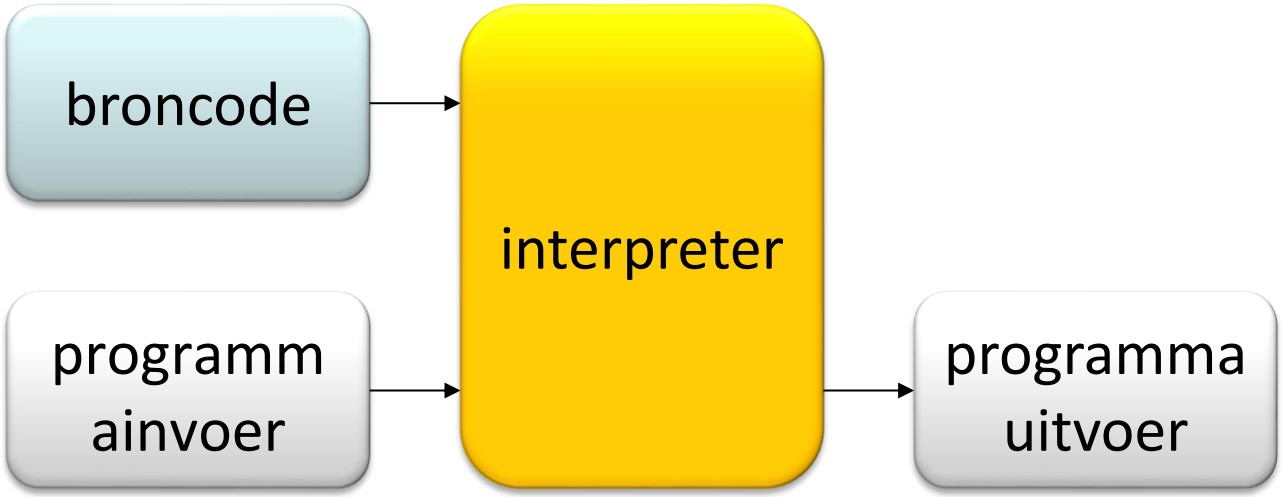
- scenario 1: gecompileerde talen (C, C++)
  - broncode vertaald naar machinetaal door compiler
  - opgeslagen als uitvoerbaar bestand (*object code*)
  - programma in geheugen geladen en uitgevoerd



UNIVERSITEIT GENT

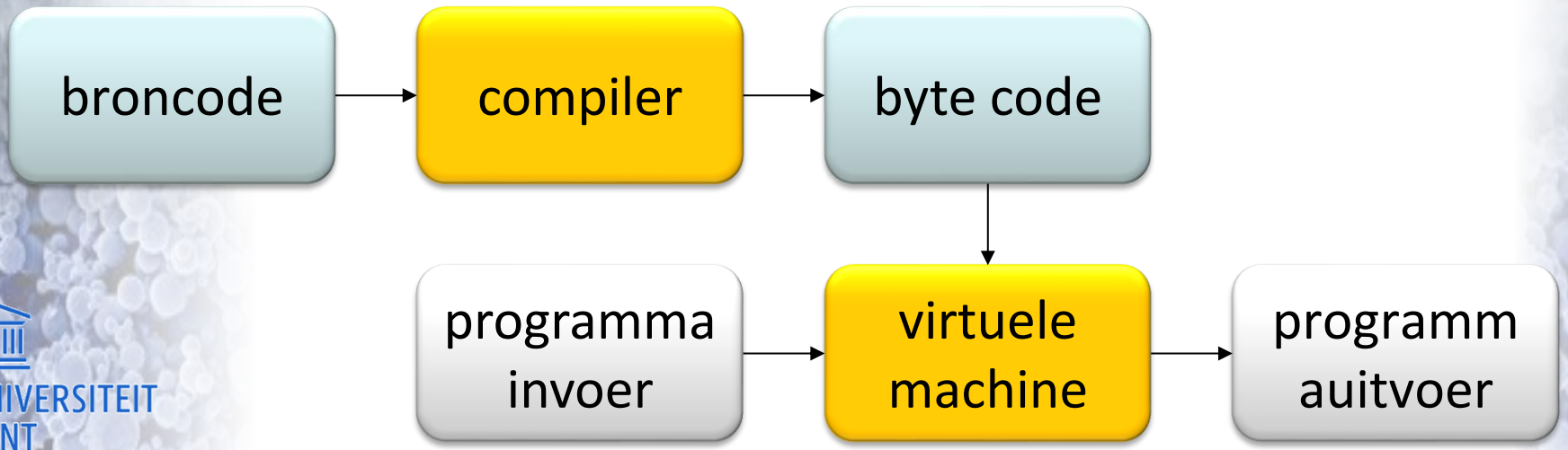
Programmacode vertalen

- scenario 2: geïnterpreteerde talen (Perl, Bash)
  - broncode vertaald naar machinetaal door interpreter
  - interpreter laat machinetaal onmiddellijk uitvoeren
  - dit gebeurt regel per regel



Programmacode vertalen

- scenario 3: hybride benadering (Python, Java)
  - broncode vertaald naar byte code door compiler
  - opgeslagen als "tussentijds" bestand
  - bij uitvoeren geïnterpreteerd door afzonderlijk programma
    - Python runtime, Java Virtual Machine (JVM)



Python runtime

- kan op twee manieren uitgevoerd worden
  - interactieve sessie (*shell mode*)
    - voer een Python-instructie in, en druk op Enter
    - resultaat wordt onmiddellijk uitgeschreven door interpreter

```
$ python3
Python 3.3.2 (default, May 16 2013, 00:03:43)
Type "help" for more information.
>>> print(1 + 1)
2
```

UNIVERSITEIT GENT

Python runtime

- kan op twee manieren uitgevoerd worden
  - interactieve sessie (*shell mode*)
    - voer een Python-instructie in, en druk op Enter
    - resultaat wordt onmiddellijk uitgeschreven door interpreter
  - niet-interactieve sessie (*script mode*)
    - sla reeks Python-instructies op in tekstbestand
    - laat tekstbestand uitvoeren door interpreter

```
$ cat eersteprogramma.py
print(1 + 1)
$ python3 eersteprogramma.py
2
```

UNIVERSITEIT GENT

Wat is een programma?

- reeks instructies die aangeven hoe een resultaat moet berekend worden

- voorbeelden
  - wiskunde
    - stelsel vergelijkingen oplossen
    - wortels van een veelterm vinden
  - symbolisch rekenen
    - tekst zoeken en vervangen in een document
    - broncode interpreteren (zoals Python runtime)

UNIVERSITEIT GENT

Wat is een instructie?

- verschillende programmeertalen
  → verschillende instructies (commando's, statements)
- gemeenschappelijke basisbewerkingen
  - invoer lezen
    - gegevens inlezen van toetsenbord, bestand, …
  - uitvoer wegschrijven
    - gegevens op scherm tonen, naar bestand schrijven, …
  - rekenen
    - wiskundige basisbewerkingen uitvoeren: optellen, aftrekken, …
  - voorwaardelijke opdrachten
    - acties enkel uitvoeren als bepaalde voorwaarden voldaan zijn
  - repetitieve opdrachten
    - acties herhaaldelijk uitvoeren, meestal met bepaalde variatie

Wat is programmeren?

Het proces waarbij grote, complexe opdrachten worden opgedeeld in steeds kleinere en kleinere deelopdrachten, totdat deze deelopdrachten uiteindelijk eenvoudig genoeg zijn om uitgevoerd te worden door de basisinstructies van een programmeertaal.

Wat is debuggen?

9/9

1000 " stopped - arctan ✓ 9.037 847 025
9.037 846 ?95 correct
13"uc (032) MP - MC ~~2.130476415 (-3)~~ 4.615925059(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 11.00 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.

1545 Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

1700 1630 Arctangent started.
closed down.

My code isn't working :-(

Start here...

Do you get an error when you run the code?

yes → **What type of error do you get?**

no → **Does the code use loops or if statements?**

neither

A variable that should contain a value does not
You are storing the return value of a function which changes the variable itself (e.g. sort)

A number which should be a fraction is coming out as zero in Python 2
You are dividing integers rather than floats. Convert the numbers to floats or from __future__ import division

I'm trying to print a value but getting a weird-looking string
You are printing an object (e.g. a FileObject) when you want the result of calling a method on the object

A regular expression is not matching when I expect it to
You have forgotten to use raw strings or escape backslash characters

I am reading a file but getting no input
You have already read the contents of the file earlier in the code, so the cursor is at the end.

loops

A list which should have a value for every iteration only has a single value
You have defined the list inside the loop: move it outside

A loop which uses the range function misses out the last value
The range function is exclusive at the finish: increase it by one.

I am trying to loop over a collection of strings, but am getting individual characters
You are iterating over a string by mistake

I am trying to write multiple lines to a file but only getting a single one
You have opened the file inside the loop: move it outside

if

Two numbers which should be equal are not
You are comparing a number with a string representation of a number (e.g. if 3 == "3")

A complex condition is not giving the expected result
The order of precedence in the condition is ambiguous - add some parentheses

also check...

Attribute Error
You are calling a method on the wrong type of object

SyntaxError
You've forgotten the quotes around a string

You have forgotten to put a colon at the end of a def/if/for line

You have different number of open and close brackets in a statement

TypeError
You're trying to use an operator on the wrong type of objects

An object which you expect to have a value is actually None

You've used non-integer numbers in a list slice

You've called a method/ function with the wrong number or type of arguments

Indentation Error
You've used a mixture of tabs and spaces

You haven't indented all lines in a block equally

NameError
You've misspelt a variable, function or method name

You've forgotten to import a module

You've forgotten to define a variable

Your code uses a variable outside the scope where it's defined

Your code calls a function before it's defined

You're trying to print a single word and have forgotten the quotes

IOError
You're trying to open a file that doesn't exist

KeyError
You're trying to look up a key that doesn't exist in a dict

http://pythonforbiologists.com

Wat is debuggen?

drie soorten programmeerfouten

- compileerfout (syntax error)
- fout tijdens het uitvoeren (run-time error)
- logische fout (semantic error)

Wat is debuggen?

drie soorten programmeerfouten

- compileerfout (syntax error)
  - interpreter voert enkel syntactisch correcte programma's uit
  - anders geeft interpreter foutboodschap terug
  - syntaxis verwijst naar grammaticaregels van programmeertaal
- voorbeeld in natuurlijke taal
  - grammaticaregel: zin begint met hoofdletter en eindigt op punt
    - "deze zin bevat een grammaticale fout."
    - "En deze zin ook"

Wat is debuggen?

drie soorten programmeerfouten

- compileerfout (syntax error)
  - interpreter voert enkel syntactisch correcte programma's uit
  - anders geeft interpreter foutboodschap terug
  - syntaxis verwijst naar grammaticaregels van programmeertaal
- Python voorbeeld
  - niet-gebalanceerde haakjes

```
>>> print(5 / 2))
  File "<stdin>", line 1
    print(5 / 2))
                ^
SyntaxError: invalid syntax
```

Wat is debuggen?

drie soorten programmeerfouten

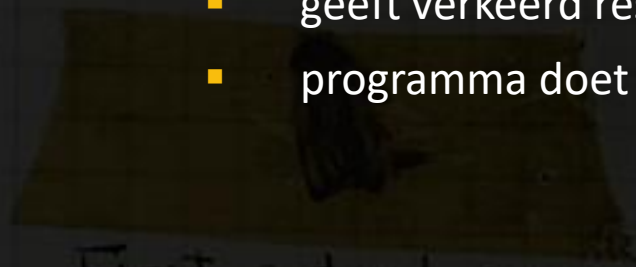
- fout tijdens het uitvoeren (run-time error)
  - doet zich pas voor tijdens uitvoeren van foutieve instructie
  - exception genoemd: uitzonderlijke situatie (negatief)
  - uitvoering van programma wordt normaalgezien beëindigd
  - omschrijving van uitzonderlijke situatie wordt weergegeven
- Python voorbeeld
  - delen door nul

```
>>> 5 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
```

Wat is debuggen?

drie soorten programmeerfouten

- logische fout (semantic error)
  - programma wordt zonder problemen uitgevoerd
    - geen foutboodschap uitgeschreven
  - programma gedraagt zich niet zoals verwacht
    - geeft verkeerd resultaat terug
    - programma doet enkel wat het is opgedragen !!



Oplossen van logische fouten vereist dat je op basis van het waargenomen resultaat (indien aanwezig) probeert te achterhalen wat het programma intern aan het doen is.

Experimenteel debuggen

Experimenteel debuggen

- debuggen vormt één van de belangrijkste vaardigheden in het programmeerproces
  - sommige programmeurs zijn van mening dat debuggen één van de intellectueel meest verrijkende, uitdagende, en interessante onderdelen van het programmeren is
  - sommige studenten voegen daar ook het adjectief frustrerend aan toe

Experimenteel debuggen

- debuggen is een beetje zoals detectivewerk
  - je wordt geconfronteerd met aanwijzingen
  - je moet proberen achterhalen welke processen en gebeurtenissen tot het resultaat geleid hebben

"When you have eliminated the impossible, whatever remains, however improbable, must be the truth"

— Sherlock Holmes
uit The Sign of Four (Arthur Conan Doyle)

Experimenteel debuggen

- debuggen lijkt ook op experimenteel onderzoek
  - formuleer een hypothese over de oorzaak van de fout
  - wijzig het programma en voorspel nieuw resultaat
  - gestelde hypothese was correct als het resultaat overeenkomt met de voorspelling
  - anders moet de hypothese bijgesteld worden

Formele vs natuurlijke talen

- natuurlijke taal
  - wordt door mensen gesproken: Nederlands, Engels, Frans
  - niet door mensen ontwikkeld maar spontaan geëvolueerd
- formele taal
  - ontwikkeld door mensen voor specifieke toepassing
    - wiskunde
      - wiskundige notatie: formele taal die relatie tussen getallen en symbolen vastlegt
    - chemie
      - chemische notatie: formele taal die chemische structuur van moleculen voorstelt
    - informatica
      - programmeertaal: formele taal voor uitdrukking van berekeningen

UNIVERSITEIT GENT

Formele vs natuurlijke talen

- natuurlijke taal
  - wordt door mensen gesproken: Nederlands, Engels, Frans
  - niet door mensen ontwikkeld maar spontaan geëvolueerd
- formele taal
  - heeft doorgaans zeer strikte regels voor syntaxis
    - **3+3=6** is een syntactisch correcte wiskundige uitdrukking
    - **3=+6$** is dat niet
    - **$H_2$0** is een syntactisch correcte chemische naam
    - **$_2$Zz** is dat niet

UNIVERSITEIT GENT

Formele talen

- twee soorten syntaxisregels
  - tokenregels
    - tokens zijn de basiselementen van een formele taal
      - woorden
      - getallen
      - chemische elementen
    - in **3=+6$** is **$** geen geldig token in de wiskunde
    - in $_2$**Zz** is er geen chemisch element met symbolische naam **Zz**
  - structuurregels
    - volgorde van tokens in een statement
    - in **3=+6$** kan plusteken niet volgen op gelijkteken
    - in $_2$**Zz** moet subscript na naam van element staan, niet ervoor

UNIVERSITEIT GENT

Formele talen

- twee soorten syntaxisregels
  - tokenregels
  - structuurregels
- proces waarbij structuur van zin in formele of natuurlijke taal bepaald wordt, noemt men *parsen*
  - voorbeeld: "Zijn frank viel."
    - "zijn frank" is het onderwerp
    - "viel" is het werkwoord
- na parsen van zin kan betekenis ervan bepaald worden

Formele vs natuurlijke talen

- formele en natuurlijke talen delen
  - woorden
  - structuur
  - syntaxis
  - betekenis
- verschil tussen formele en natuurlijke talen is vergelijkbaar met verschil tussen proza en poëzie

UNIVERSITEIT GENT

Formele vs natuurlijke talen

- formele en natuurlijke talen verschillen in
  - dubbelzinnigheid
    - natuurlijke talen zitten vol dubbelzinnigheid; menselijk brein gebruikt contextuele en andere informatie om betekenis te achterhalen
    - formele talen zijn ontworpen om ondubbelzinnig te zijn; elke instructie heeft exact één betekenis
  - overtolligheid
    - natuurlijke talen bevatten overtolligheid, net om misverstanden door dubbelzinnigheid uit de weg te gaan
    - natuurlijke talen zijn daardoor breedsprakerig
    - formele talen zijn minder redundant en beknopter
  - letterlijkheid
    - natuurlijke talen zitten vol gezegden en metaforen: er is in werkelijkheid geen frank gevallen
    - formele talen betekenen letterlijk wat er gezegd wordt

UNIVERSITEIT GENT

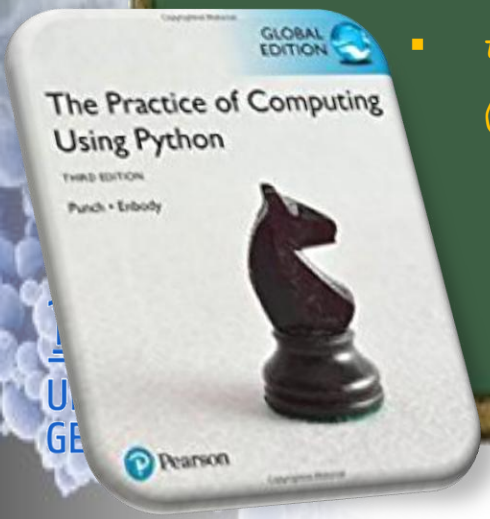
Hello, World !


```
$ cat HelloWorld.py
print('Hello, World!')
$ python3 HelloWorld.py
Hello, World !
```

UNIVERSITEIT GENT

Huiswerk

- handboek: lees hoofdstukken 0 en 1
  - als voorbereiding op eerste werkcollege
  - Punch W & Enbody R (2017, global edition)
    The practice of computing using Python. Addison-Wesley
    ISBN-13: 978-1-29216-662-9
    - PDF beschikbaar op Ufora
    - tweede en derde editie van handboek ook nog bruikbaar (behandelt ook al Python 3.x)



Huiswerk

- videohandleidingen
  - eerste Python programma in PyCharm
  - interactieve Python sessies in PyCharm
- bekijk praktische informatie
  - resterende slides
  - Ufora
  - Dodona

Huiswerk

- afwerken opgelegde oefeningen reeks 01 (deadline dinsdag 30 september 2025, 22:00)
  - ISBN (demo, video)
  - Som van twee getallen
  - Examens verbeteren
  - The pudding guy
  - Ontwikkelingsindex
  - De gestopte klok

Werkvormen

- begeleide zelfstudie
  - gedetailleerde planning op de Dodona-cursus
  - nalezen handboek: must als voorbereiding op hoorcollege (inwerken)
  - extra voorbereidingstaken: must als voorbereiding op oplossen oefeningen
  - opgelegde oefeningen: must op te leren programmeren
  - probeer zoveel mogelijk oefeningen af te werken voor het werkcollege
  - stel vragen via Dodona, Ufora forum of tijdens hoor- en werkcolleges
  - goede voorbereiding is beste manier om te leren programmeren

UNIVERSITEIT GENT

Werkvormen

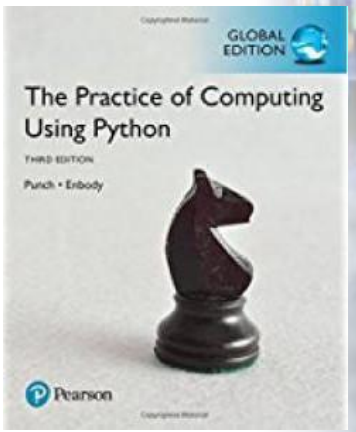
- hoorcolleges
  - programmeertechnieken uitgelegd via uitwerken van voorbeelden
  - voorbereiding noodzakelijk: lezen handboek, voorbereiden lesopgaven
  - gelegenheid tot vragen stellen !!
  - kom op tijd en schakel je GSM uit

- werkcolleges
  - los opgelegde programmeeroefeningen op in de PC-klas
  - gebruik Python handboek en documentatie als referentiemateriaal
  - gebruik je eigen laptop om oefeningen op te lossen (BYOD)
  - gelegenheid tot vragen stellen aan medestudenten en begeleiders
  - samenwerking tussen studenten toegelaten (max 3 studenten per groep)

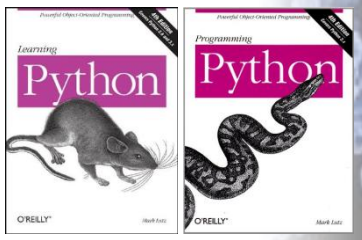
UNIVERSITEIT GENT

Leermateriaal

- handboek
  - Punch W & Enbody R (global edition, 2017). The practice of computing using Python. Addison-Wesley, 978-1-29216-662-9.



- referentieboeken
  - Mark Lutz (2009). Learning Python: Powerful Object-Oriented Programming (4th edition). O'Reilly Media, 978-0596158064.
  - Mark Lutz (2011). Programming Python (4th edition). O'Reilly Media, 978-0596158101.



UNIVERSITEIT GENT

Extra leermateriaal

- powerpoint presentaties uit lessen
  - beschikbaar gesteld via Ufora
  - gelieve fouten of onduidelijkheden te melden
  - bijgewerkte versies van presentaties kunnen oude vervangen
- achtergrondmateriaal
  - bijkomende documenten
  - beschikbaar gesteld via Ufora
- naslagwerken
  - help-functie in interactieve sessie (ingebouwde documentatie)
  - The Python Tutorial
  - The Python Standard Library

UNIVERSITEIT GENT

Contacturen

- hoorcolleges
  - sluit aan bij één van deze twee sessies
    - initiële toewijzing in TimeEdit op basis van jouw opleiding
    - *feel free* om bij andere sessie aan te sluiten (toestemming vragen niet nodig)
    - beschikbaar voor alle hoorcolleges: livestream + lesopname (zie Ufora agenda)
  - Nederlandstalig traject
    - maandag 11:30 – 12:45 (week 1-4)
      - Campus Sterre, gebouw S9, auditorium A2 (tweede verdieping)
    - donderdag 08:30 – 09:45 (week 1-12)
      - Campus Sterre, gebouw S9, auditorium A0 (gelijkvloers)
  - Engelstalig traject
    - maandag 14:30 – 17:15 (week 1-12)
      - online (MS Teams)

Contacturen

- werkcolleges
  - sluit aan bij één van deze twee sessies
    - initiële toewijzing in TimeEdit op basis van jouw opleiding
    - *feel free* om bij andere sessie aan te sluiten (toestemming vragen niet nodig)
  - vrijdag 10:00 – 12:15 (week 1-12)
    - Campus Boekentoren, lokaal 0.9 (gelijkvloers & verdieping -1)
    - opleidingen: biologie, BCBT, geografie, pharmaceutical engineering
  - vrijdag 14:30 – 17:15 (week 1-12)
    - Campus Boekentoren, lokaal 0.9 (gelijkvloers & verdieping -1)
    - opleidingen: chemie, fysica & sterrenkunde, geologie, statistical data analysis

UNIVERSITEIT GENT

Studiebegeleiding

- alle "ad valvas" aankondigingen via Ufora
- stel vragen en geef opmerkingen
  - tijdens de hoor- en werkcolleges
  - via forum op Ufora
  - onmiddellijk na hoorcolleges
  - tijdens een persoonlijke afspraak (vastgelegd via e-mail)
    - verstuur e-mails niet vanaf anoniem account, maar onder je eigen naam
    - gebruik verplicht je UGent e-mailadres
    - vermeld steeds je naam, opleiding en betreffende opleidingsonderdeel
    - vermijd typfouten, dt-fouten, SMS-taal ()
    - hanteer duidelijk taalgebruik en correcte aanspreektitel

UNIVERSITEIT GENT

Studiebegeleiding: hoorcolleges

- verantwoordelijke lesgever
  - prof. dr. Peter Dawyndt
    - vakgroep wiskunde, informatica en statistiek
    - bureau: Campus Sterre, Krijgslaan 281, S9
    - telefoon: +32 9 264 47 79
    - email: peter.dawyndt@ugent.be
  - onderzoeksdomeinen
    - computationele biologie (bioinformatica)
    - computer science education

UNIVERSITEIT GENT

Studiebegeleiding: werkcolleges

Maarten Stevens
maarten.stevens@ugent.be

Thomas Van Mullem
thomas.vanmullem@ugent.be

Rien Maertens
rien.maertens@ugent.be

Mustapha Regragui
mustapha.regragui@ugent.be

Vincent Batens
vincent.batens@ugent.be

Emma Vandewalle
emmavdwa.vandewalle@ugent.be

Tibo De Peuter
tibo.depeuter@ugent.be

Wouter De Bolle
wouter.debolle@ugent.be

UNIVERSITEIT GENT

Evaluaties

- opgelegde oefeningen
  - omzetten van theorie in praktijk
  - wekelijkse deadlines
    - reeks 01 – reeks 05: dinsdag 22:00
    - reeks 06 – reeks 10: dinsdag 22:00
- evaluaties
  - twee oefeningen oplossen binnen twee uur
  - simulatie van examen
  - lagere moeilijkheidsgraad dan op examen: testen basisvaardigheden
- andere opgaven
  - vrijblijvend
  - leren programmeren = oefenen, oefenen, oefenen, …

UNIVERSITEIT GENT

Evaluaties

- permanente evaluaties tellen mee voor 4/20
  - driemaal winst !!!
  - niet herneembaar tijdens tweede zittijd
- score voor evaluatie houdt rekening met aantal opgelegde oefeningen die opgelost werden voor wekelijkse deadlines
- beoordeling
  - oplossingen van opgelegde oefeningen voor wekelijkse deadlines
  - oplossingen van twee nieuwe oefeningen tijdens evaluatie
  - feedback
    - voorbeeldoplossingen gepubliceerd na wekelijkse deadlines
    - punten en persoonlijke feedback gepubliceerd kort na evaluatie

UNIVERSITEIT GENT

Vragen of opmerkingen ?

UNIVERSITEIT GENT

The sky is the limit …

"Whatever you thought, think again"

— National Geographic Magazine