HKU

Empowering Creative Thinking Through Programming

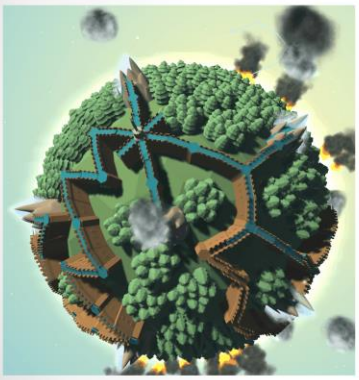
Valentijn Muijrers

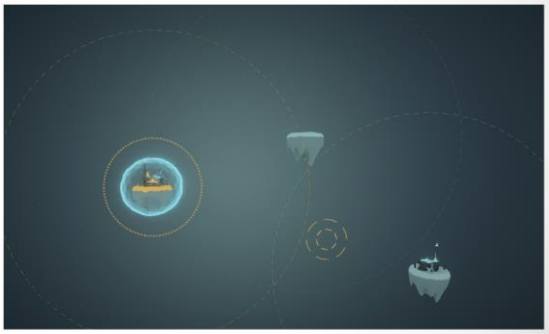
https://github.com/vmuijrers/ECTTP

Wie ben ik

- Valentijn Muijrers (29)
- 5 jaar HKU teacher in programming courses
- I make games!
- Game Jams and AI

My Games





You can play them here:
www.huntedwumpus.nl/Aquaduct/index.html

www.huntedwumpus.nl/TinyWorlds/index.html

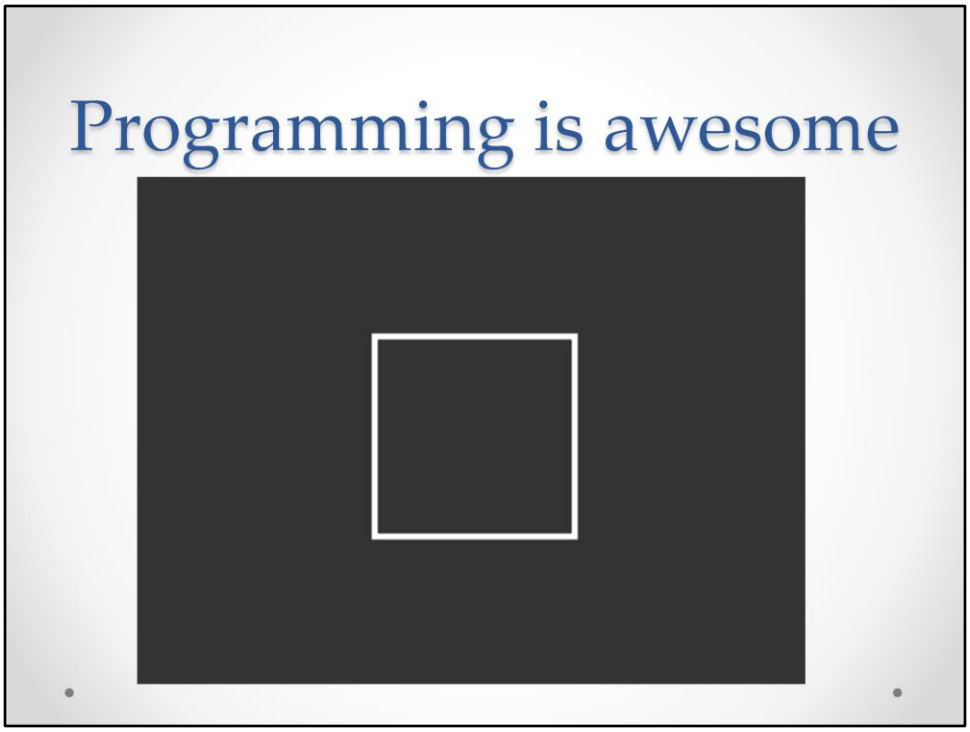
Course Overview

- Week One: Course overview
- Week Two: Variables
- Week Three: Operators
- Week Four: Conditions
- Week Five: Loops
- Week Six: Functions
- Week Seven: **Toets 1**
- Week Eight: Feedback Test
- **End of block 1**
- Week Eleven: Lists
- Week Twelve: Classes and Objects
- Week Thirteen: Preparation for Toets 2
- Week Fourteen: **Toets 2**

Deliverables

- Every week there is a lab (there will be 10 in total)!
- 100% presence
- 2 tests (individually!)

- Grade:
- 40% labs
- 20% test 1
- 40% test 2
- To pass the course, T1 and T2 have to be >= 5.5 and overall average has to be >= 5.5



Example! Can you make a squircle?

Can you follow instructions?

- Rule 1: Only execute rules, when Valentijn says "Go"
- Rule 2: When you read "Kipje", clap your hands once
- Rule 3: When you read "Koei", clap hands with your neigbour(s) once
- Rule 4: When you read "Marsupilami", wait 2 seconds then execute "Kipje" followed by "Koei"

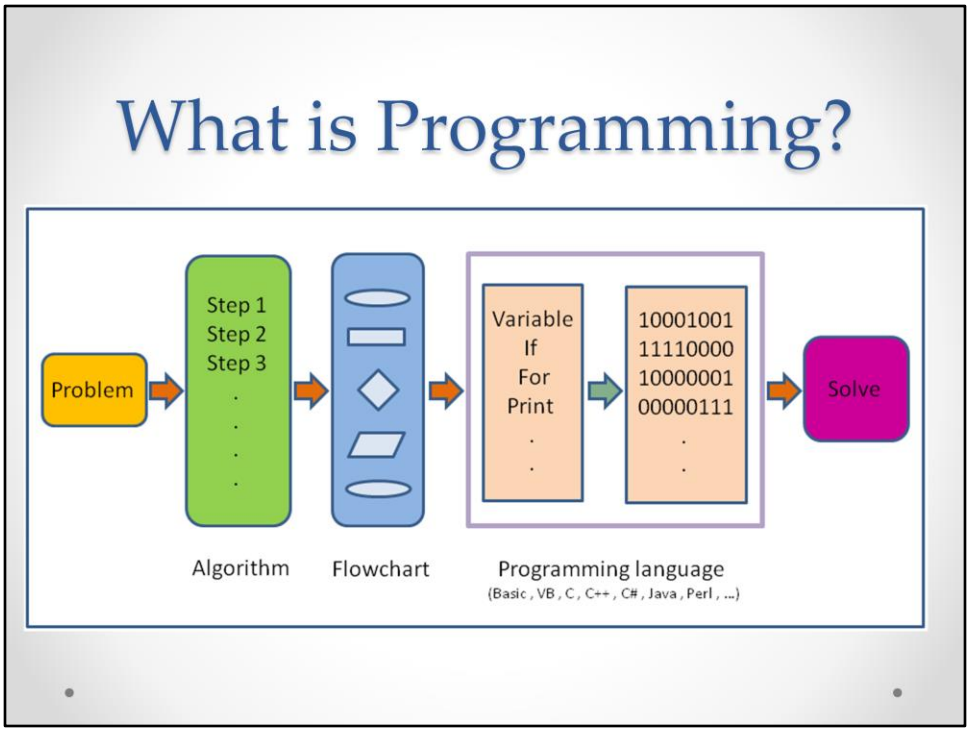
Execute:
Kipje, Kipje, Kipje;
Kipje, Kipje, Koei;
Marsupilami;
while(Valentijn's hand up):
    Marsupilami;

Super Power

- With great power comes great responsibility

Adhere to the code of conduct:
- Do not steal code!
- Do not use code for evil!
- Be nice to your computer, it just executes the code you give it, nothing else (in most cases)

Adhere to the Code of conduct. Do not steal code! Do not use code for evil! Also, be nice to your computer, it just executes the code you give it, nothing else (in most cases).

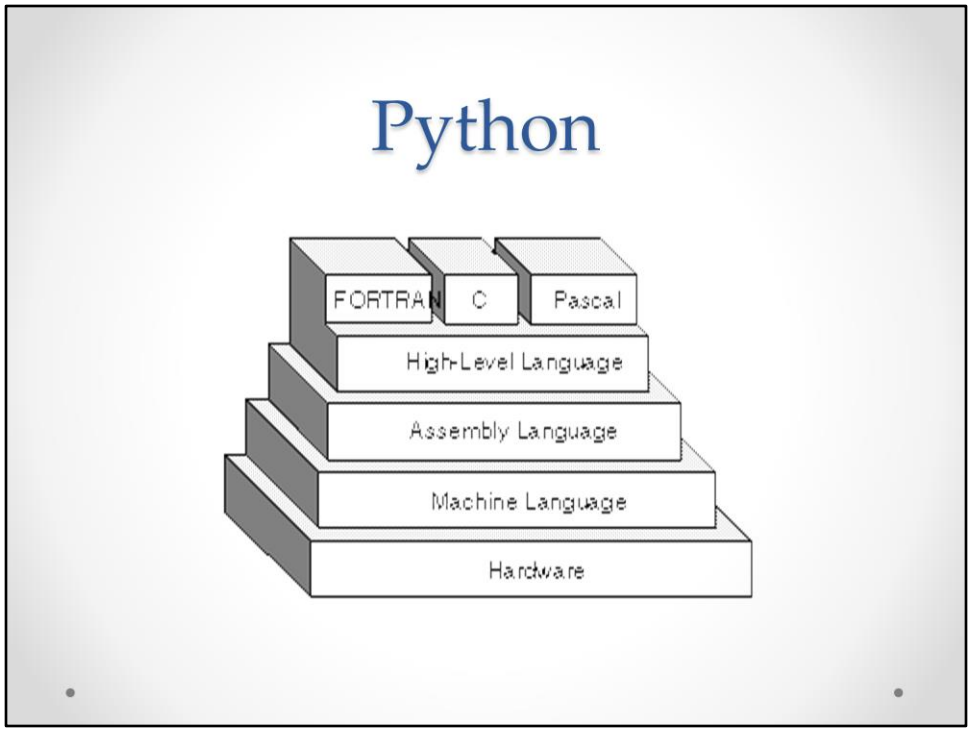


Problem solving means the ability to formulate problems, think creatively about solutions, and express a solution clearly and accurately

How you solve the problem is called an algorithm

An algorithm is a step by step list of instructions, or a program , that if followed exactly will solve the problem under consideration

These algorithms form programs which are written in programming languages (such as Python, Java, C++ or C# and many others!)



Python is an example of a high-level language

There are also low level programming languages (assembly, machine language)

High level languages use human friendly sets of instructions to create programs.
Low level languages are not as human friendly and are written for the hardware.

High vs Low

High level language:

- print("Hello")

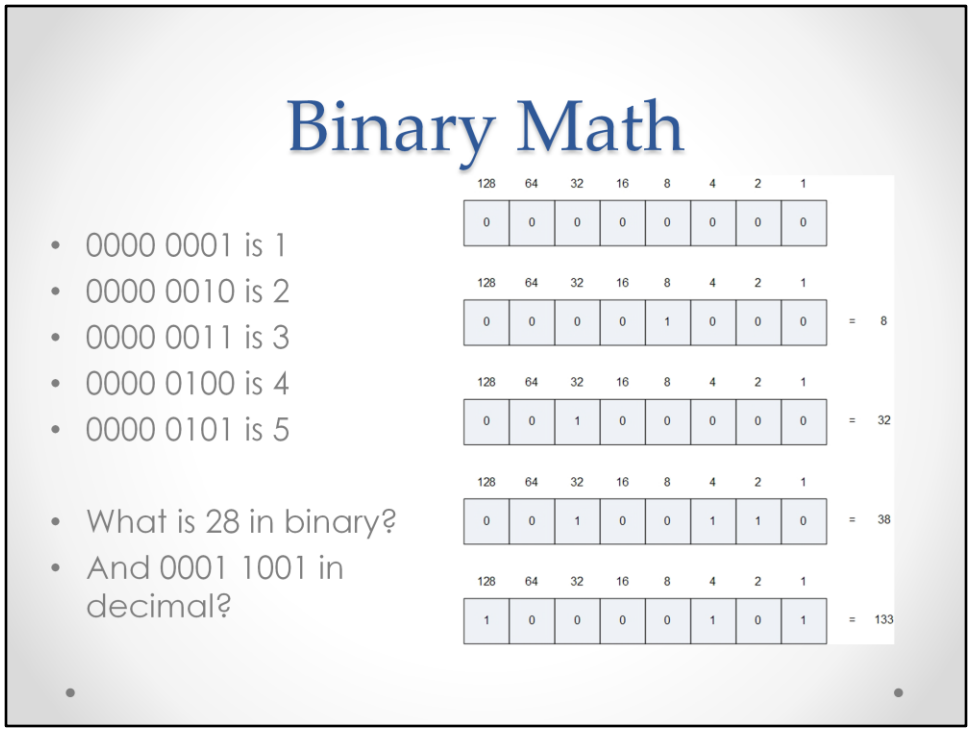
Low level language:

- 01010101 00011101 00111100

- High level is less prone to error

Binary

- All low level languages are in binary.
- Binary consists of1's and 0's which are called **bits**
- Series of bits form 8-bit patterns (called a byte) which represent numeric values.
- Each bit represents either on or off. You can think of them like little switches.



Each bit (starting from the right), represents a power of 2. If the bit is 1 the power of 2 can be added, if the bit is 0, the power of 2 can be ignored.

The number 133 is 10000101 in binary because 2^0 + 2^2 + 2^7 = 133 (note here that the most right bit is at position 0)

Ascii

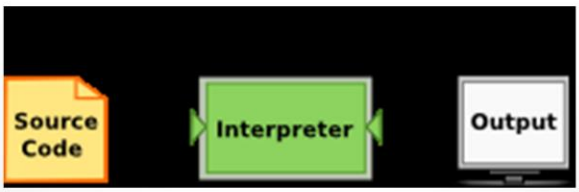
- Ascii is a way to translate letters into numbers
-

ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

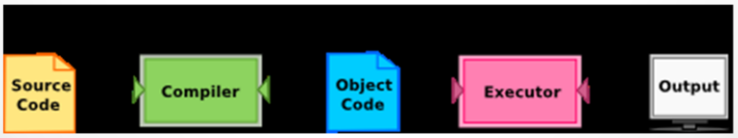
Interpreters

- An interpreter reads a high-level program and executes it, meaning that it does what the program says.
- It processes the program a little at a time, alternately reading lines and performing computations.

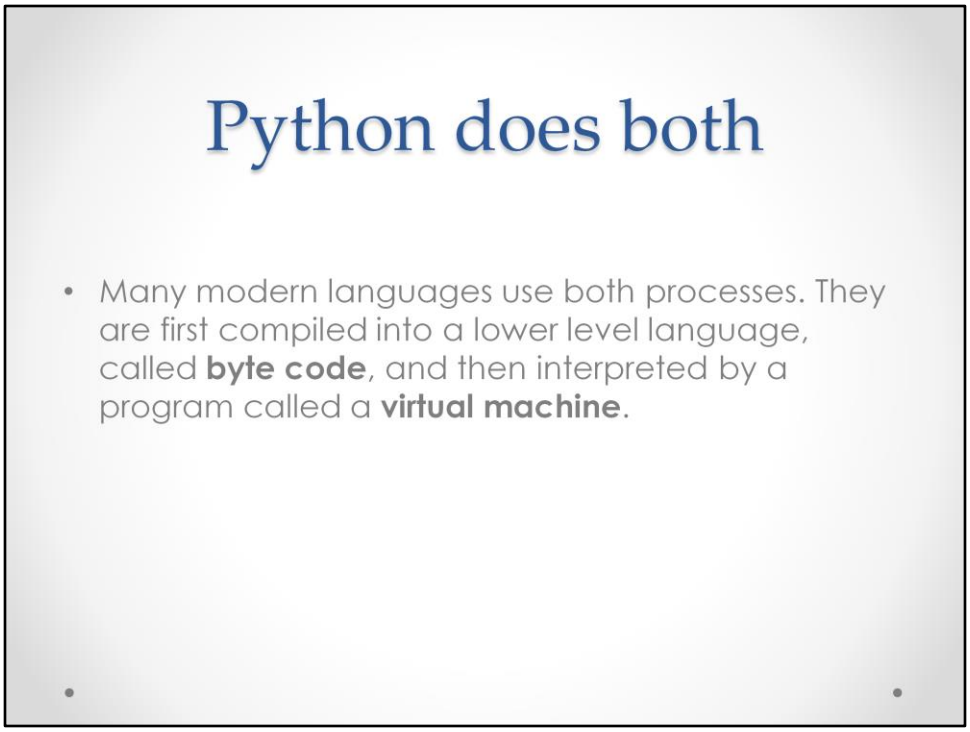


Compiler

- A compiler reads the program and translates it completely before the program starts running. In this case, the high– level program is called the **Source code**, and the translated program is called the **object code** or the **executable**. Once a program is compiled, you can execute it repeatedly without further translation.

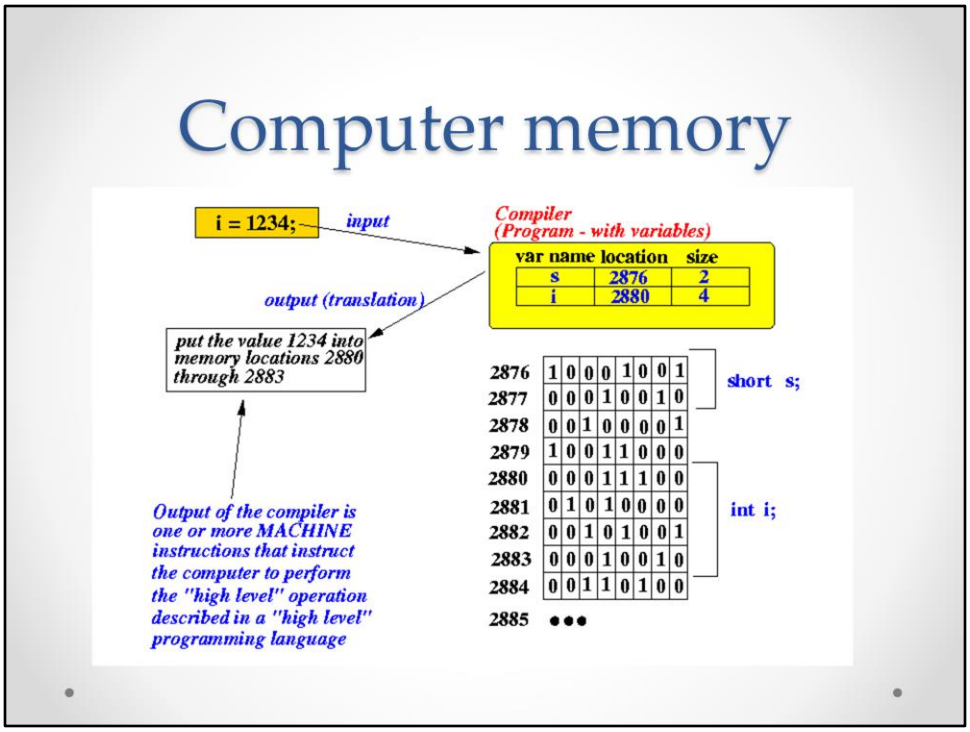


Python is uses both an interpreter and a compiler



Python uses both processes, but because of the way programmers interact with it, it is usually considered an interpreted language.

There are two ways to use the Python interpreter: *shell mode* and *program mode*. In shell mode, you type Python expressions into the **Python shell**. In program mode you type them into a file.



A program lives on your hard disk

The instructions of a program get loaded into ram. The machine code is fetched into your CPU

These instructions get decoded and then executed on your CPU

How does a program work?

- A program executes lines of code in order (the order matters)
- Each line is a **statement** which is first evaluated and then executed
- A statement does something (print something, assign a new value to a variables etc.)

- A program can consist of thousands of statements

Variables

- In order to write a program, we need to use our computer's memory to store data
- Data can be stored in **variables**
- **Variables** can be reused throughout the rest of the program

Input Variables and Output

x = 10

'x' is the variable name

'=' is the assignment operator and sets the value on the left of the equation to be equal to the value on the right

'10' is the value which is an integer number or a whole number

Variables are types of data stored in containers. Think of them like a box that holds something in it.

Data types

- Basic types:

- Integer: 10, 125, -12408 , 0
- Float: 10.0, -4.2123, 0.124
- Boolean: True or False (0 or 1)
- String: "a set of words or characters in quotes"

Naming Variables

- A variable can have **any** name but…
- Don't use spaces in your naming
- Don't use special characters: &!$*+-~,/\%^()
- Don't use numbers as the first letter of your name (recommended: don't use numbers in your names at all)

Other reserved words in Python which you cannot use as variable names:

False True as finally is return None continue for lambda try
Def from nonlocal while and del global not with
As elif if or yield assert else import pass break except in raise

Some Examples

Right:

- myIntegerVar = 10
- myStringVar = "Hello"
- myFloatVar = 1.0

Wrong:

My Integer = 10 #Do not use spaces in your variable names

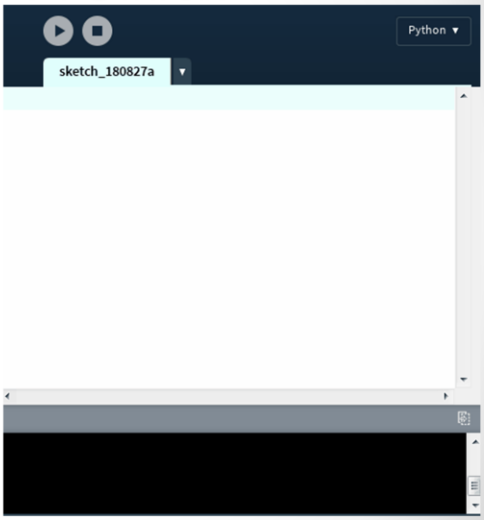
34MyInt = 5 # Do not use numbers at the start of a variable

Print Something

- Print is a function in python. A function is a set of instructions that does something. You sometimes put things into it. Sometimes it returns a value.
- Functions are like little machines
- I don't really care how a coffee pot works. All I know is if I put water and coffee grinds into it, it makes coffee and I can take coffee out of it

Hello World!

- Try typing the following in Processing:
- print("Hello World!")
- And press Play, below in the Console, you will see your output stating the words "Hello World!"
- The Console only shows feedback from your program when it runs. It shows errors (mistakes in your program) and all of your print- calls.



Where to start?

- Download Processing 3.4:
- https://processing.org/download/?processing

For all functionality of python:
https://docs.python.org/3.5/tutorial/index.html

For all functionality of processing:
http://py.processing.org/

First lab is online

- https://github.com/vmuijrers/ECTTP/blob/master/Labs/Lab_1.md
- Read carefully what it says on the lab page and be sure to hand in your lab in the right way and on time!
- Deadline is at the end of the lab!