







Guide to GrandeOmega (GO)

Installation

- Download node.js (LTS) from: <https://nodejs.org/en/download/>
- Download Python from: <https://www.python.org/downloads/>
 - For Windows users, make sure to have Python added to your environment variables (<https://www.architectryan.com/2018/03/17/add-to-the-path-on-windows-10/>)
- Download the client of GO from: http://grandeomega.com/go_student_win.7z
 - Note: the file is archived with 7zip (download it, if needed)
- Unzip the compressed folder downloaded at the previous step
- Execute the **GrandeOmega.exe** file:

	d3dcompiler_47.dll	9/7/2018 11:16 AM	Application extension	4,077 KB
	ffmpeg.dll	9/7/2018 11:16 AM	Application extension	1,910 KB
	GrandeOmega.exe	9/7/2018 11:19 AM	Application	66,003 KB
	icudtl.dat	9/7/2018 11:16 AM	DAT File	9,959 KB
	libEGL.dll	9/7/2018 11:16 AM	Application extension	18 KB
	libGLESv2.dll	9/7/2018 11:16 AM	Application extension	3,602 KB

Use

- After the client starts, you need to login with your credentials (you will receive via your student email instructions to get access):

Email	<input type="text"/>	Password	<input type="password"/>	<input type="button" value="Login"/>
-------	----------------------	----------	--------------------------	--------------------------------------

- After having logged in, you will see a screen with the courses you are subscribed to:

Welcome (student), abbam@hr.nl

Logout

Courses

Development 1

Basic imperative computational concepts and flow control.

Development 6A

Introduction to algorithms.

Development 5

Web development: from ORM's to API's to single page applications.

Guide to GrandeOmega

An introductory guide to GrandeOmega.

Development 2

Functions, lambda expressions, recursion, higher order functions.

Development 6B

Advanced databases

Software engineering 1

Functors, monads, and other advanced data structures in practice.

Development 3

Static typing, object orientation, subtyping polymorphism

Development 4

Polymorphism, exception-handling, and basic design patterns

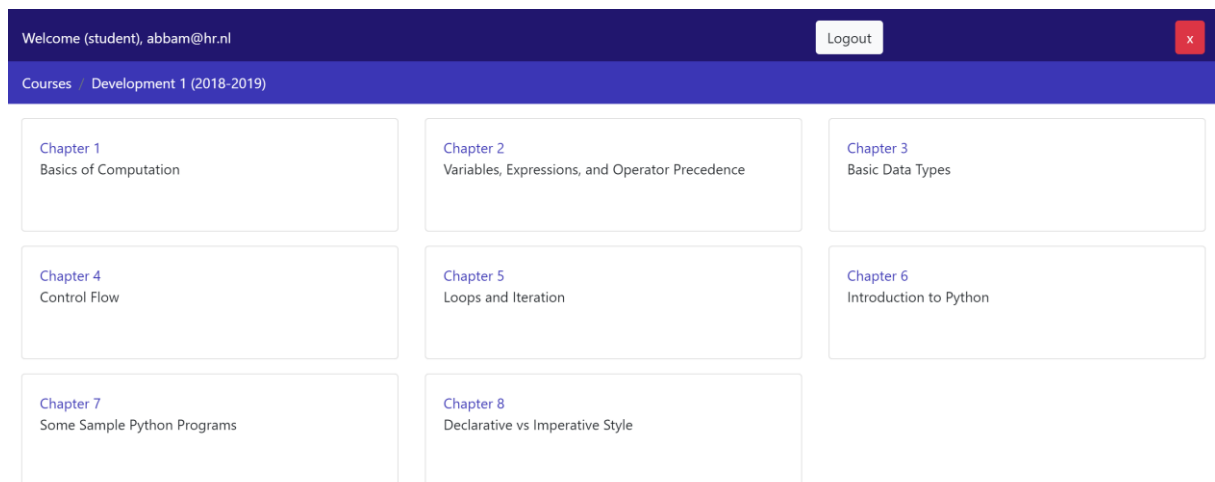
Software engineering 2

A categorical perspective on functors and monads via Bi-Cartesian Closed Categories.

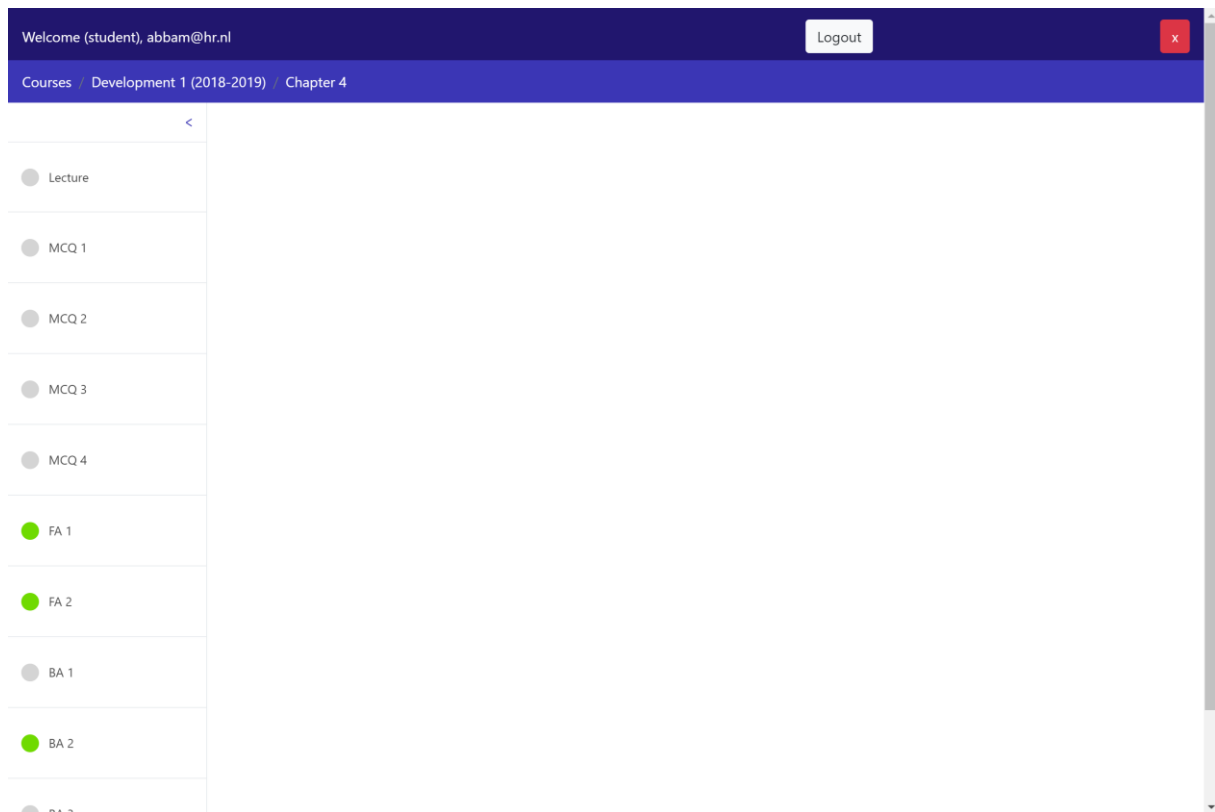
Development 1 (2018-2019)

Basic imperative computational concepts and flow control.

- Clicking on a course, you will see the chapters of materials available for such course:



- Clicking on a chapter, you will see the materials associated to such chapter in a column on the left of the screen. Click on the name of an item to open its associated content.



- A single chapter is usually composed by:
 - The reader of the corresponding lecture
 - A series of exercises which are a combination of:
 - Multiple Choice Questions (MCQ)
 - Forward Assignments (FA)
 - Backward Assignments (BA)
- During the practicums, the teachers will show you more in detail how to solve the Forward and Backward assignments.

- In short, a **Forward Assignment** shows you a program and the (sometimes incomplete) state associated to certain steps of the execution of such program (marked with red blocks to the left of the code). To solve a FA, you need to insert the missing values of variables in *all* incomplete states (remember to click “Next” until the last state is reached). For example:

```

1  x=5
2  y=1
3  x=3
4  y=x
5

```

Reset

Globals

x	
---	--

Heap

Stack

Prev
Next

The state on the right (Globals, etc.) corresponds to the state of the program when the line of code marked with a yellow block is about to be executed (in the example above, when line 2 is about to be executed).

A **Backward assignment**, instead, shows you an incomplete program and the states associated to some steps of the execution of the complete program (again, marked by red/yellow blocks to the left of the code). By looking at such states, you should be able to fill in the missing parts of the program. For example:

```

1  a = 
2  b = 1
3  = 2
4

```

Validate
Cancel validation

Globals

a	0
b	1
c	2

Heap

Stack

0
1
2
3

To see if your code solves the BA, click on “Validate” and you will get feedback.

When an assignment is correctly solved (both FA and BA) a “Success!” green message will appear on screen:

```

1  a = 
2  b = 1
3  = 2
4

```

Validate
Cancel validation

Globals

a	0
b	1
c	2

Heap

Stack

0
1
2
3

Success!

Otherwise, a “Wrong!” red message appears (and in BAs the wrong values of your program are shown in red close to the correct ones in green in the state):

The screenshot shows a web application interface. At the top, a dark blue header bar contains the text "Welcome (student, abbam@hr.nl)" and a "Logout" button. Below the header, a pink box displays the message "Wrong!". The main content area is divided into two columns. The left column contains a sidebar with a list of items: "Lesson", "Exercises", and "MCQ 1". The "Lesson" item is highlighted with a red background, and the "Exercises" item is highlighted with a yellow background. The right column displays a state diagram with three nodes labeled "a", "b", and "c". Node "a" has a value of 1, node "b" has a value of 1, and node "c" has a value of 2. Below the state diagram, there are two buttons: "Validate" and "Cancel validation". On the right side of the interface, there is a "Globals" table with two columns, "0" and "1". The "0" column has a green background, and the "1" column has a red background. The table contains the following data:

	0	1
a	1	1
b	1	1
c	2	2

Below the "Globals" table, there is a "Heap" section and a "Stack" section. The "Stack" section contains a row of four colored boxes: green (0), red (1), red (2), and blue (3).

The round icon close to the assignment name in the left column also gets such color (orange for incomplete/wrong and green for complete):

The diagram shows two assignment status icons. The first icon is a green circle next to the text "FA 5". The second icon is an orange circle next to the text "BA 1".