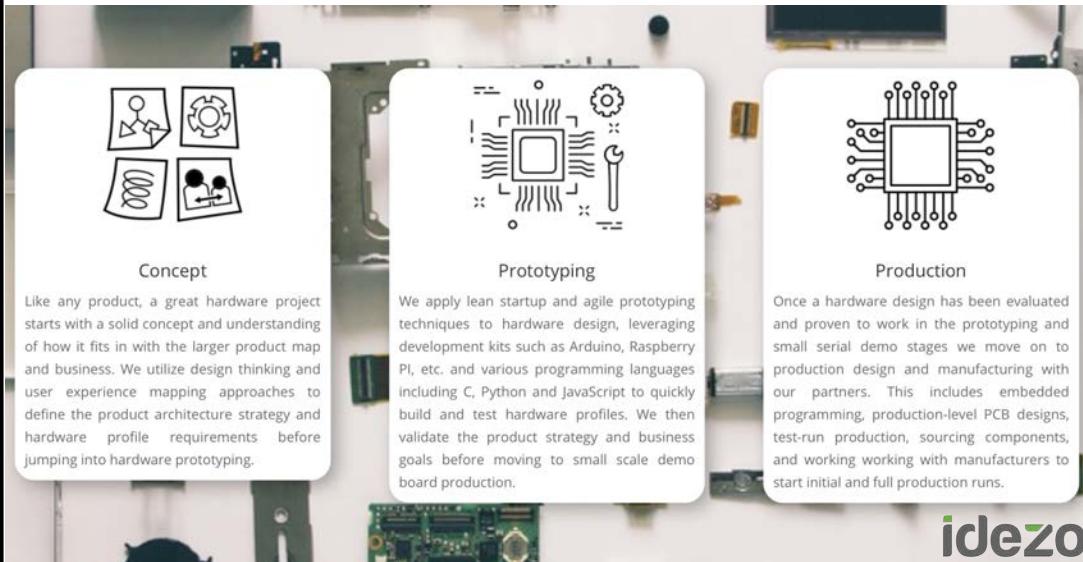


HANDS ON APPROACH TO

Data Science

for (the)

IoT



Rob van der Willigen

<https://github.com/robvdw/CMIDAT01K-DATA-SCIENCE-for-IOT>

Rui Santos en Sara Santos

RASPBERRY Pi PROJECT HANDBOEK

20 MAKKELIJKE
PROJECTEN OM DE
LEUKSTE GADGETS
MEE TE MAKEN



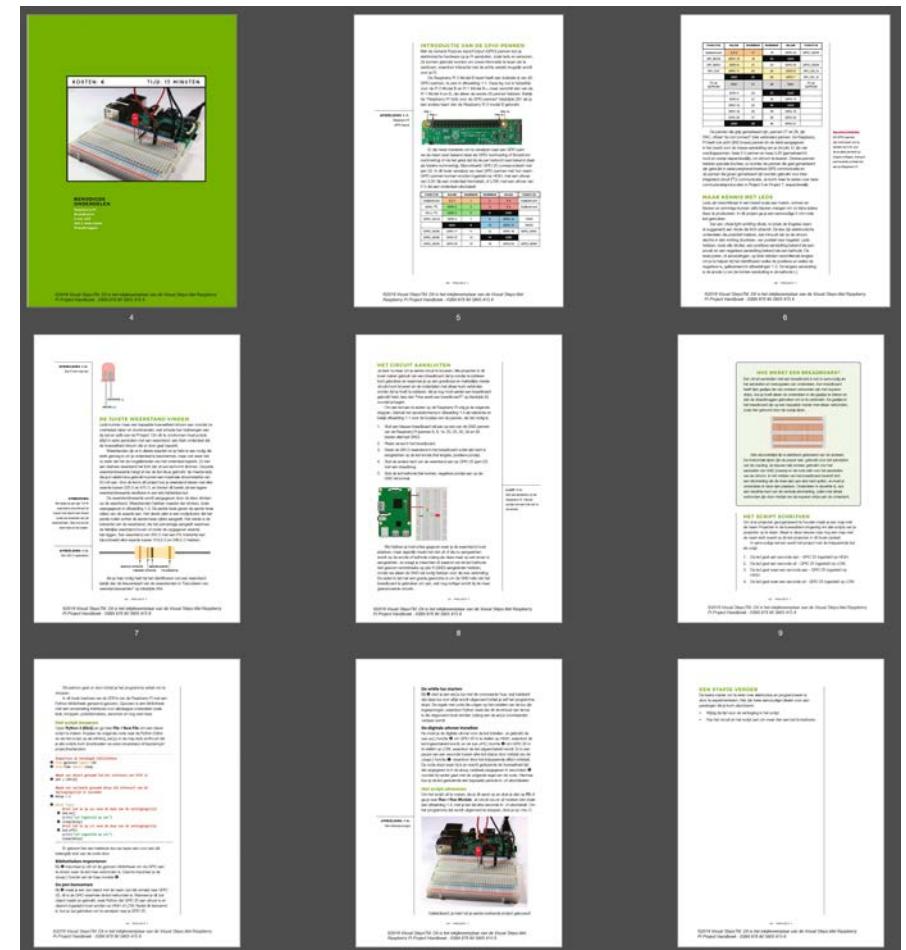
<https://www.visualsteps.nl/raspberrypi/>

<https://www.visualsteps.nl/inkijkexemplaar/957.pdf>

1 EEN LED LATEN KNIPPEREN

IN DIT EERSTE PROJECT GA JE
EEN LED AANSLUITEN OP JE PI
EN DEZE LATEN KNIPPEREN MET
EEN PYTHON SCRIPT. HET LEREN
HOE JE EEN LED LAAT KNIPPEREN
MET DE GPIO-PENNEN IS EEN
BELANGRIJKE STAP IN HET LEREN
OVER JE PI. ALS JE EENMAAL
WEET HOE JE EEN LED KUNT
BESTUREN, KUN JE BIJNA ELKE
UITVOER BESTUREN, OF HET nou
EEN MOTOR, EEN LAMP OF ZELFS
EEN BROODROOSTER IS.

©2019 Visual Steps™. Dit is het inkijkexemplaar van de Visual Steps-titel **Raspberry Pi Project Handboek** - ISBN 978 90 5905 415 8





Learning O'Reilly is een Engelstalig online leerplatform gericht op ICT vaardigheden. Het biedt tienduizenden e-books en vele video's, case studies en "learning paths", waarbij je zelf je voortgang kunt monitoren.

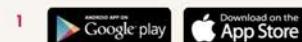
Toegang

Via de website

De Learning O'Reilly website is gekoppeld aan de Hogeschool Rotterdam login. Kom je toch een O'Reilly inlogscherm tegen, vul dan alleen je @hr.nl e-mailadres in en klik op **Sign In with Single Sign On**. Krijg je een blanco pagina te zien? Schakel eventuele adblockers uit!

Via de O'Reilly app

Leer wat je wil, waar je wil. Met de O'Reilly app download je e-books en ga je verder met een playlist waar je op een ander apparaat gebleven was.

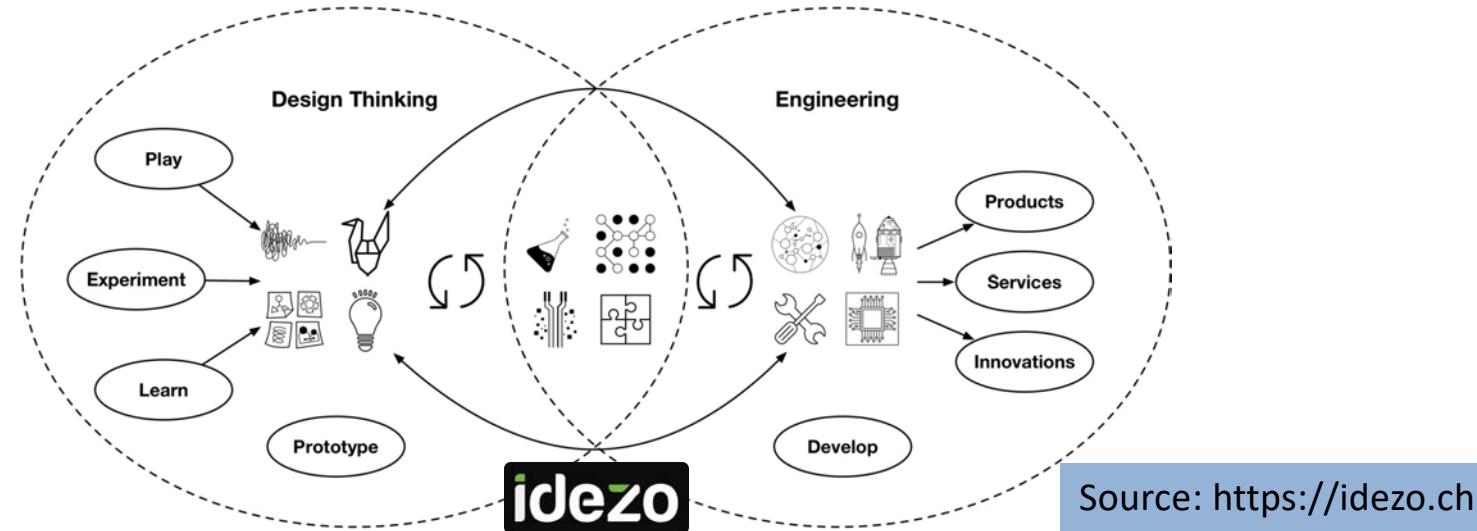


- 2 Gebruik je @hr.nl e-mailadres om in te loggen.

Inhoud

- Meer dan 8000 instructievideo's en 42.000 (hoofdzakelijk Engelstalige) e-books.

HANDS ON APPROACH TO DATA SCIENCE for (the) IoT



This Course material is distributed under the Creative Commons Attribution- NonCommercial-ShareAlike 3.0 license. You are free to copy, distribute, and transmit this work. You are free to add or adapt the work. You must attribute the work to the author(s) listed above.

You may not use this work or derivative works for commercial purposes. If you alter, transform, or build upon this work you may distribute the resulting work only under the same or similar license.

This Data Science Course was developed for Hogeschool Rotterdam (**Rotterdam University of Applied Sciences, RUAS**) through the Program for AI & ethics (SPAiCE). If you find errors or omissions, please contact the author, Rob van der Willigen, at r.f.van.der.willigen@hr.nl. Materials of this course and code examples used will become available at:

<https://github.com/robvdw/CMIDAT01K-DATA-SCIENCE-for-IOT>

Course Setup

Lesson 01: week 02 **Discovering the IoT Data Science Domain**

Lesson 02: week 03 **Defining project requirements**

Lesson 03: week 04 **Learn to write code**

Lesson 04: week 05 **Data Science: How to start your own IoT Project**

Lesson 05: week 06 **IoT Platforms & MiddleWare**

Lesson 06: week 07 **Core IoT Concepts + Code-testing via IoT middleware**

Lesson 07: week 08 **Explaining Grading + Summary + Q & A**

Week 08: **FEEDBACK**

Week 10: **Submit your IoT-Project via LMS**

lesson four

LES: 04

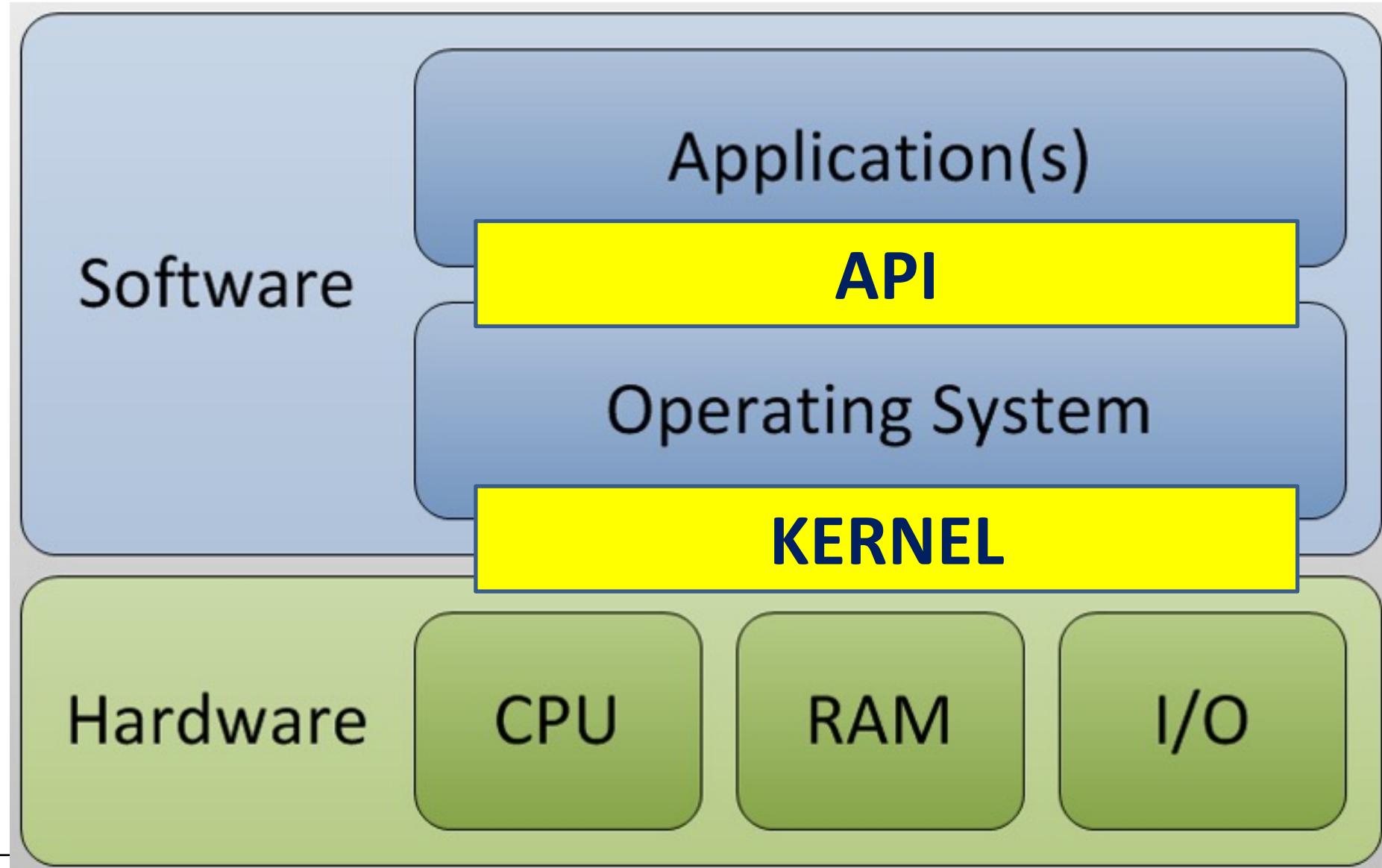
Think like a programmer & Rapid prototyping
Write your first line of code (Python)

Preview Les 05

Data Science Dymistified

How to start Coding

Software vs Hardware



API v.s. SDK v.s. IDE

Application Programming Interface (API)

A library of functions and methods.

You don't need to know how it works, but you have to know how to call them.

Software Development Kit (SDK)

Many tools are included in SDK. Different platform, different SDK.

Integrated Development Environment (IDE)

Usually includes code editor, debugger, compiler.

Types of programming languages

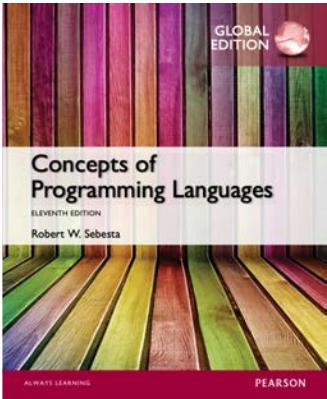
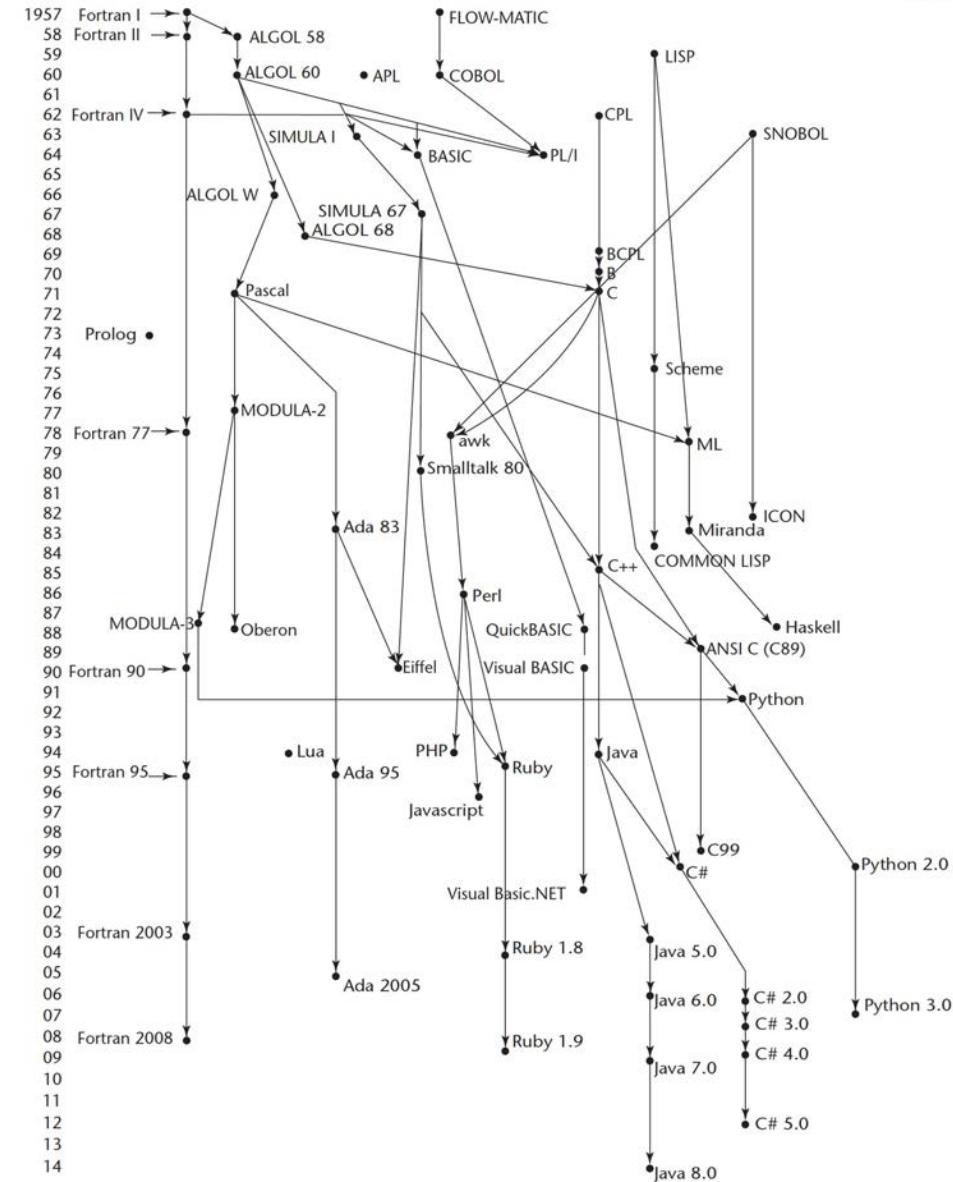


Table 1.1 Language evaluation criteria and the characteristics that affect them

Characteristic	CRITERIA		
	READABILITY	WRITABILITY	RELIABILITY
Simplicity	•	•	•
Orthogonality	•	•	•
Data types	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•



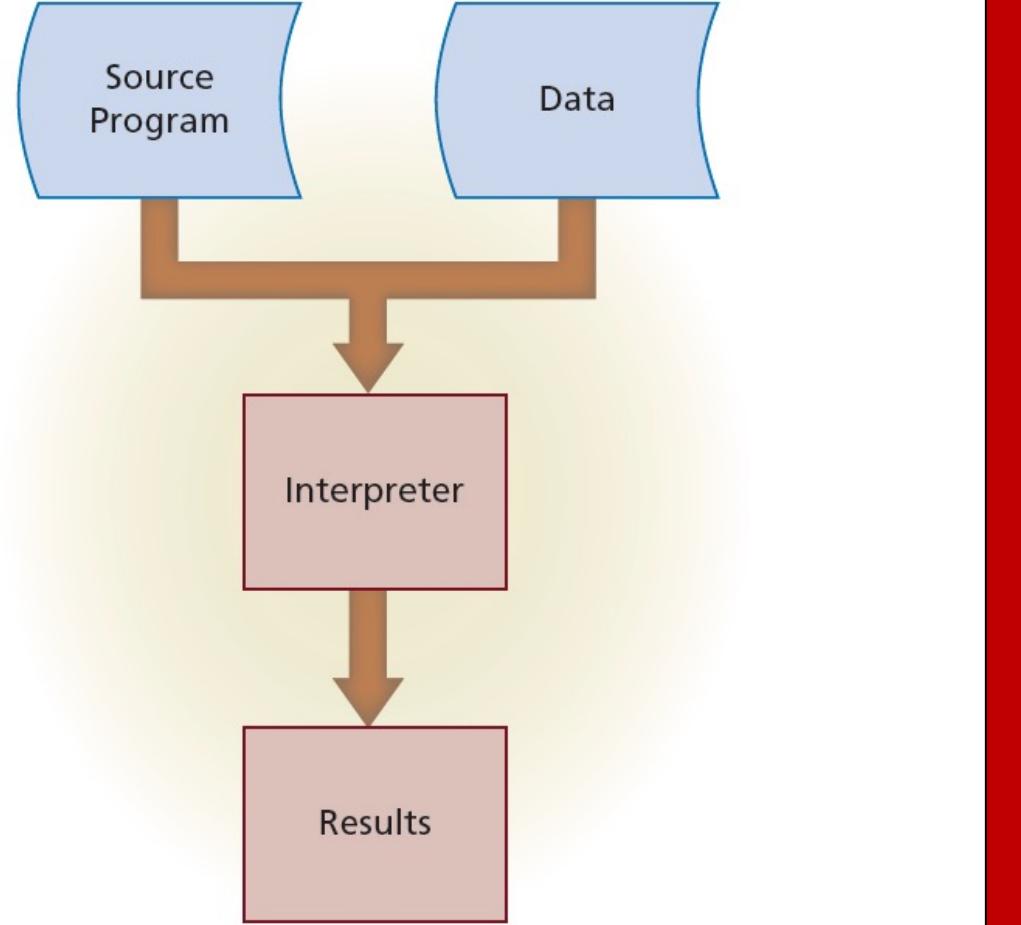
High vs low level Languages

```
Var1 = 0.5
if var1 > 1.3 or var1 < 0.9:
```

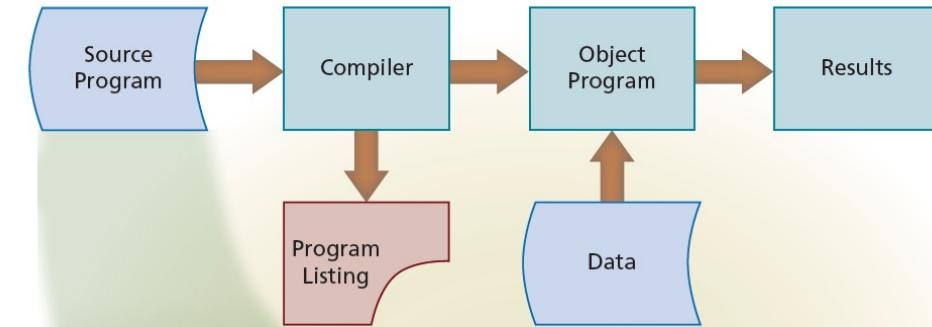
```
10100101010010101010101
010101010010101010100
```

High-level language	Low-level language
Easy for humans to read, write and modify	Hard for humans to read, write and modify
Programs run slower as they make worse use of the CPU and are not very memory efficient	Direct control over the CPU means memory use is more efficient and programs run faster
No understanding of how hardware components work is needed	Good understanding of how the different hardware components work is needed
Examples are python, java, C#, C++ etc.	Examples are machine code and assembly language

Script(ing) languages Do NOT need to be Compiled



High-level Program language need to be Compiled



```
/* Compute Regular Time Pay
rt_pay = rt_hrs * pay_rate;
*/
/* Compute Overtime Pay
if (ot_hrs > 0)
    ot_pay = ot_hrs * 1.5 * pay_rate;
else
    ot_pay = 0;
*/
/* Compute Gross Pay
gross = rt_pay + ot_pay;
*/
/* Display Gross Pay
printf("The gross pay is %d\n", gross);
```

Mark-up vs Programming Languages

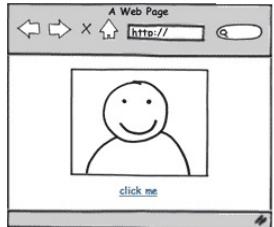
Markup Languages

HTML

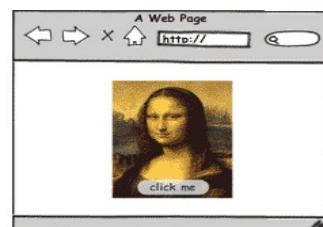
CSS

XML

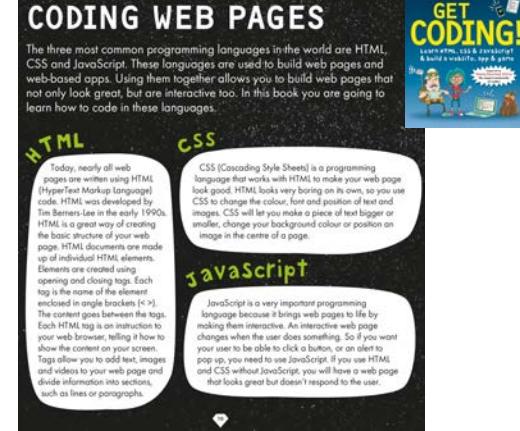
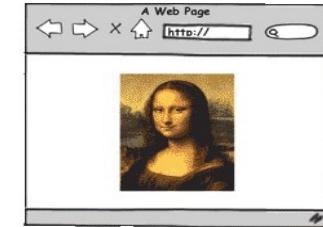
HTML defines content



CSS defines presentation



JavaScript defines behaviour



Programming languages

C

C++

C#

Java

Scripting languages

JavaScript

PHP

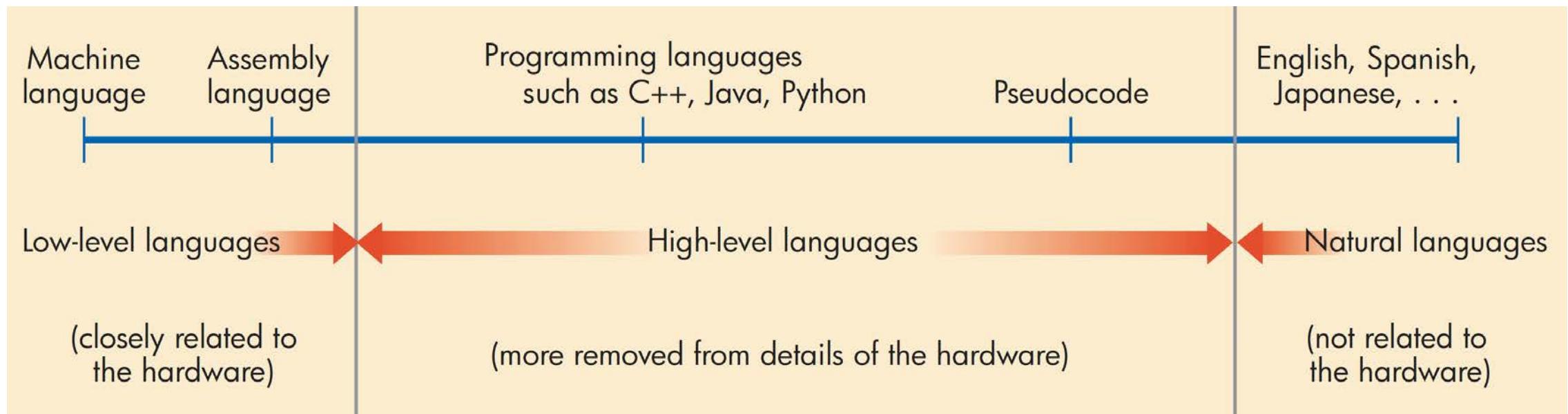
Perl

VBScript

Coding-paradigma's

	Scripting	Compiled
Feature	Can run without compilation	Need to compile to machine code before running
Examples	Python, Perl, PHP, Ruby, Javascript	C/C++, Fortran, Pascal, Java
Advantage	Easy to understand and extend Portable	Optimized and run faster, Source code protected
Running and Debugging model	Use an interpreter to understand and run the code on-the-fly	Codes are compiled to executables using compiler

Coding-paradigma's



Coding: a definition

Anyone interested in **developing software**, such as a source-code, game, or online service, **must start by learning a programming language & decide which (source-code) Editor/Debugger to use.**

<https://www.computerhope.com/issues/ch000675.htm>

CLT vs GUI

Command Line Terminal vs Graphical User Interface

Command Line Terminal (interface)

What Is the Command Line?

The command-line interface, sometimes referred to as the CLI, is a tool into which you can type text commands to perform specific tasks—in contrast to the mouse's pointing and clicking on menus and buttons.

Since you can directly control the computer by typing, many tasks can be performed more quickly, and some tasks can be automated with special commands that loop through and perform the same action on many files—saving you, potentially, loads of time in the process.

Command Line Terminal (interface)

What Is the Command Line?

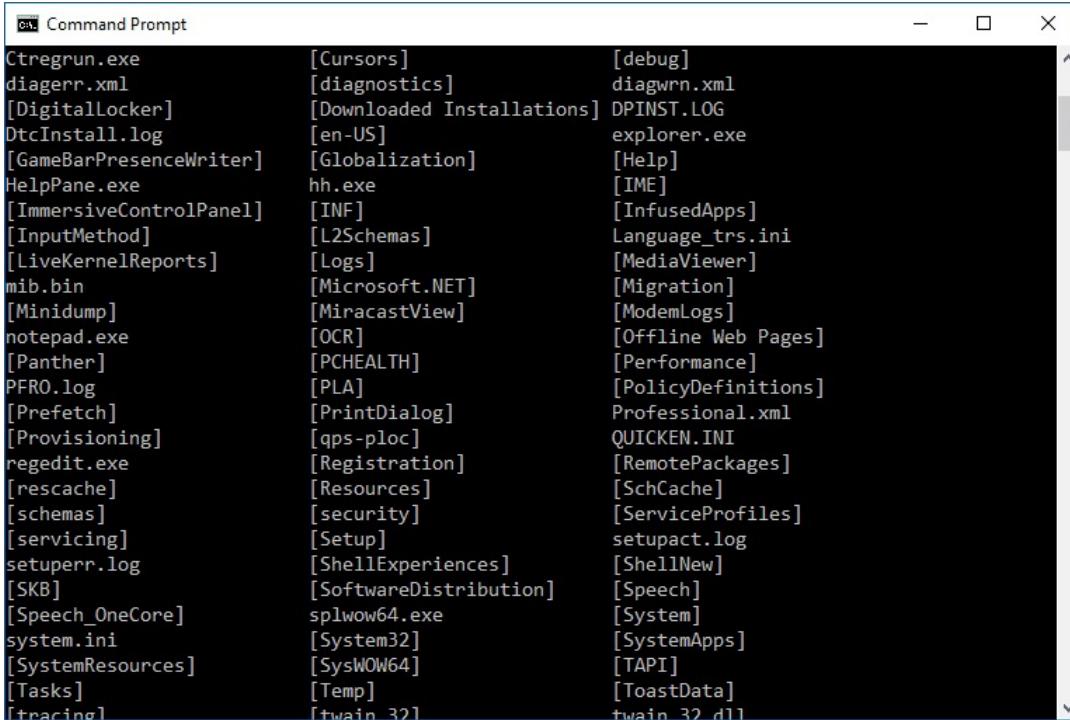
The command-line interface, sometimes referred to as the CLI, is a tool into which you can type text commands to perform specific tasks—in contrast to the mouse's pointing and clicking on menus and buttons.

Since you can directly control the computer by typing, many tasks can be performed more quickly, and some tasks can be automated with special commands that loop through and perform the same action on many files—saving you, potentially, loads of time in the process.

Command Line Terminal (windows)

List of Command Prompt Commands

Complete list of CMD commands for Windows



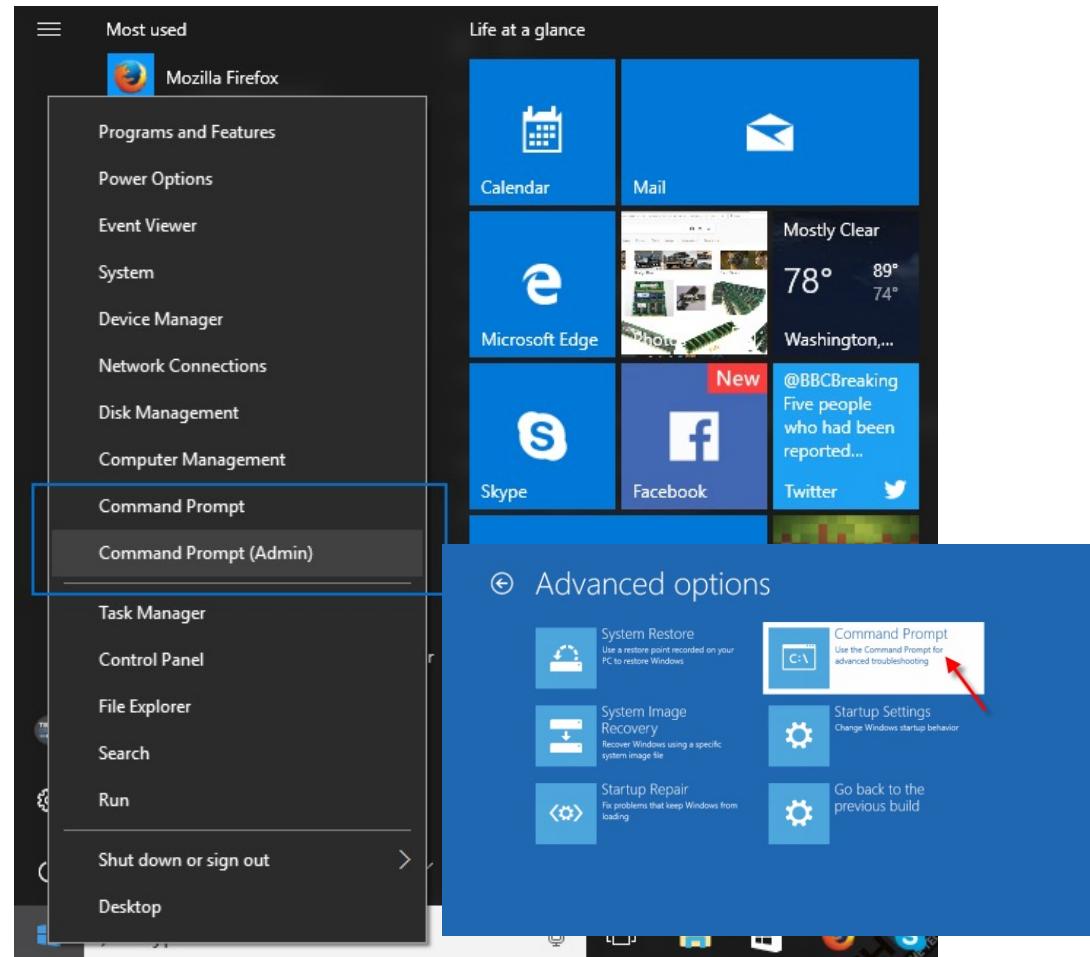
```

cmd Command Prompt
C:\Windows\system32> dir
[...]
  
```

The screenshot shows a Command Prompt window with a large list of command names, each preceded by a short description in brackets. The commands are organized into several groups, such as Cursors, diagnostics, Globalization, INF, L2Schemas, Logs, Microsoft.NET, MiracastView, OCR, PCHEALTH, PLA, PrintDialog, qps-ploc, Registration, Resources, security, Setup, ShellExperiences, SoftwareDistribution, System32, SysWOW64, Temp, and twain_32.dll.

The [Command Prompt](#) in Windows provides access to over 280 [commands](#)! These commands are used to do certain [operating system](#) tasks from a [command line interface](#) instead of the graphical Windows interface we use most of the time.

Note: It's important to know that the commands in Windows 10, 8, 7, Vista, and XP are called *CMD commands* or *Command Prompt commands*, and the commands in Windows 98/95 and MS-DOS are called *DOS commands*.



<https://www.lifewire.com/command-prompt-tricks-and-hacks-2618104>

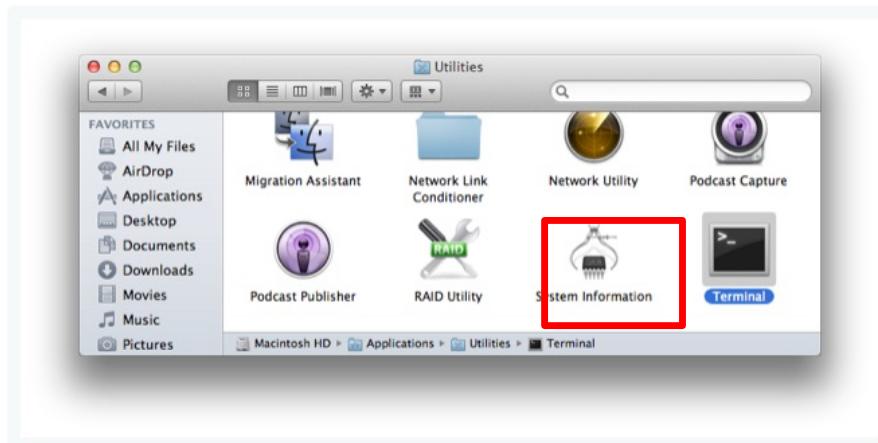
Command Line Terminal MacOS



How to open the command line.

Before you can use it, you need to be able to find it.

So what we need to do is open the terminal. On OS X, open your Applications folder, then open the Utilities folder. Open the Terminal application. You may want to add this to your dock. I like to launch terminal by using Spotlight search in OS X, searching for "terminal".



<http://blog.teamtreehouse.com/introduction-to-the-mac-os-x-command-line>

Anatomy of the Console

First let's clarify a few terms.

Console: This is the system as a whole. This is both the command line as well as the output from previous commands.

Command Line: This is the actual line in a console where you type your command.

Prompt: This is the beginning of the command line. It usually provides some contextual information like who you are, where you are and other useful info. It typically ends in a \$. After the prompt is where you will be typing commands.

Terminal: This is the actual interface to the console. The program we use to interact with the console is actually a "terminal emulator", providing us the experience of typing into an old school terminal from the convenience of our modern graphical operating system.

Running a Command.

Nearly all commands follow a common pattern with 3 main parts. The program, the options, and the arguments. Let's see an example.

```
$ ls -l ~
```

A screenshot of a Mac OS X terminal window titled 'jim — bash — 102x19'. The window shows the output of the command 'ls -l ~'. The output lists the contents of the user's home directory (~) in a long-form listing format, showing file names, permissions, owner, group, size, and modification date.

```
Last login: Tue Sep 25 12:52:00 on console
jim-Hoskinss-iMac:~ jim$ ls -l ~
total 8
drwxr-xr-x+ 7 jim staff 238 Sep 25 12:57 Desktop
drwxr-xr-x+ 4 jim staff 136 Sep 25 12:58 Documents
drwxr-xr-x+ 4 jim staff 136 Sep 25 12:58 Downloads
drwxr-xr-x@ 31 jim staff 1854 Sep 25 12:53 Library
drwxr-xr-x+ 3 jim staff 102 Sep 25 12:58 Movies
drwxr-xr-x+ 3 jim staff 102 Sep 25 12:58 Music
drwxr-xr-x+ 4 jim staff 136 Sep 25 12:58 Pictures
drwxr-xr-x+ 5 jim staff 170 Sep 25 12:58 Public
jim-Hoskinss-iMac:~ jim$
```

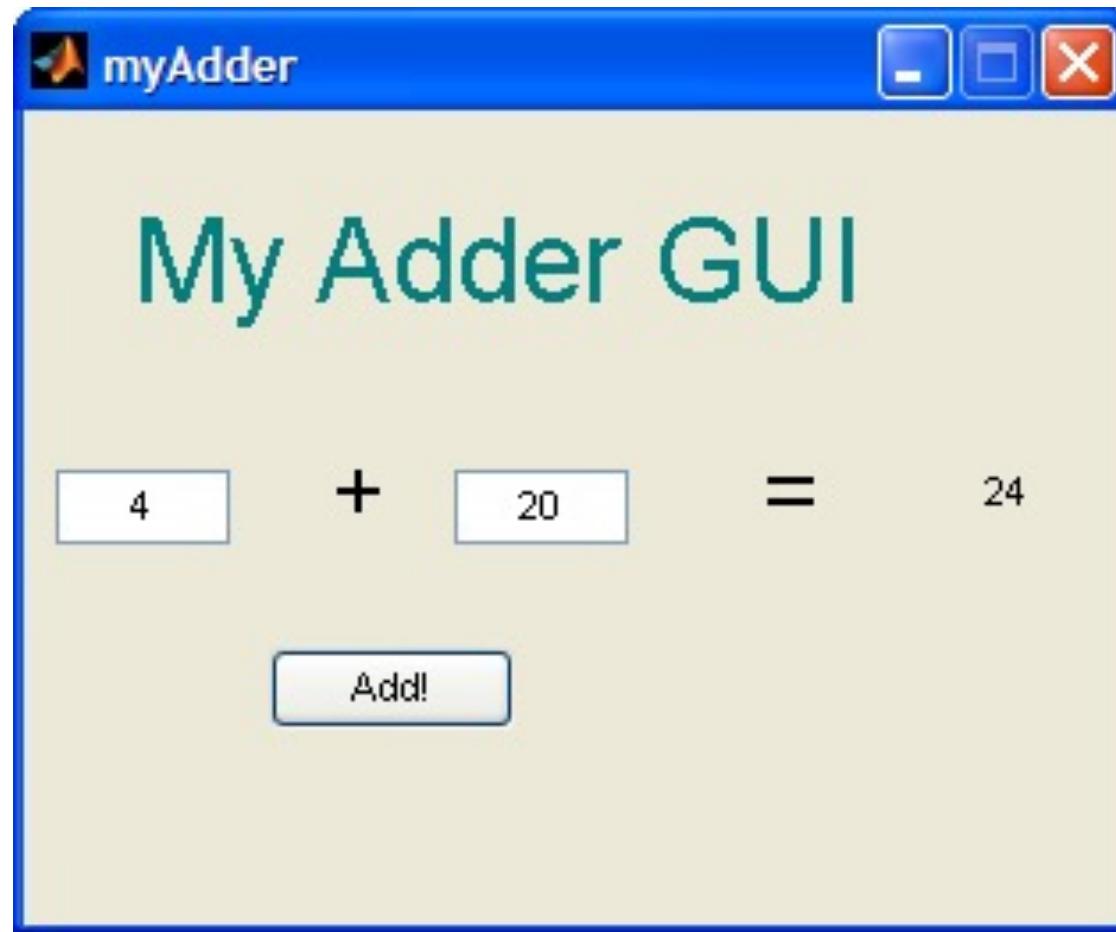
Command Line Terminal

Online Terminals

 CentOS	 Ipython	 Python-3	 Lua	 Memcached	 Mongo DB
 MySQL	 Node.js	 Numpy	 Oracle	 Octave	 PowerShell
 PHP	 R Programming	 Redis	 Ruby	 Scipy	 Sympy

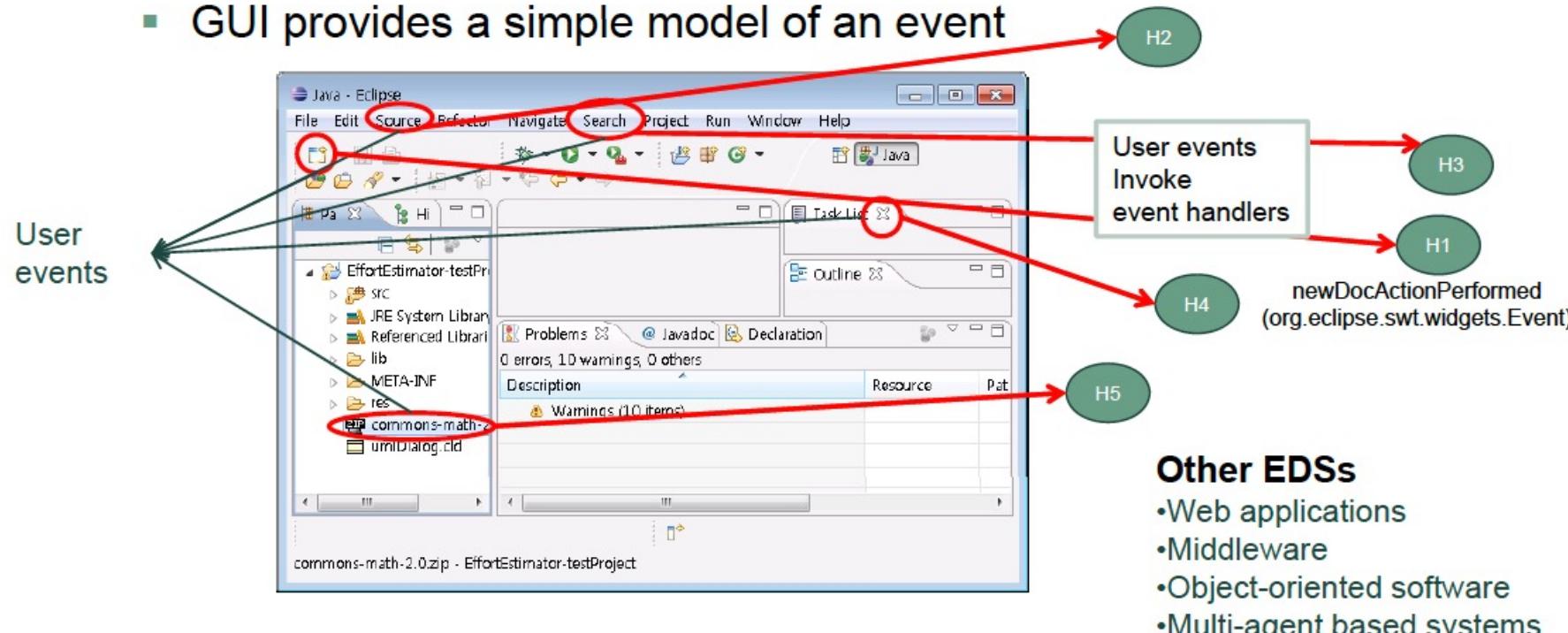
<http://www.tutorialspoint.com/codingground.htm>

Graphical User Interface (GUI)



Graphical User Interface (GUI)

- GUI provides a simple model of an event



GUI is a type of user interface item that allows people to interact with programs in more ways than typing.
GUI software is an Event-driven Software(EDS)

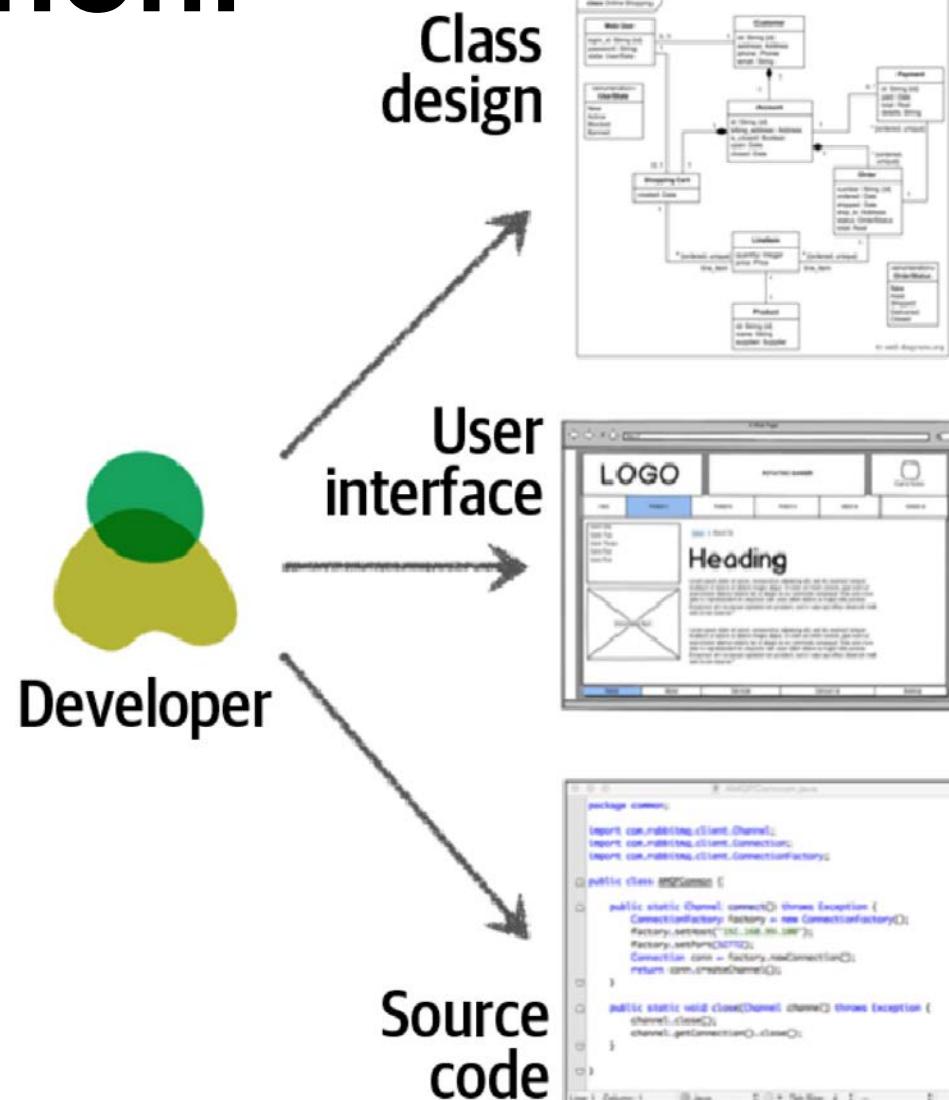
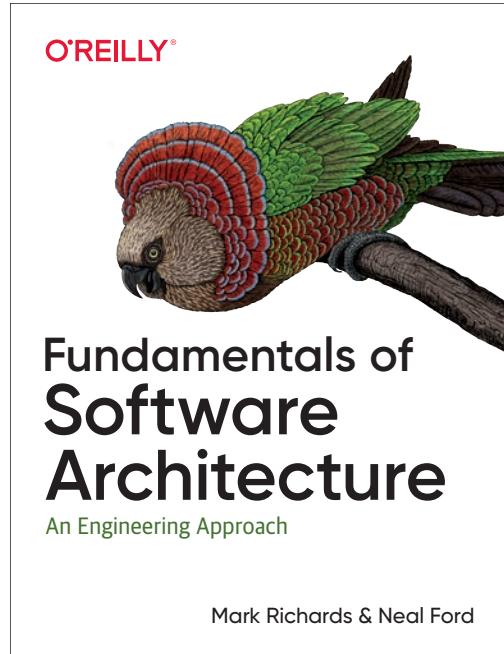
CLI vs GUI

Topic	Command line (CLI)	GUI
Ease	Due to a higher degree of memorization and familiarity needed for operation and navigation, new users find operating a command line interface more difficult than a GUI.	Because a GUI is much more visually intuitive, users typically pick up on how to use a GUI faster than a command line interface.
Control	Users have a good bit of control over both the file and operating systems in a command line interface. However, for new or novice users, it is not as user friendly as a GUI.	A GUI offers a lot of access to files, software features, and the operating system as a whole. Being more user friendly than a command line, especially for new or novice users, a GUI is utilized by more users.
Multitasking	Although many command line environments are capable of multitasking, they do not offer the same ease and ability to view multiple things at once on one screen.	GUI users have windows that enable a user to view, control, manipulate, and toggle through multiple programs and folders at same time.
Speed	Command line users only need to utilize a keyboard to navigate the interface, often resulting in faster performance.	While newer technology is making a GUI faster and more efficient than ever before, using both a mouse and keyboard to navigate and control the GUI is still a bit slower than a command line interface.
Resources	A computer that is only using the command line takes a lot less of the computer's system resources than a GUI.	A GUI requires more system resources because of the elements that require loading, such as icons and fonts. Video, mouse, and other drivers need to be loaded, taking up additional system resources.

Scripting	A command line interface mostly requires users to already know scripting commands and syntax, making it difficult for new or novice users to create scripts.	Creating scripts using a GUI has become much easier with the help of programming software, which allows users to write the scripts without having to know all the commands and syntax. Programming software provides guides and tips for how to code specific functions, as well as preview options to see if and how the script will work.
Remote access	When accessing another computer or device over a network, a user can manipulate the device or its files with a command line interface. However, you must know the commands to do so and is not as easy for new or novice users.	Remotely access another computer or server is possible in a GUI and easy to navigate with little experience. IT professionals typically use a GUI for remote access, including the management of servers and user computers.
Diversity	After you've learned how to navigate and use a command line, it's not going to change as much as a new GUI. Although new commands may be introduced, the original commands almost always remain the same.	Each GUI has a different design and structure when it comes to performing different tasks. Even different iterations of the same GUI, such as Windows, can have hundreds of different changes between each version.
Strain	A command line interface is often very basic and can be more of a strain on a user's vision. Carpal Tunnel Syndrome can also be a bit of a risk when using a command line interface because users are only using a keyboard. There is little need to change hand positions and strain to the wrists or even fingers can occur.	The use of shortcut keys and more frequent movement of hand positions, due to switching between a keyboard and a mouse, strain may be reduced. Visual strain can still be a risk, but a GUI has more colors and is more visually appealing, leading to a potential reduction in visual strain.

Integrated Development Environment {IDE}

Code development



Integrated Development Environment

Text Editor

Syntax Coloring

Autocomplete

Indexer

Doc Viewer

Code Templates



Builder

Compiler

Lexer

Parser

Assembler

Linker



Debugger GUI

Debugger



What is (source) Program/DEV Software: Integrated Development Environment (**IDE**)?

IDEs are designed to encompass
all **programming tasks** in one **application**.

IDE is a software application that provides comprehensive facilities or graphical user interface **[GUI]** to **programmers** for **software development** in addition to **CLI** (command line interfacing).

An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs have intelligent code completion.

What is (**source-code/DEV**) Program Software: Integrated Development Environment (**IDE**)?

Code editor: This feature is a **text editor** designed for writing and editing source code.

Source code editors are distinguished from text editors because they enhance or simplify the writing and editing of code.

Compiler/Interpreter: This tool **transforms source code** written in a human readable/writable language into a form **executable** by a computer.

Debugger: This tool is used during testing to help **debug application** programs.

Build automation tools: These tools automate common **developer** tasks.

Source-Code Editor

A **source code editor** is a text editor program **designed specifically for editing source code** of computer programs by programmers.

It may be a **Standalone Application** or it may be built into an **Integrated Development Environment (IDE)** or **Web Browser**.

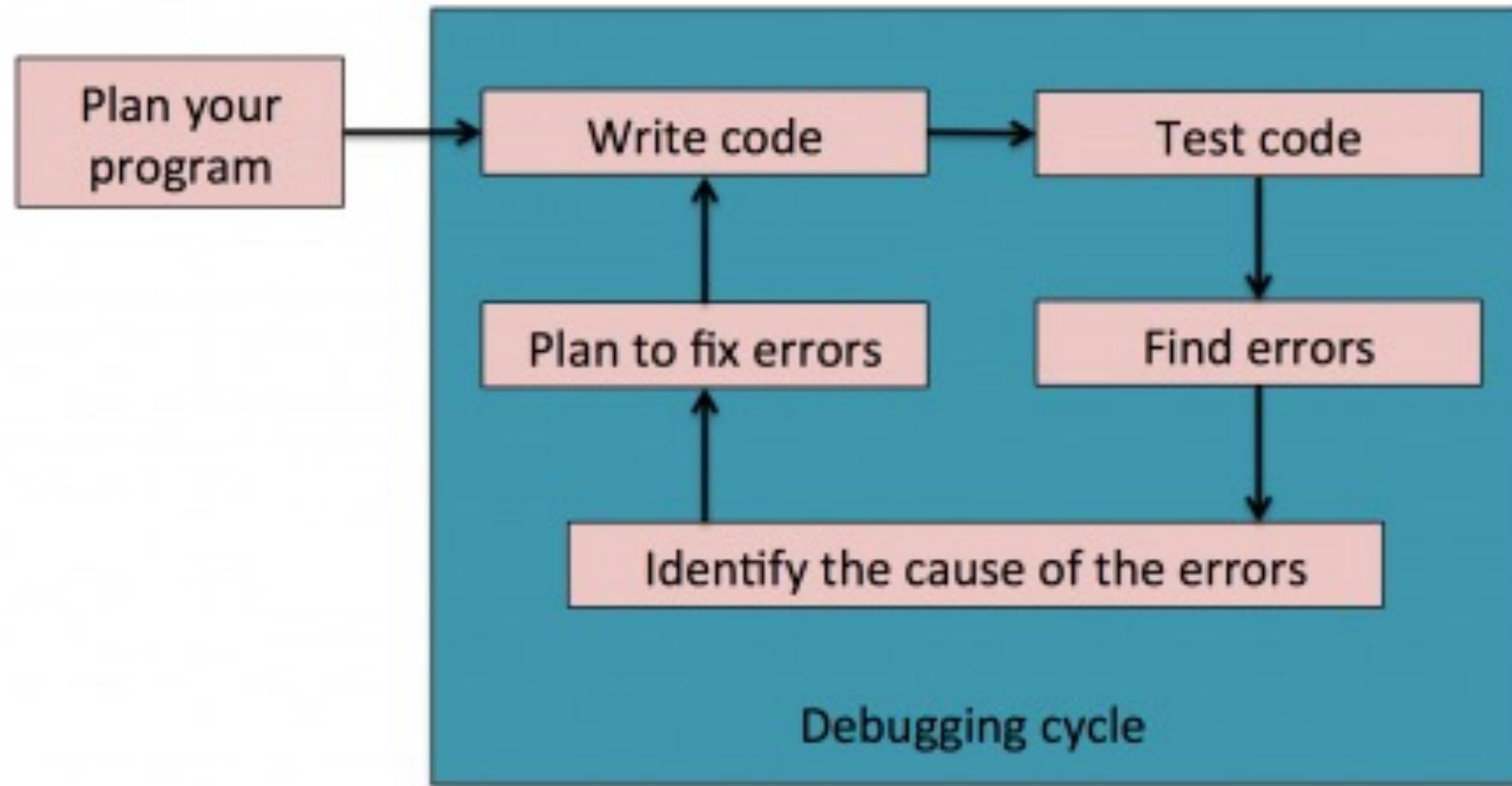
Source code editors are the **most fundamental programming tool**, as the fundamental job of programmers is **to write and edit source code**.

Debugging

Debugging is a process of analyzing a computer program **source-code** and **removing** its **logical** or **syntactical errors**.

Software which assists this process is a **debugger**.

Using a **debugger**, a **software developer** can step through a **program's code** and **analyze** its variable values, searching for **errors**.



**Things you
should
Do**

To Do's



Try Visual Studio Code

Visual Studio Code

Editing evolved

Start

- New file
- Open folder...
- Add workspace folder...

Recent

- SHANKEY_D3-workspace (Workspace) /Volumes/RESTORE_BACKUP/PARALLELS_9/HRO/HOO...
- THEMA_CMI (Workspace) /Volumes/RESTORE_BACKUP/PARALLELS_9/HRO/HOOFDDOCENTE...
- SHANKEY /Volumes/RESTORE_BACKUP/PARALLELS_9/HRO/HOOFDDOCENTEN_OVERLEG/TH...
- SHANKEY_FORM (Workspace) /Volumes/RESTORE_BACKUP/PARALLELS_9/HRO/CoP_MAKING...

More... (^R)

Help

- Printable keyboard cheatsheet
- Introductory videos
- Tips and Tricks
- Product documentation
- GitHub repository
- Stack Overflow

Show welcome page on startup

Customize

Tools and languages

Install support for JavaScript, TypeScript, Python, PHP, Azure, Docker and more

Install keyboard shortcuts

Install the keyboard shortcuts of Vim, Sublime, Atom and others

Color theme

Make the editor and your code look the way you love

Learn

Find and run all commands

Rapidly access and search commands from the Command Palette (^⌘P)

Interface overview

Get a visual overlay highlighting the major components of the UI

Interactive playground

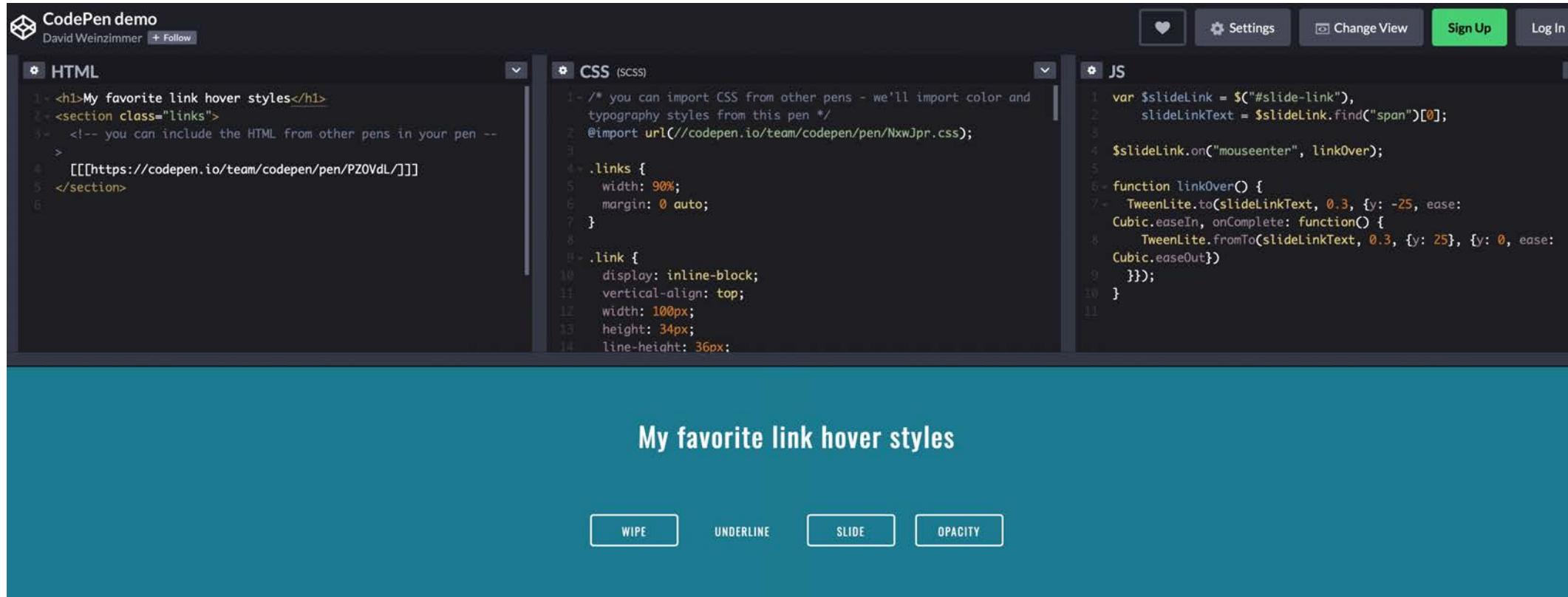
<https://visualstudio.microsoft.com/services//github-codespaces/>

GitHub Codespaces

Access cloud-hosted dev environments from Visual Studio, Visual Studio Code, or your browser.

[Learn more about GitHub Codespaces >](#)

Try CODEPEN.IO



The screenshot shows the CodePen interface with a dark theme. At the top, there's a navigation bar with a heart icon, settings, change view, sign up, and log in buttons. Below the navigation is a title bar for "CodePen demo" by David Weinzimmer.

The main area is divided into three panels:

- HTML:**

```
<h1>My favorite link hover styles</h1>
<section class="links">
  <!-- you can include the HTML from other pens in your pen -->
  <a href="https://codepen.io/team/codepen/pen/PZ0VdL/">
</section>
```
- CSS (SCSS):**

```
/* you can import CSS from other pens - we'll import color and
typography styles from this pen */
@import url(//codepen.io/team/codepen/pen/NxwJpr.css);

.links {
  width: 90%;
  margin: 0 auto;
}

.link {
  display: inline-block;
  vertical-align: top;
  width: 100px;
  height: 34px;
  line-height: 36px;
}
```
- JS:**

```
var $slideLink = $("#slide-link"),
slideLinkText = $slideLink.find("span")[0];

$slideLink.on("mouseenter", linkOver);

function linkOver() {
  TweenLite.to(slideLinkText, 0.3, {y: -25, ease: Cubic.easeIn, onComplete: functionO {
    TweenLite.fromTo(slideLinkText, 0.3, {y: 25}, {y: 0, ease: Cubic.easeOut})
  }});
}
```

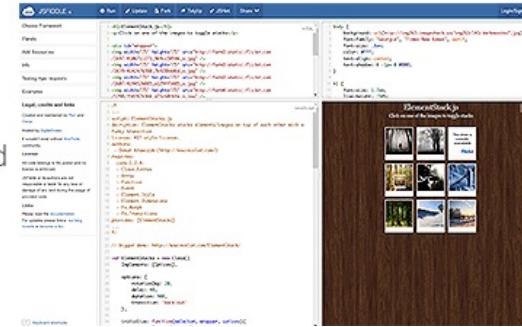
The preview area below shows the text "My favorite link hover styles" with three buttons below it: WIPE, UNDERLINE, SLIDE, and OPACITY. The "SLIDE" button is highlighted.

<https://codepen.io/dweinz/pen/GdMbPz>

Source-Code Play-Grounds

JSFiddle

[JSFiddle](#) was one of the earliest code playgrounds and a major influence for all which followed. Despite the name, it can be used for any combination of HTML, CSS and JavaScript testing. It's looking a little basic today, but still offers advanced functionality such as Ajax simulation.



CodePen

The prize for the best-looking feature-packed playground goes to [CodePen](#). The service highlights popular demonstrations ("Pens") and [Projects](#), which is an online Integrated Development Environment you can use to build and deploy web projects, a feature only added in March 2017. It offers advanced functionality such as sharing and embedding of Pens, adding external JS and CSS libraries, popular preprocessors, and tons more. The PRO service provides cross-browser testing, pair-programming and teaching options starting from just \$9 per month.



<https://www.sitepoint.com/7-code-playgrounds/>

Plunker

[Plunker](#) lets you add multiple files, including community generated templates, to kick-start your project. Just like CodePen, with Plunker you can create working demos, also in collaboration with other devs, and share your work. Plunker's source code is free and lives on its [GitHub repository](#).

Plunker Helping developers make the web
Plunker is an online community for creating, collaborating on and sharing your web development ideas.

[Search the Editor](#) [Browse Plasma](#) [Tags](#) [Users](#) [Editor](#) [Groups](#)

Design goals

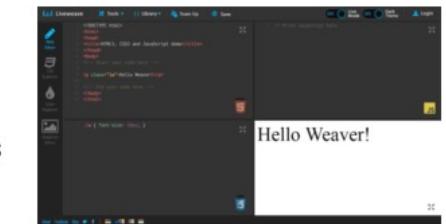
- Speed: Demos in complexity, the Plunker editor is designed to be in under 2 seconds
- Ease of use: Plunker's features should just work and not require additional explanation
- Collaboration: Encouraging real-time collaboration by forking and commenting, Plunker wants to encourage users to work together on their code.

Features

- Real-time code collaboration
- Fully featured, customizable syntax editor
- Live previewing of code changes
- As-you-type code linting
- Comments and forks for real-time collaboration for forking and commenting
- Forking and cloning for easy sharing of Plasma
- Fully open-source on GitHub under the MIT license
- And many more to come.

Liveweave

[Liveweave](#) is one more online HTML5, CSS3 & JavaScript editor with live preview capabilities. It offers code-hinting for HTML5, CSS3, JavaScript and jQuery and lets you download your project as a zip file.



You can also add external libraries such as jQuery, AngularJS, Bootstrap etc. quite easily in your workspace. Furthermore, Liveweave offers a ruler to help you code responsive designs and a "Team Up" feature which has the same features as JSFiddle's collaborative editing.

https://en.wikipedia.org/wiki/Comparison_of_online_source_code_playgrounds

Watch: “How to think like a programmer”



<https://www.youtube.com/watch?v=azcrPFhaY9k>

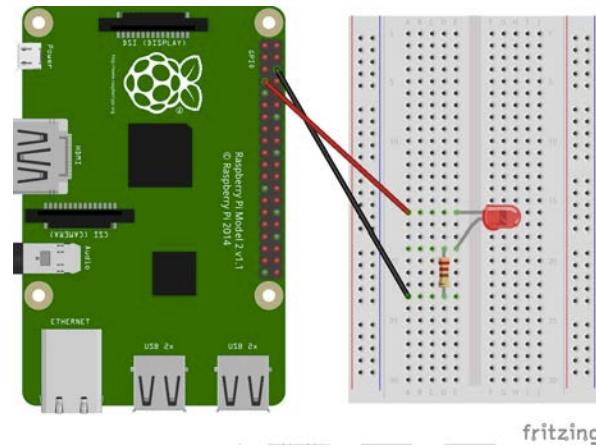


DIY Blinking an LED on the Raspberry Pi

Do it yourself (DIY) : [Make a LED blinking](#)

Raspberry Pi, Power Supply,
 microSD Card with Raspbian Installed and Setup,
 Internet connectivity (Optional),
 Bread board,
 connecting wires,
 an LED 5mm of preferred colour
 and a resistor of 220ohms

<https://embeddedcode.wordpress.com/2017/01/18/blinking-an-led-on-the-raspberry-pi/>



LEARNING OBJECTIVES

In this Rapid prototyping experiment,
 You will learn the essentials of Raspberry Pi GPIO control
 by toggling an LED at predefined intervals of time.

KEY WORDS, CONCEPTS, & PRACTICES

- + LED
- + Resistor
- + GPIO
- + Rapid Prototyping
- + Coding (Python)



Reminder lesson 01

1st STEP Managing Data Science for IoT Projects

→→→ Getting started with GitHub ←←←

A Computational Thinking culture has an intellectual dimension, engaging with a set of creative concepts and practices. It has a physical dimension, encouraging interactions with others through the placement of desks, chairs, and computers. Most importantly, it has an affective dimension, cultivating a sense of confidence and fearlessness.

LEARNING OBJECTIVES

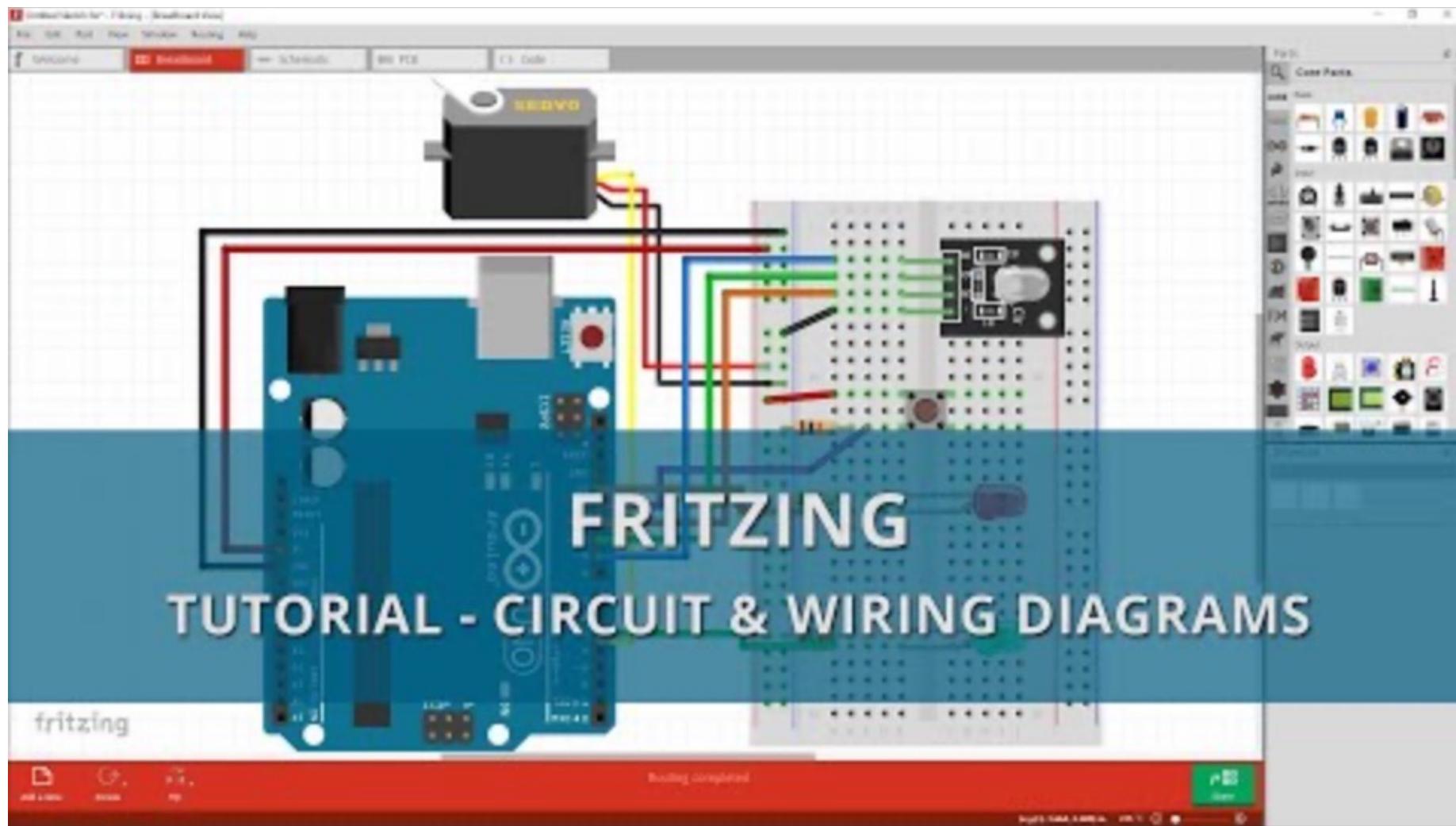
Students will:

- + be introduced to the concept of computational Thinking, in the context of GitHub
- + be able to imagine possibilities for their own GitHub-based computational thinking
- + become familiar with resources that support their computational thinking
- + prepare for creating GitHub projects by establishing a GitHub account, exploring GitHub, creating design journals

KEY WORDS, CONCEPTS, & PRACTICES

- + GitHub
- + Computational Thinking

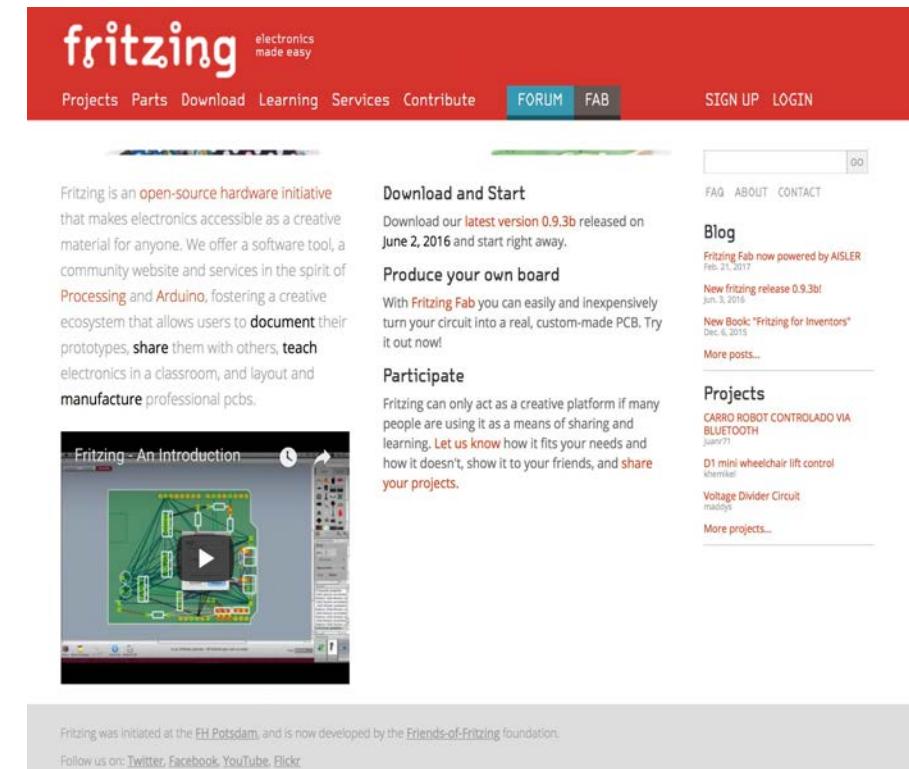
Fritzing beginners guide



Fritzing

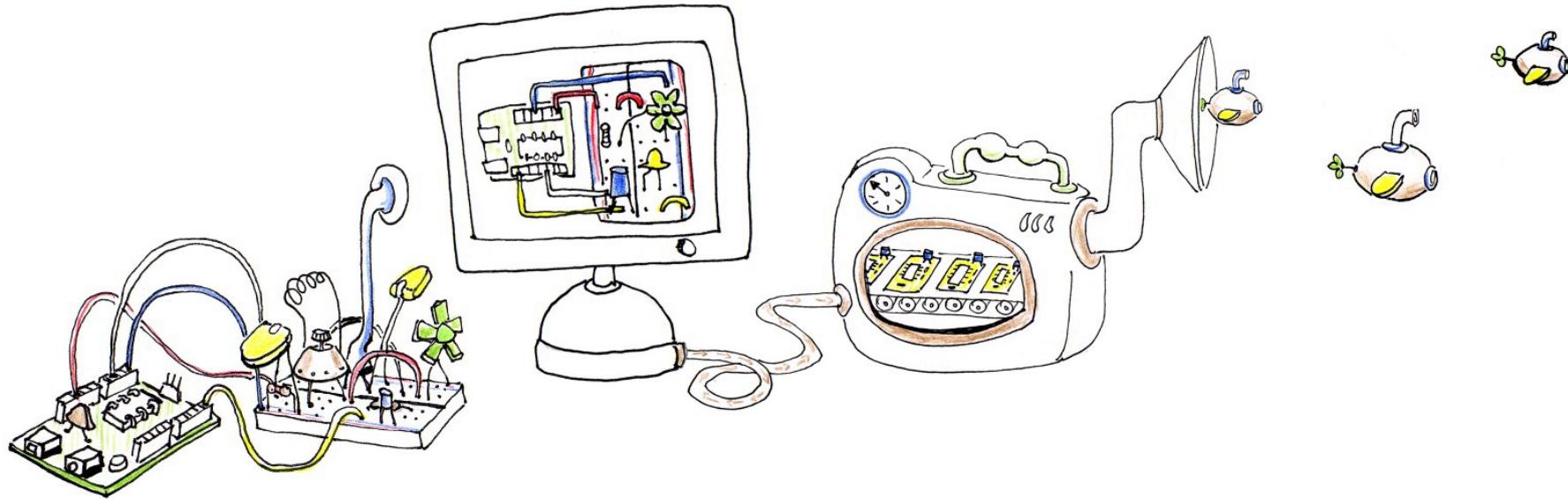
A tool for advancing electronic prototyping for designers

Today a growing community of DIY-practitioners, artists and designers are using microcontroller-based toolkits to express their concepts for digital artifacts by building them.



The screenshot shows the Fritzing website homepage. The header features the Fritzing logo with the tagline "electronics made easy". Below the header, there are navigation links for Projects, Parts, Download, Learning, Services, Contribute, Forum (highlighted in blue), FAB, SIGN UP, and LOGIN. The main content area has several sections: "Download and Start" (with a link to version 0.9.3b from June 2, 2016), "Produce your own board" (describing Fritzing Fab), "Participate" (encouraging sharing and learning), and a video player showing "Fritzing - An Introduction". On the right side, there's a sidebar with a search bar, links for FAQ, ABOUT, CONTACT, and a Blog section listing recent posts. At the bottom, a footer notes that Fritzing was initiated at FH Potsdam and developed by the Friends-of-Fritzing foundation, with links to social media platforms.

Concept of Fritzing



The recreation of their breadboard prototype in the Fritzing software enables designers to produce professional PCBs and also to share their designs.

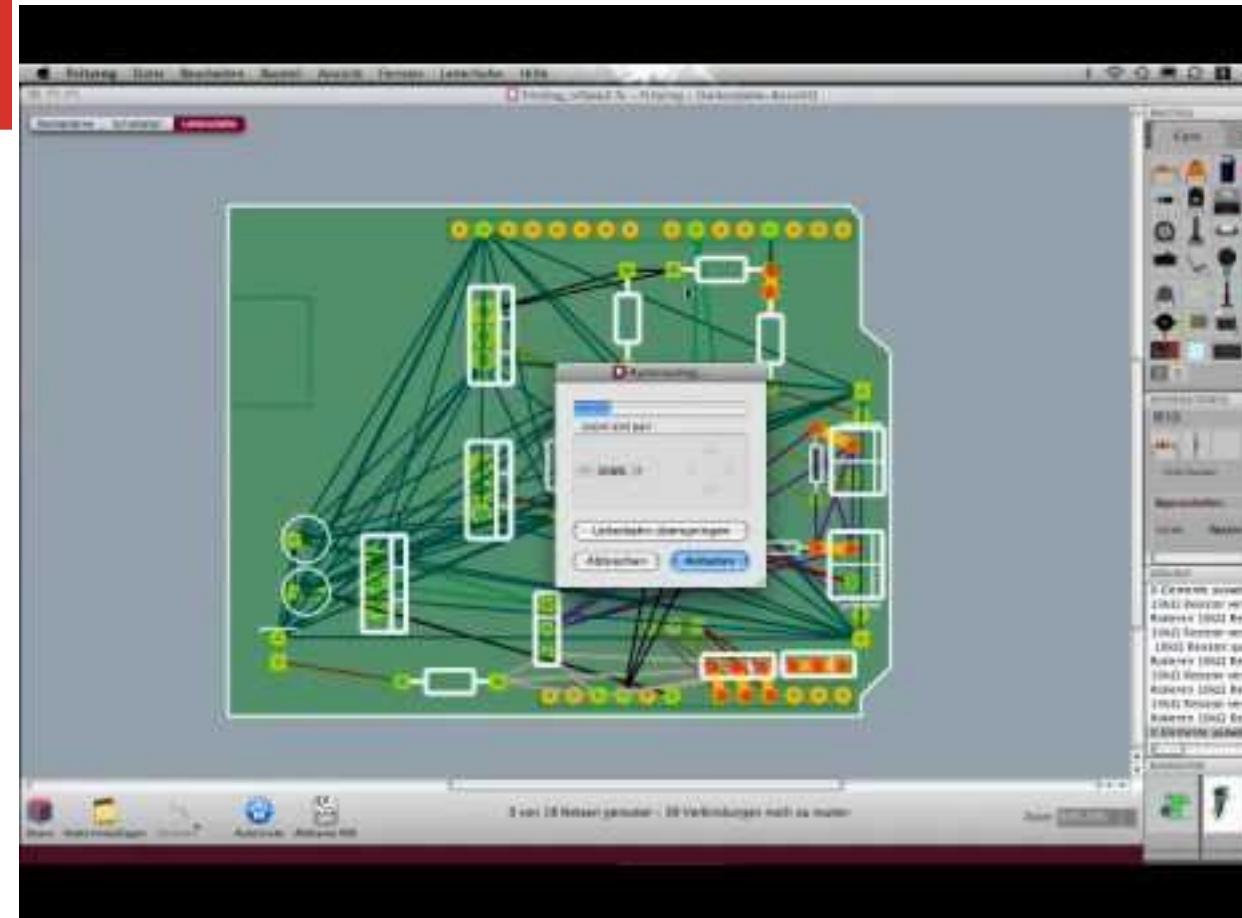
Installing Fritzing



So new Raspberry Pi 3 Model B has just been released. There are two giant upgrades in the Pi 3. The first is a next generation Quad Core Broadcom BCM2837 64-bit ARMv8 processor, making the processor speed increase from 900 MHz on the Pi 2 to up to 1.2GHz on the Pi 3.

The second giant upgrade (and this is the one we're personally most excited about) is the addition of a BCM43143 WiFi chip BUILT-IN to your Raspberry Pi. No more pesky WiFi adapters - this Pi is WiFi ready. There's also Bluetooth Low Energy (BLE) on board making the Pi an excellent IoT solution (BLE support is still in the works, software-wise).

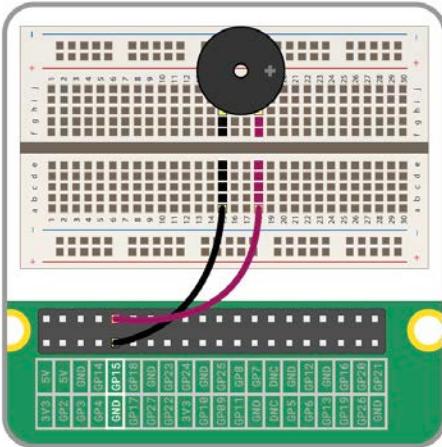
<https://fritzing.org/projects/raspberry-pi-3>



Rapid Prototyping with Raspberry Pi

How It Works:

1 ASSEMBLE

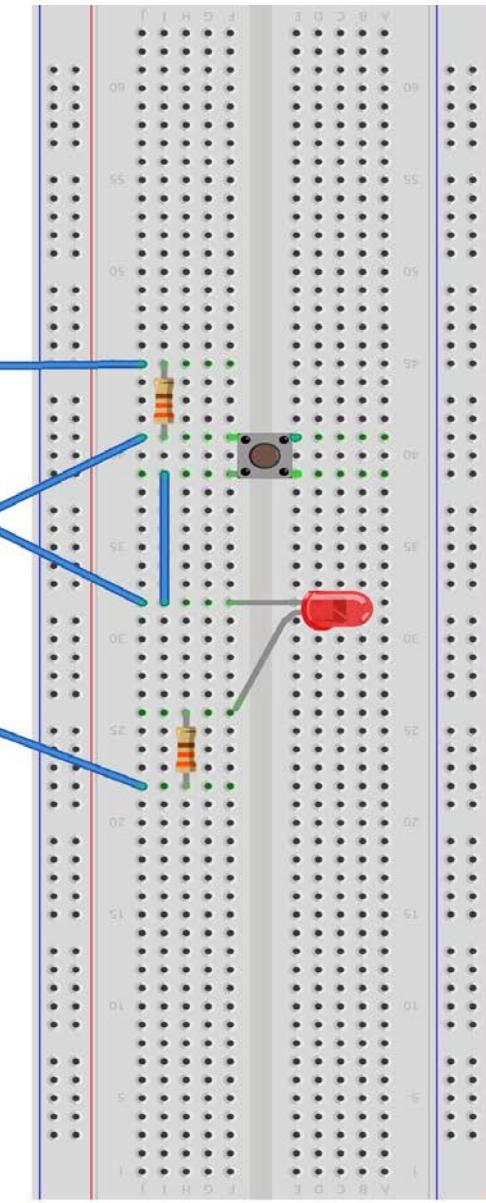
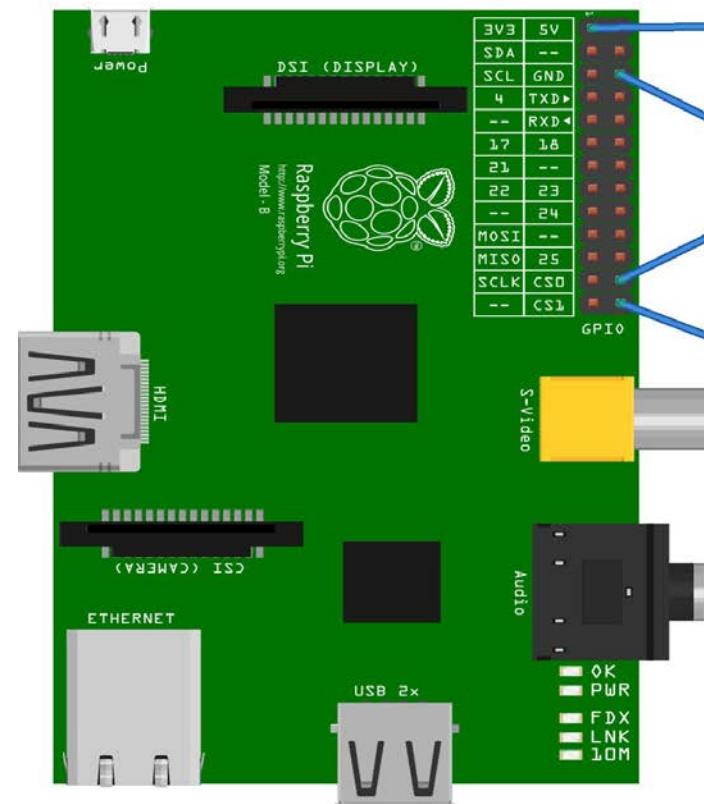
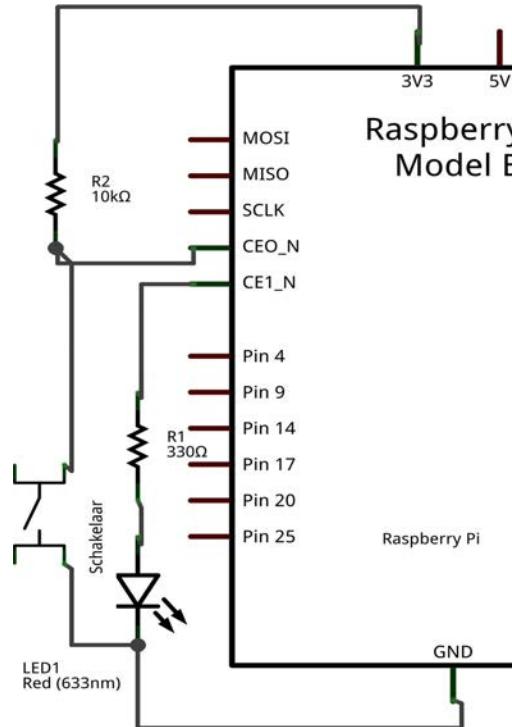


2 WRITE



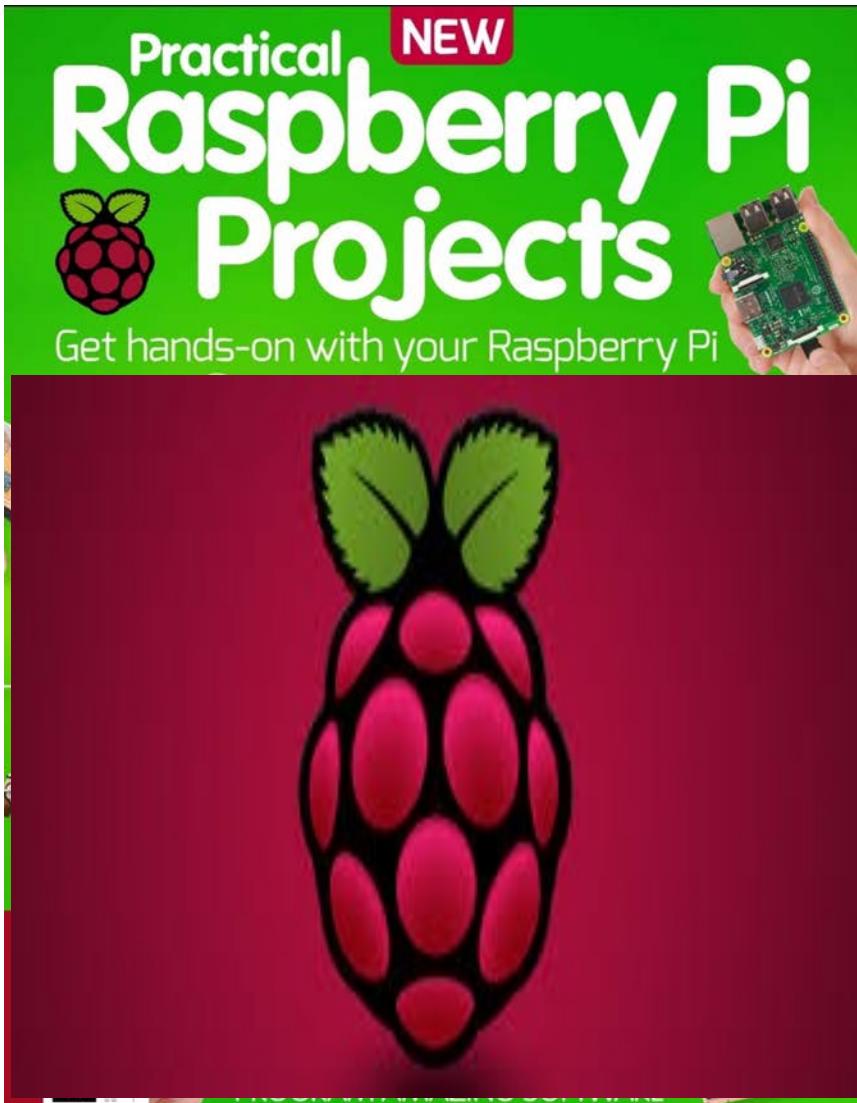
3 UPLOAD

Rapid Prototyping with Raspberry Pi + Breadboard



Made with  Fritzing.org

Raspberry Pi



Get interactive with Scratch

Experiment with physical computing by using Scratch to interact with buttons and lights on your Pi

What you'll need

- Breadboard
- LEDs
- Buttons
- Resistors
- Jumper wires
- ScratchGPIO3

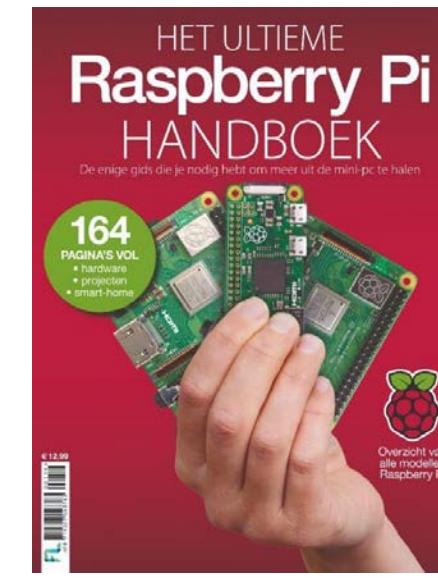
Scratch is a very simple visual programming language, commonly used to teach basic programming concepts to learners of any age. In this project we'll learn how to light up an LED when a button is pressed in Scratch, and then change a character's colour when a physical button is pressed. With these techniques you can make all manner of fun and engaging projects, from musical keyboards to controllers for your Scratch games and animations.

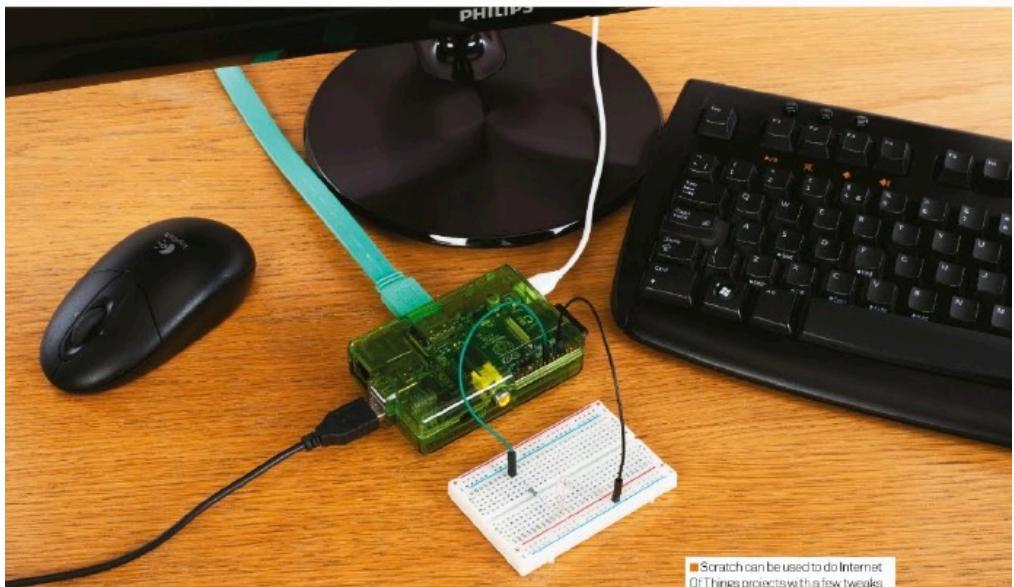
01 Installing the required software

Log into the Raspbian system with the username Pi and the password raspberry. Start the LXDE desktop environment using the command startx. Then open LXTerminal and type the following commands:

```
wget http://liamfraser.co.uk/lud/install_scratchgpio3.sh
chmod +x install_scratchgpio3.sh
sudo bash install_scratchgpio3.sh
```

This will create a special version of Scratch on your desktop called ScratchGPIO3. This is a normal version of Scratch with a Python script that handles communications between Scratch and the GPIO. ScratchGPIO was created by simplesi (cymplecy.wordpress.com).





Get interactive with Scratch

Experiment with physical computing by using Scratch to interact with buttons and lights on your Pi

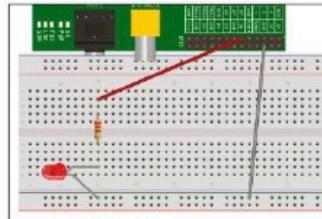
What you'll need

- Breadboard
- LEDs
- Buttons
- Resistors
- Jumper wires
- ScratchGPIO3

01 **Installing the required software**
Log into the Raspberry system with the username Pi and the password raspberry. Start the LXDE desktop environment using the command startx. Then open LXTerminal and type the following commands:

```
 wget http://lancfraser.co.uk/lxd/install_scratchpi3.sh
 chmod +x install_scratchpi3.sh
 sudo bash install_scratchpi3.sh
```

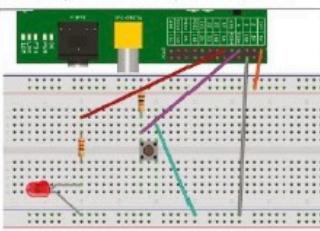
This will create a special version of Scratch on your desktop called ScratchGPIO3. This is a normal version of Scratch with a Python script that handles communications between Scratch and the GPIO. ScratchGPIO was created by simplepigameplay.wordpress.com.



02 Connecting the breadboard

Power off your Pi and disconnect the power cable. Get your breadboard, an LED, a 330-ohm resistor and two GPIO cables ready. You'll want to connect the 3.3V pin (top-right pin, closest to the SD card) to one end of the 330-ohm resistor, and then connect the positive terminal of the LED (the longer leg is positive) to the other end. The resistor is used to limit the amount of current that can flow to the LED.

Then put the negative terminal of the LED into the negative rail of the breadboard. Connect one of the GROUND pins (for example, the third pin from the right on the bottom row of pins) to this negative rail. Now connect the power to your Pi. The LED should light up. If it doesn't, then it's likely that you've got it the wrong way round, so disconnect the power, swap the legs around and then try again.



05 Wiring up our push button

Power off the Pi again. This circuit is a little bit more complicated than the LED one we created previously. The first thing we need to do is connect 3.3V (the top-right pin we used to test our LED) to the positive rail of the breadboard. Then we need to connect a 10kOhm resistor to the positive rail, and the other end to an empty track on the breadboard. Then on the same track, add a wire that has one end connected to GPIO 4. This is two pins to the right of GPIO 17. Then, on the same track again, connect one pin of the push button. Finally, connect the other pin of the push button to ground by adding a wire that is connected to the same negative rail that ground is connected to.

When the button is not pressed, GPIO 4 will be receiving 3.3V. However, when the button is pressed, the circuit to ground will be completed and GPIO 4 will be receiving 0V (and have a value of 0), because there is much less resistance on the path to ground.

We can see this in action by watching the pin's value and then pressing the button to reveal change:

```
echo 4 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio4/direction
watch -n 0.5 cat /sys/class/gpio/gpio4/value
```

You'll learn...

1. Simple circuits

If you're new to programming, Scratch is the perfect place to learn the same programming principles employed by all programming languages.

2. Coding principles

If you're new to programming, Scratch is the perfect place to learn the same programming principles employed by all programming languages.

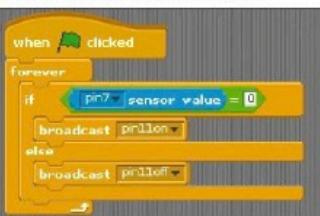
3. Physical computing

There's nothing more magical than taking code from your computer screen and turning it into a real-life effect. Your first project might just be turning on and off, but with that skill banked, the sky is the limit.



04 Controlling the LED from Scratch

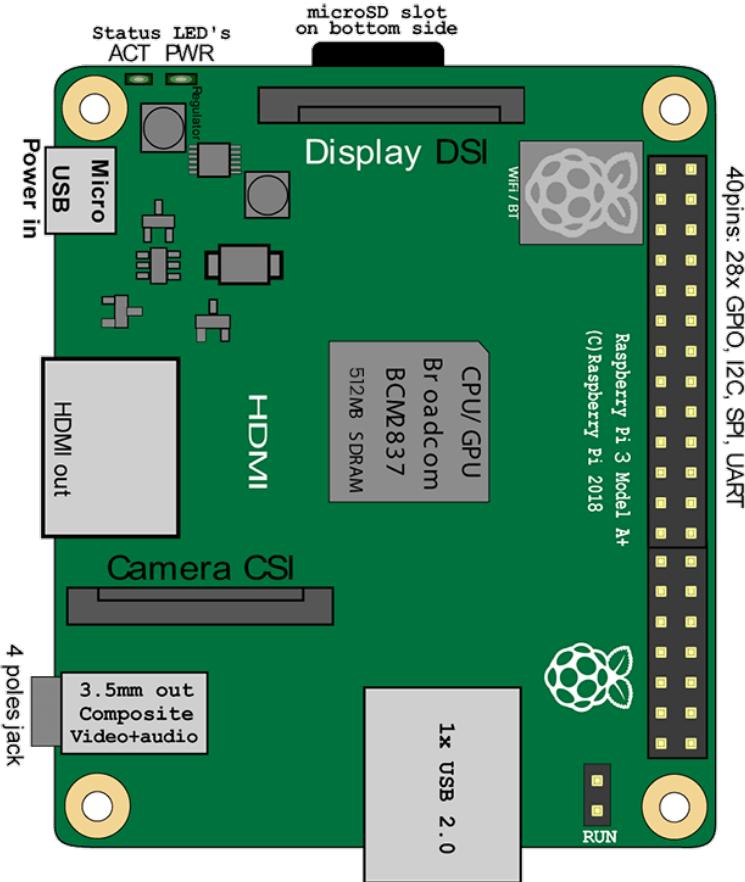
Start the LXDE desktop environment and open ScratchGPIO3. Go to the control section and create a simple script that broadcasts pin1on when Sprite1 is clicked. Then click the sprite. The LED should light up. Then add to the script to wait 1 second and then broadcast pin1off. If you click the sprite again, the LED will come on for a second and then go off. ScratchGPIO3



06 Let there be light!

Boot up the Pi and start ScratchGPIO3 as before. Go to the control section and add when green flag clicked, then attach a forever loop, and inside that an if else statement. Go to the operators section and add on if [] = [] operator to the if statement. Then go to the sensing section and add a value sensor to the left side of the equality statement, and set the value to pin7. On the right side of the equality statement, enter 0. Broadcast pin1on if the sensor value is 0, and broadcast pin1off otherwise. Click the green flag. If you push the button, the LED will light up!

Raspberry Py 3A+



Here we have included some schematics of the Raspberry Pi 3 A Plus hardware. These schematics show the general positioning of all the vital circuitry on the board. The Raspberry Pi 3 A+ is a cut down version of the Pi 3B. As you can tell by its diagram, it features a single USB 2.0 Port. It's only means of network connectivity is the inbuilt Wi-Fi.

CPU: 1.4 GHz quad core ARM Cortex-A53

GPU: 250MHz Broadcom VideoCore IV

RAM: 512mb (Shared with GPU)

Storage: Micro SD

USB 2.0 Ports: 1

USB 3.0 Ports: 0

Networking: 802.11b/g/n/ac dual band 2.4/5 GHz wireless, Bluetooth 4.2 LS BLE

Video Input: 15-pin MIPI camera interface (CSI) connector

Video Outputs: HDMI 1.3, MIPI display interface, DS1

Audio Inputs: Audio over I2S

Audio Outputs: 3.5mm phone jack, Digital Audio via HDMI

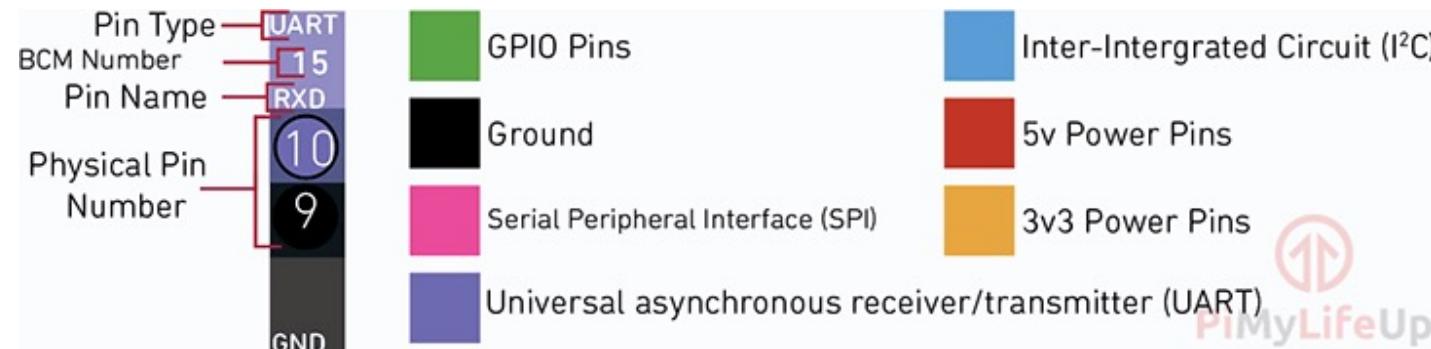
Low-Level peripherals: 17 x GPIO, +3.3v, +5v, ground, Plus the following that can be used as GPIO: UART, I2C Bus, SPI bus with two chip select, I2S audio

Power Source: 5v via MicroUSB or GPIO header

Size: 65.00mm x 56.50mm x 17mm

Weight: 23 g (0.81 oz)

GPIO (general-purpose input/output), connecting to the outside world



GPIO pins groups on the Raspberry 3

GPIO#	NAME			NAME	GPIO#
	3.3 VDC Power			5.0 VDC Power	
8	GPIO 8 SDA1 (I2C)	3	1	5.0 VDC Power	
9	GPIO 9 SCL1 (I2C)	5		Ground	
7	GPIO 7 GPCLK0	7		GPIO 15 TxD (UART)	15
	Ground	9		GPIO 16 RxD (UART)	16
0	GPIO 0	11		GPIO 1 PCM_CLK/PWM0	1
2	GPIO 2	13			
3	GPIO 3	15		Ground	
	3.3 VDC Power	17		GPIO 4	4
12	GPIO 12 MOSI (SPI)	19		GPIO 5	5
13	GPIO 13 MISO (SPI)	21		Ground	
14	GPIO 14 SCLK (SPI)	23		GPIO 6	6
	Ground	25		GPIO 10 CE0 (SPI)	10
30	SDA0 (I2C ID EEPROM)	27		GPIO 11 CE1 (SPI)	11
21	GPIO 21 GPCLK1	29		SCL0 (I2C ID EEPROM)	31
22	GPIO 22 GPCLK2	31			
23	GPIO 23 PWM1	33		Ground	
24	GPIO 24 PCM_FS/PWM1	35		GPIO 26 PWM0	26
25	GPIO 25	37			
	Ground	39		Ground	
				GPIO 27	27
				GPIO 28 PCM_DIN	28
				GPIO 29 PCM_DOUT	29

<https://pi4j.com/1.1/pins/model-3b-rev1.html>

- **Power:** Pins that are labeled 5.0v supply 5 volts of power and those labeled 3V3 supply 3.3 volts of power. There are two 5V pins and two 3V3 pins.
- **GND:** These are the ground pins. There are eight ground pins.
- **Input/Output pins:** These are the pins labeled with the # sign, for example, #17, #27, #22, etc. These pins can be used for input or output.
- **I2C:** I2C is a serial protocol for a two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces, and other similar peripherals in embedded systems. These pins are labeled **SDA** and **SCL**.
- **UART:** The **Universal Asynchronous Receiver/Transmitter** allows your Raspberry Pi to be connected to serial peripherals. The UART pins are labeled **TxD** and **RxD**.
- **SPI:** The **Serial Peripheral Interface** is a synchronous serial communication interface specification used for short distance communication, primarily in embedded systems. The SPI pins are labeled **MOSI**, **MISO**, **SCLK**, **CE0**, and **CE1**.
- **ID EEPROM:** **Electrically Erasable Programmable Read-Only Memory** is a user-modifiable read-only memory that can be erased and written to repeatedly through the application of higher than normal electrical voltage. The two EEPROM pins on the Raspberry Pi (**EED** and **EEC**) are also secondary I2C ports that primarily facilitate the identification of Pi Plates (e.g., Raspberry Pi Shields/Add-On Boards) that are directly attached to the Raspberry Pi.

GPIO general purpose IO

Voorgedefinieerde pennen

- 0V, 3.3V, 5V, Transmit, Receive
- I2C, 1-Wire

Vrijbeschikbare pennen

- GPIO 4, 17, 18, 8, 7

Naamgeving

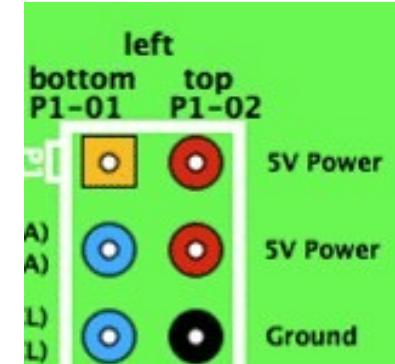
- Pinnummer van connector P1
- Broadcom documentatie van de SoC
(System on Chip)



GPIO

Voordat je iets aansluit

- input maximaal 3.3V
 - let met name op de 5V van pin 2 en 4
- output maximaal 16 mA
 - LED over $330\ \Omega$ weerstand mag
 - motortje heeft een buffer nodig (ULN2003)
- totale output maximaal 50 mA



Programmeren

Schrijven (led laten knipperen)

- Eerst: GPIO pennetje definieren als output
- Dan: herhaaldelijk schrijven

Pennetje 26 = GPIO 7

Lezen (schakelaar uitlezen)

- Eerst: GPIO pennetje definieren als input
- Dan: herhaaldelijk lezen

Pennetje 24 = GPIO 8

Programmeren (bash shell)

- bash is de Linux command line interpreter
- als root in directory /sys/class/gpio werken
- GPIO 7 voor uitvoer

```
echo "7" >/sys/class/gpio/export
```

```
echo "out" >/sys/class/gpio/gpio7/direction
```

- led aan (1), led uit (0)

```
echo "1" > /sys/class/gpio/gpio7/value
```

```
echo "0" > /sys/class/gpio/gpio7/value
```

Programmeren (bash shell)

Knipperen

```
while sleep 0.5
do echo "1" > /sys/class/gpio/gpio7/value sleep 0.5
    echo "0" > /sys/class/gpio/gpio7/value done
```

GPIO 8 voor invoer

```
echo "8" > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio8/direction
```

eenmalig lezen

```
cat /sys/class/gpio/gpio8/value
```

Programmeren (bash shell)

herhaald lezen

```
while sleep 0.1
do cat /sys/class/gpio/gpio8/value
done
```

opruimen

```
echo "7"      > /sys/class/gpio/unexport
echo "8"      > /sys/class/gpio/unexport
```

Programmeren (python)

Pi in Raspberry Pi staat voor Python

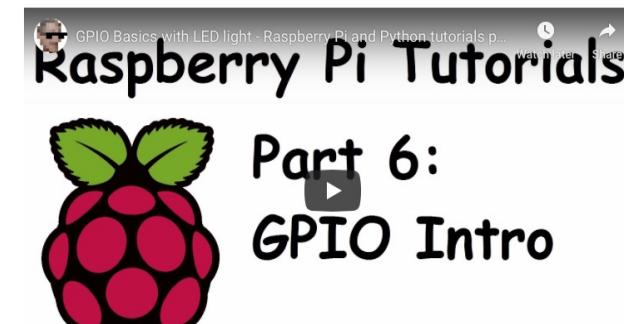
Standaard modules en bibliotheken

- keuze in naamgeving pennetjes / meer functies
- <https://pythonprogramming.net/gpio-raspberry-pi-tutorials/>

RPi.GPIO

- standaard onderdeel van Raspbian
- <https://pypi.org/project/RPi.GPIO>

GPIO (General Purpose Input Output) Pins - Raspberry Pi tutorial



Preview lesson five

Making sense of Data (Types)

THE [Data Science] HYPE

There's huge and growing demand especially in business

Predicting the future take a lot of effort & expertise

Because of the hype, everyone wants to “own” data science!

– many of them are just selling their stuff with a new label

Here, I will ignore the hype and talk about Data
and how to use it for IoT-projects

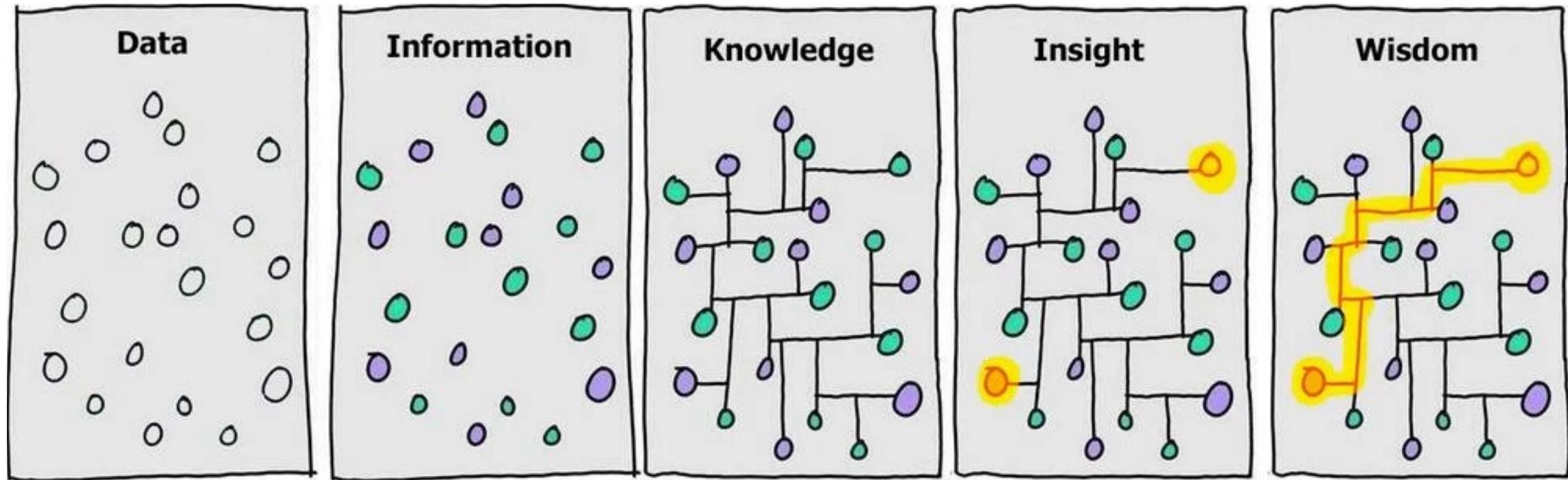
Data Science is about:

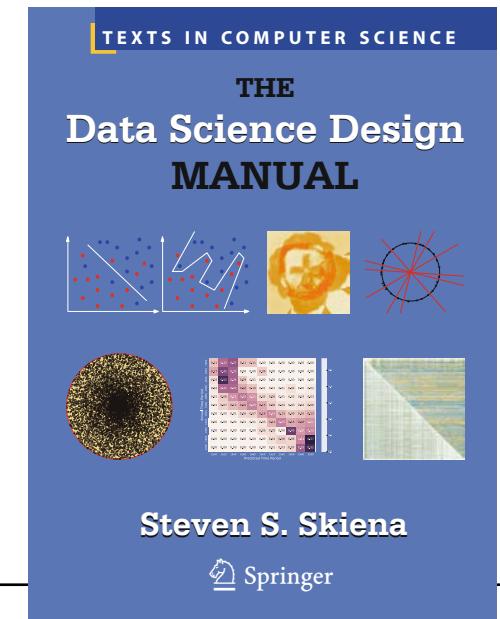
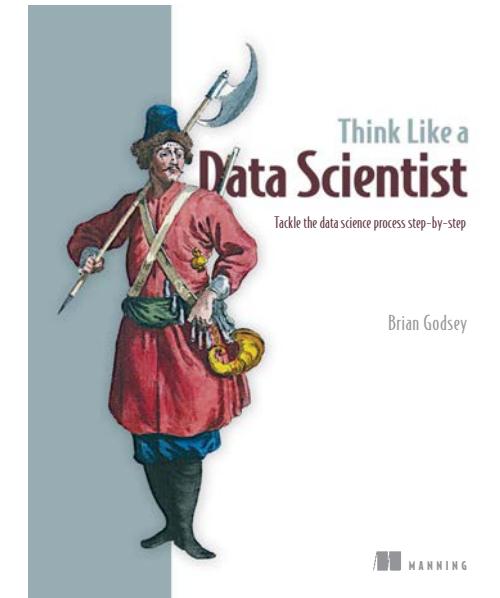
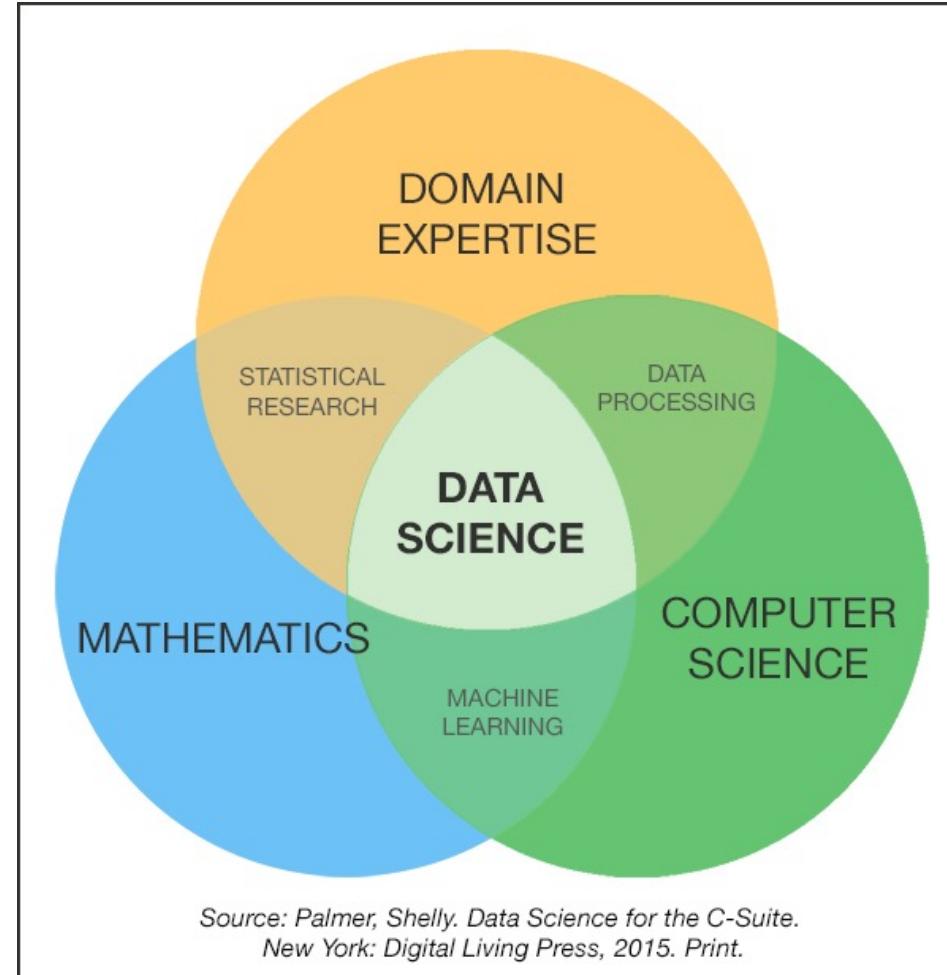
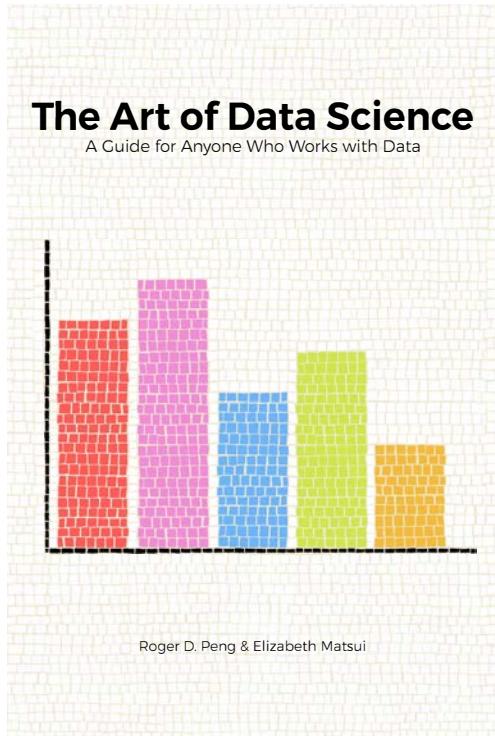
A Data Scientist can:

- *understand* the background domain
- *design* solutions that produce added value to the organization
- *implement* the solutions efficiently
- *communicate* the findings clearly (important!)

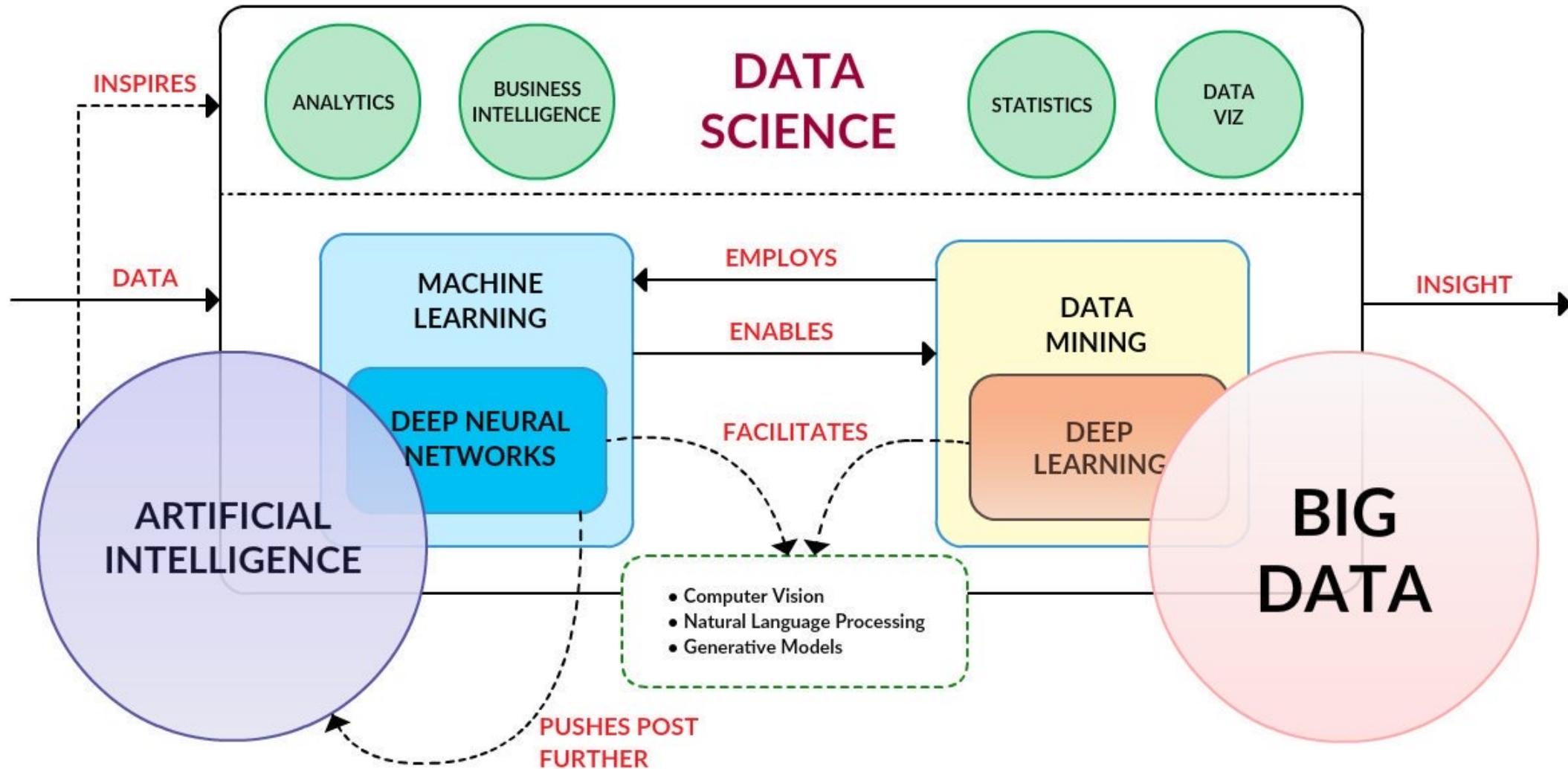
Data Scientist is a *practitioner* with sufficient expertise in software engineering, statistics/machine learning, and the application domain.

How to make sense of IoT Data (Science)

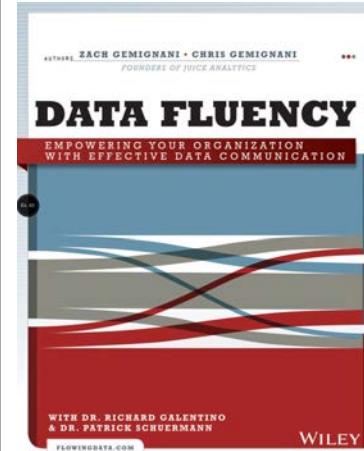
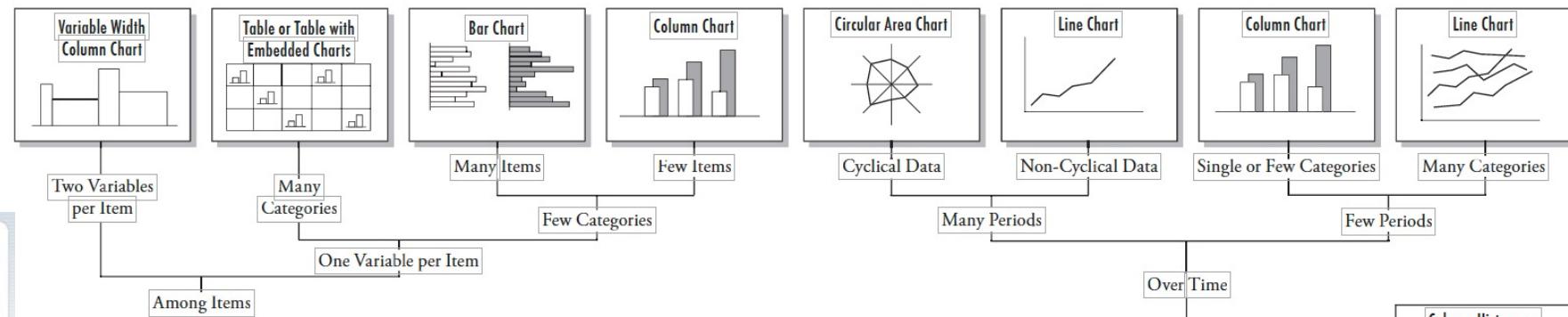




Wat geeft grip op DATA?



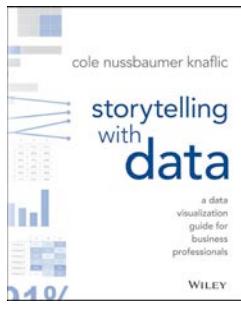
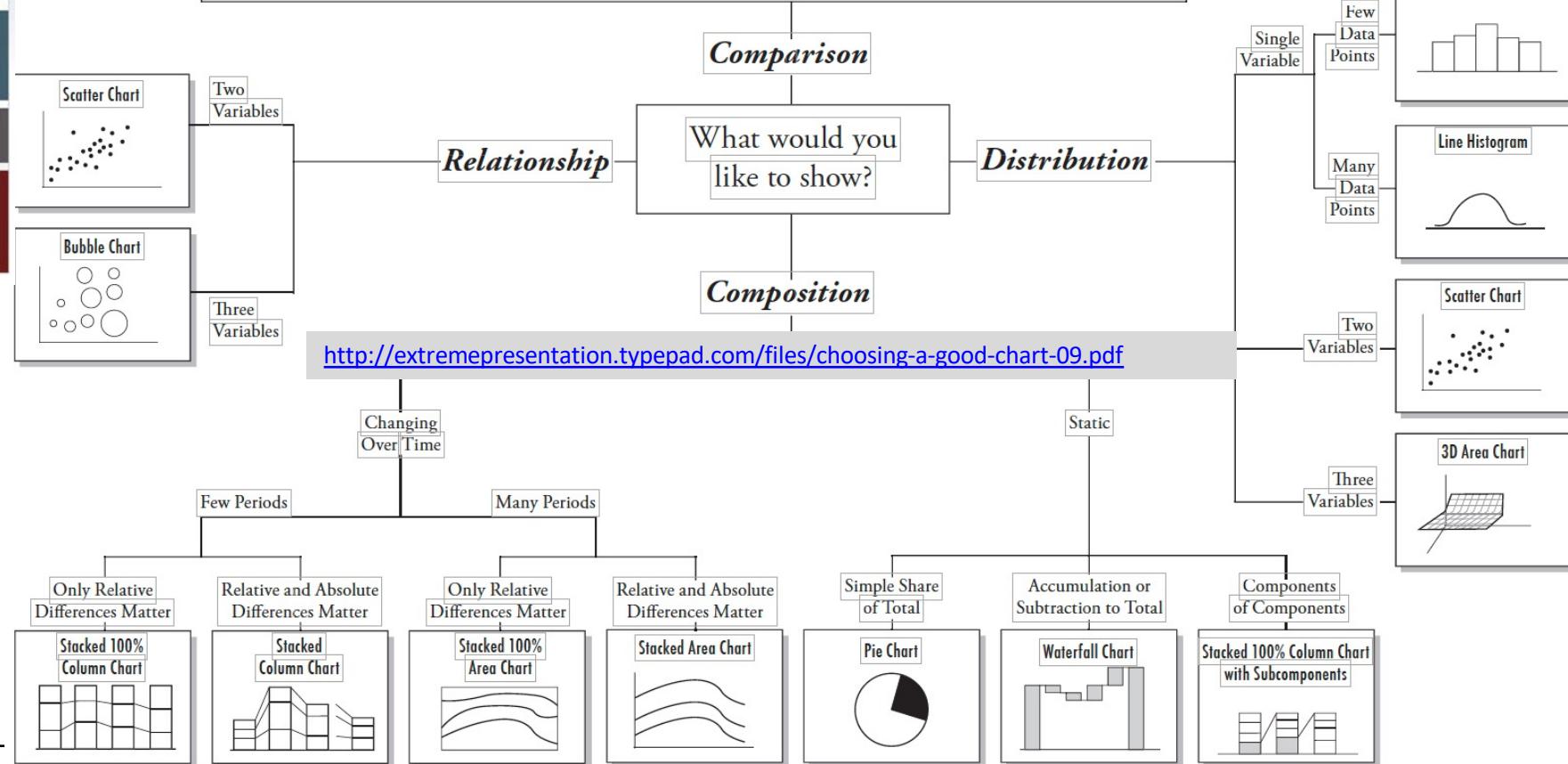
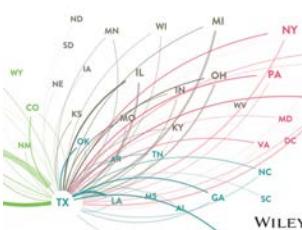
<http://labs.juiceanalytics.com/chartchooser/index.html>



Graph Analysis and Visualization

Discovering Business Opportunity in Linked Data

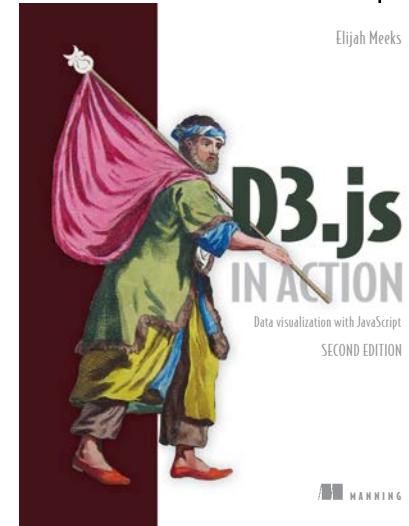
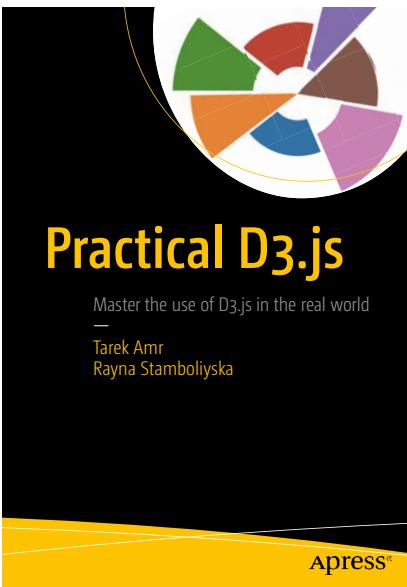
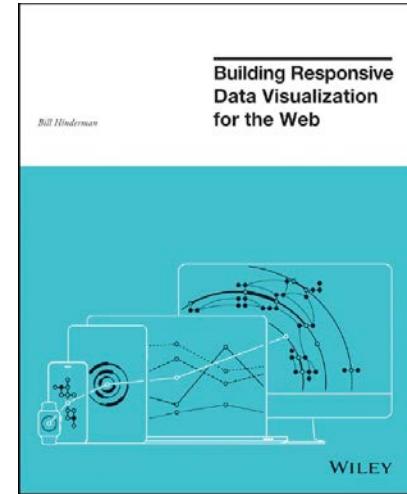
Richard Brath and David Jonker



Wat geeft grip op DATA?

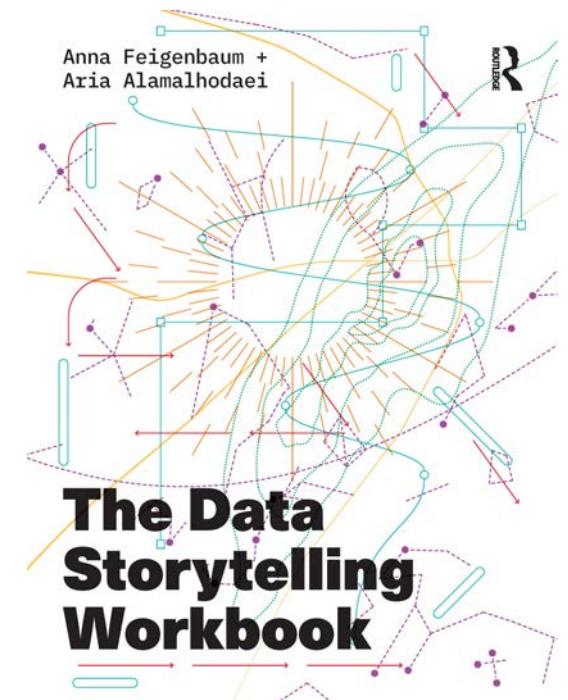
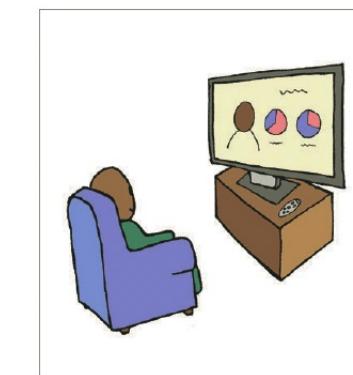
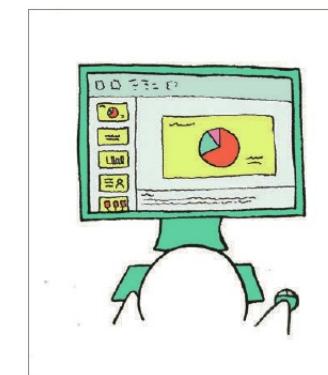
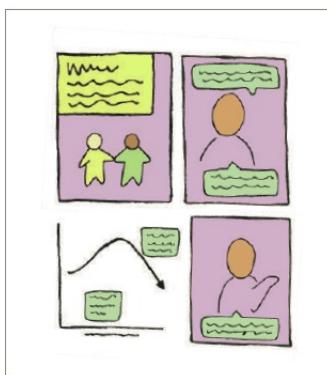
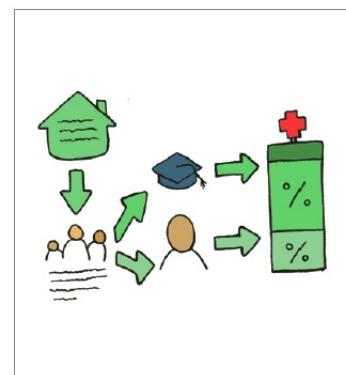
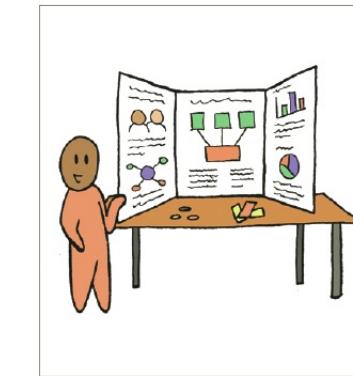
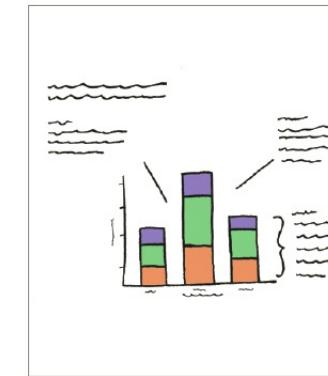


Wat geeft grip op DATA?



Data Types

Data VIZ → Top-down **Cognition**
requires a Narrative (Story Telling)



DATA-DRIVEN IoT: WHAT IS DATA?

Data [gegevens]

Raw Facts

No Context

Numbers

Symbols

Data comes from the Latin word, "datum," meaning a "thing given."

Although the term "data" has been used since as early as the 1500s, modern usage started in the 1940s and 1950s as practical electronic computers began to input, process, and output data.

98734975471894614398734578

20875980542158009258202908

12349823094823048002343423

98734975471894614398734578

20875980542158009258202908

12349823094823048002343423

Can you find the
the mistake?

1 2 3 4 5 6 7 8 9

TYPES OF DATA: Quantitative versus Qualitative [numerical vs categorical]

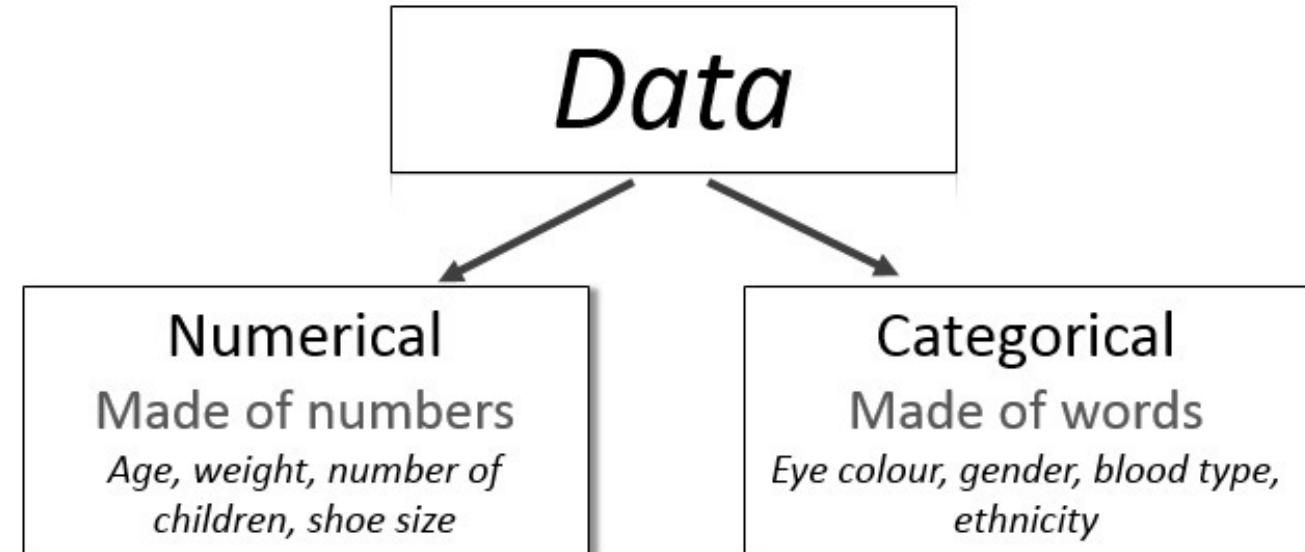
Data Quantification

Quantitative [Numerical] data:

This data can be described using **numbers**, and basic mathematical procedures, including addition, are possible on the set. It can be **discrete** (countable numbers) or **continuous** (infinitely large or small)

Qualitative [Categorical] data:

This data are categories. It cannot be described using numbers and basic mathematics. Is generally thought of as being described using "**natural**" categories and language.

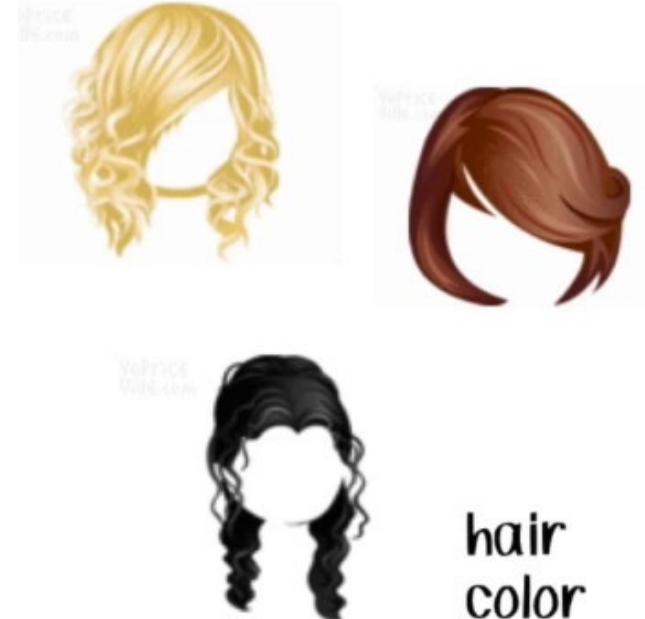


- Quantitative values
 - **Measure** things
 - *Revenue, Units, Marketshare, Duration, Customer Satisfaction, Visits, Price, etc.*
- Categorical values
 - Subdivide things into **groups**
 - *Region, product, category, employee, etc.*

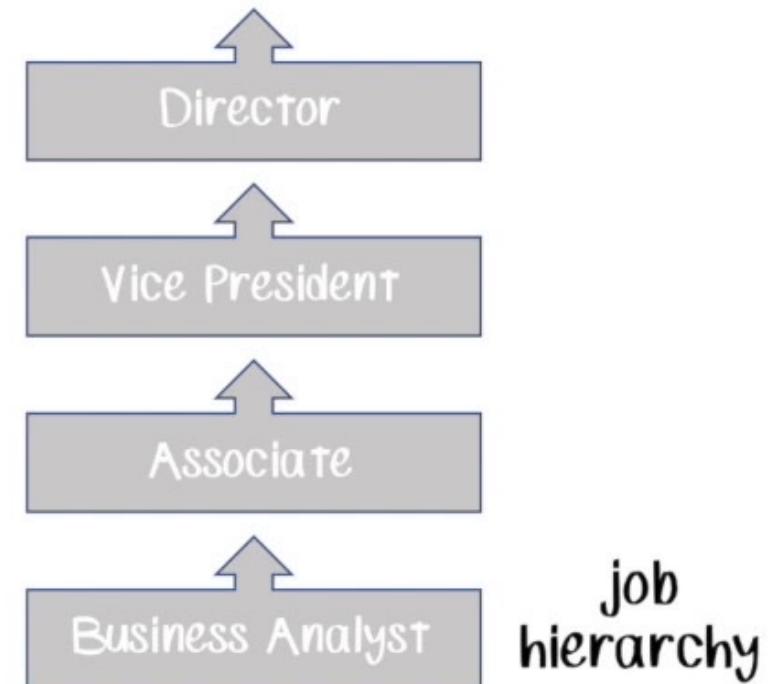


CATEGORICAL DATA

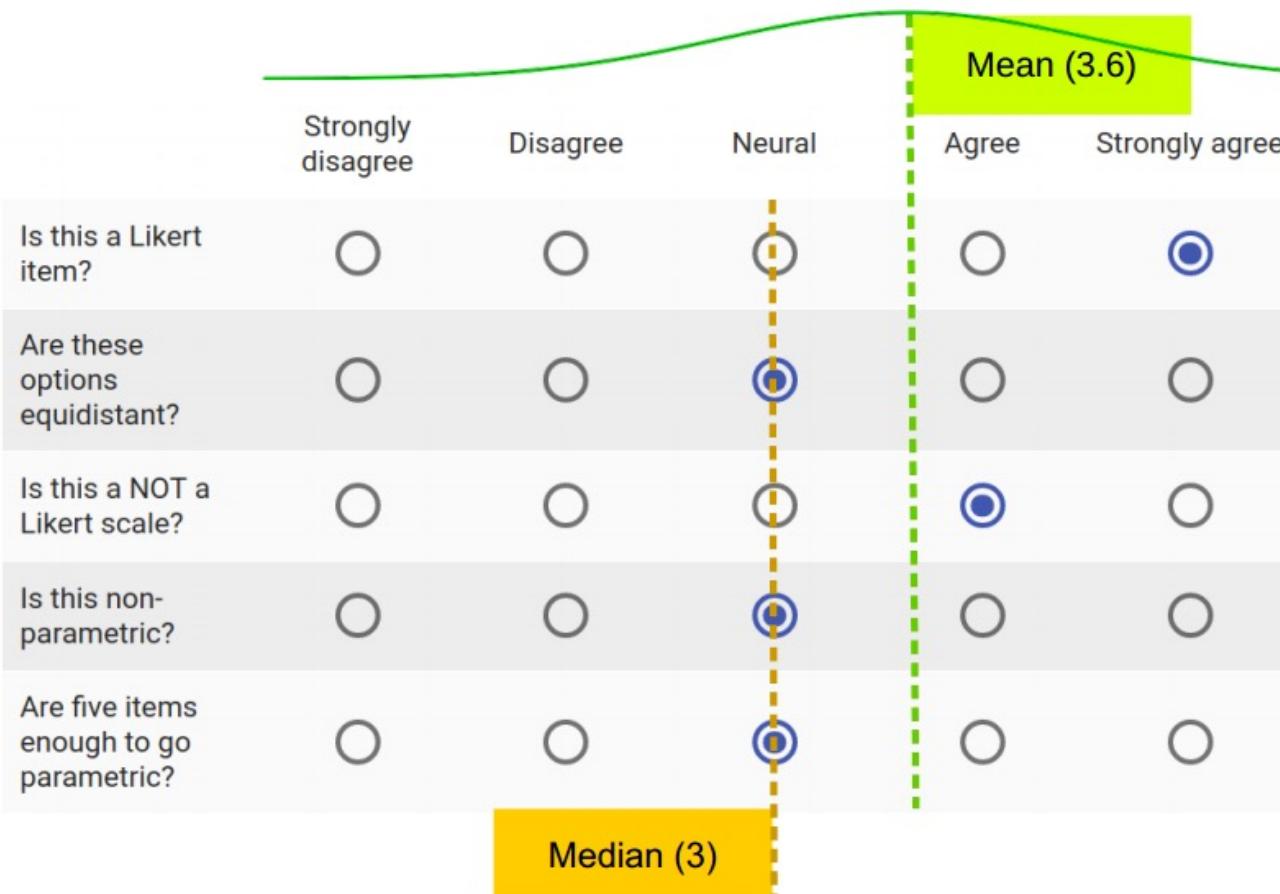
NOMINAL DATA



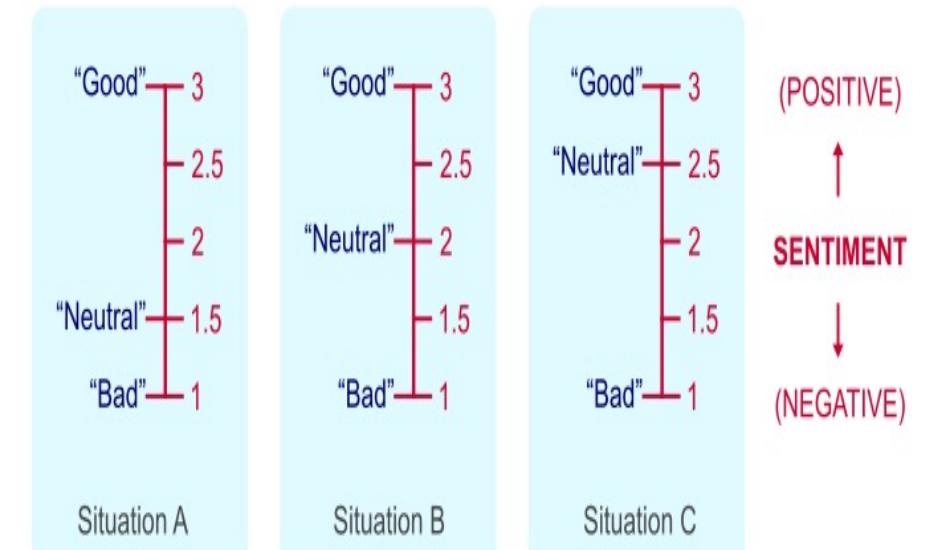
ORDINAL DATA



Likert-scale data



ORDINAL VARIABLE - INTERVALS ARE UNKNOWN

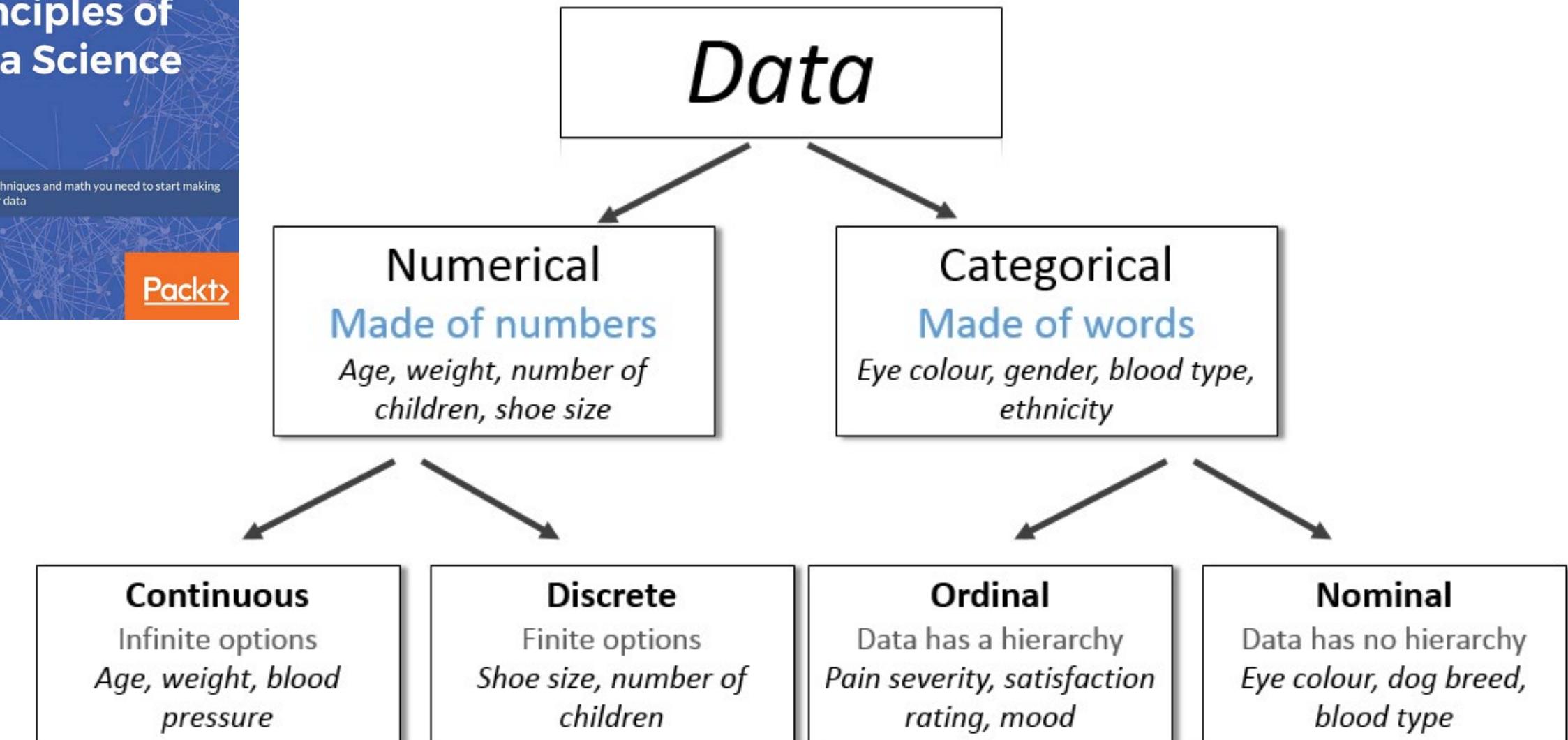


Principles of Data Science

Learn the techniques and math you need to start making sense of your data



Packt



LEVELS OF DATA: LEVELS OF MEASUREMENTS/OBSERVATIONS

We onderscheiden 4 meetniveaus:

nominaal + ordinaal [discrete data]

interval + ratio [continue data]

Meetniveaus / Meetschalen:

Wanneer je onderzoek doet heb je vaak **variabelen** die je hierin moet verwerken.

Variabelen zijn elementen uit een onderzoek die verschillende waarden kunnen aannemen. Deze waarden kunnen worden gecategoriseerd in verschillende meetniveaus.

Meetniveaus kunnen iets vertellen over welke data-analyse geschikt is voor structurering.

LEVELS OF DATA: LEVELS OF MEASUREMENTS/OBSERVATIONS

Meetniveau	Wat je kunt berekenen met behulp van waarden op het meetniveau
Nominaal	Tellen, percentages berekenen
Ordinaal	Tellen, percentages berekenen en hoger/lager aangeven
Interval	Tellen, hoger/lager aangeven, verschillen in eenheden aangeven, gemiddelde, spreiding
Ratio	Tellen, hoger/lager aangeven, verschillen in eenheden aangeven, gemiddelde, spreiding en het berekenen van verhoudingen

LEVELS OF DATA: LEVELS OF MEASUREMENTS/OBSERVATIONS

Meetniveaus [level] /Meetschalen [scale]:

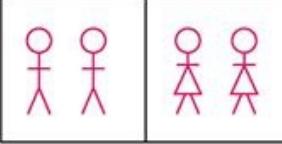
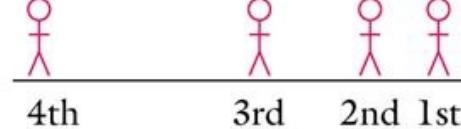
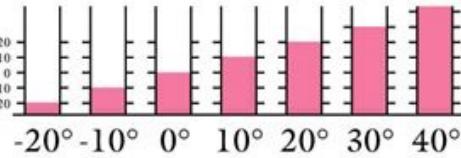
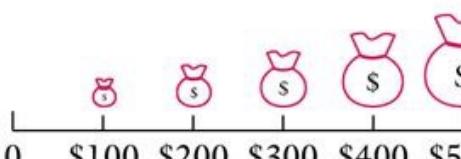
De hoogte van het meetniveau is bepalend voor:

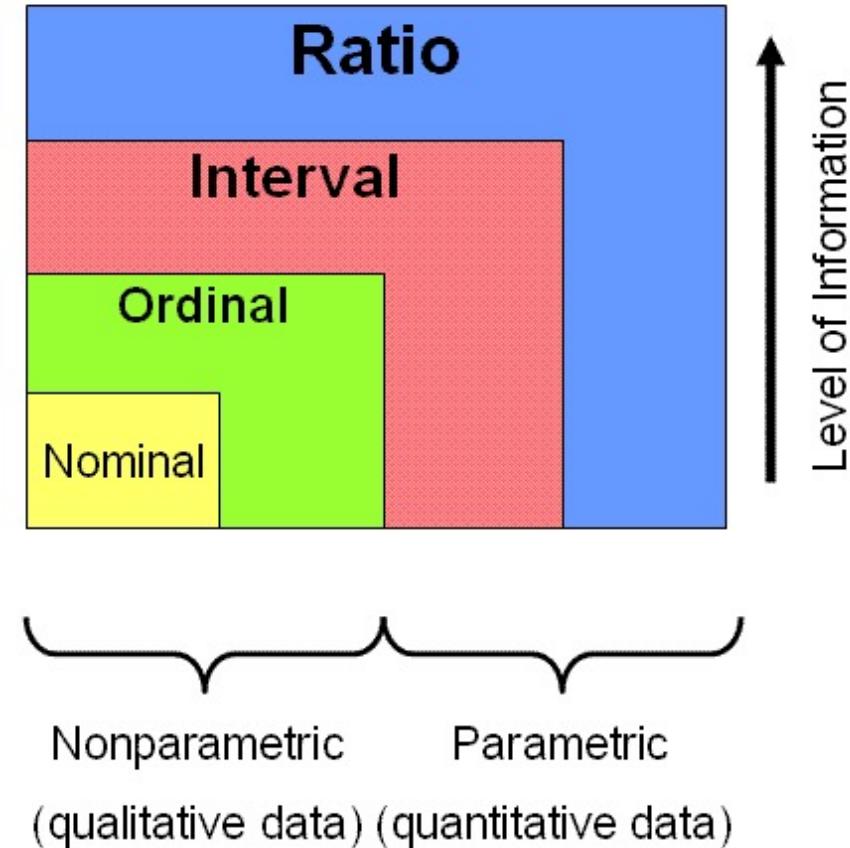
Statische-analyse + Grafische weergave

Meetniveaus & hun kenmerken

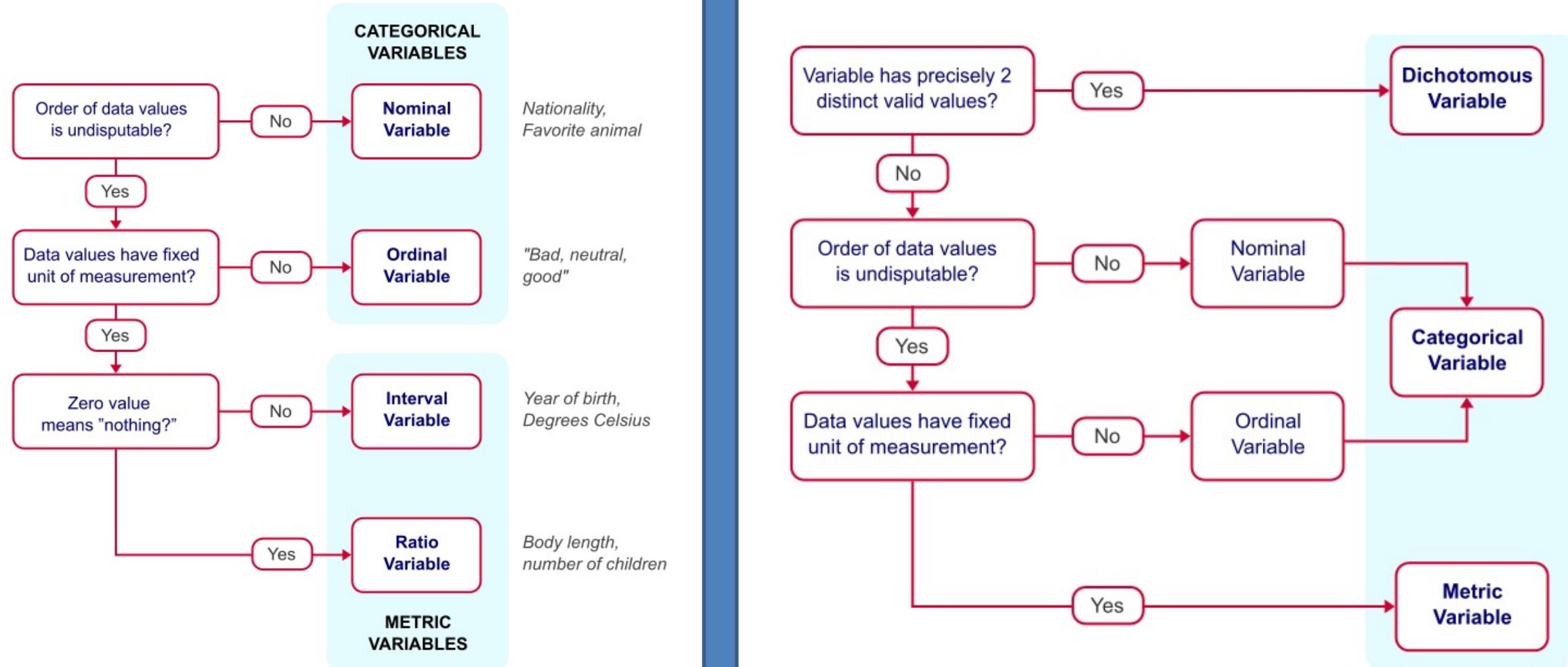
		Scale	Rationiveau
	Ordinaal niveau	Intervalniveau	Verhouding blijven gelijk
Nominale niveau	Ordening	Ordening	Gelijke verschillen
Onderscheid	Onderscheid	Onderscheid	Ordening
Geslacht	Opleidingsniveau	Intelligentie	Onderscheid
			Leeftijd

LEVELS OF DATA: LEVELS OF MEASUREMENTS/OBSERVATIONS

SCALE	EXAMPLE
Nominal	 Gender
Ordinal	 Position in race
Interval	 Temperature (in Fahrenheit)
Ratio	 Money



LEVELS OF DATA: What data to measure or observe?



LEVELS OF DATA: What data to measure or observe?

Differences between measurements, true zero exists

Ratio Data

Differences between measurements but no true zero

Interval Data

Ordered Categories (rankings, order, or scaling)

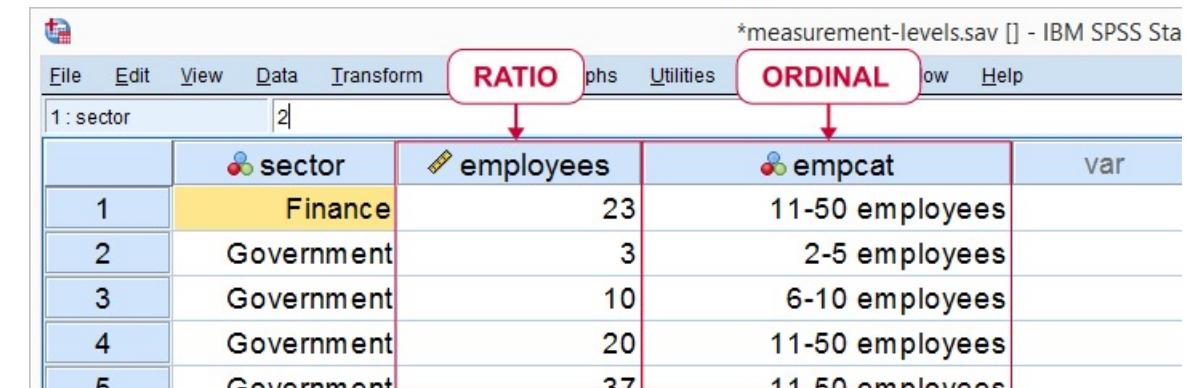
Ordinal Data

Categories (no ordering or direction)

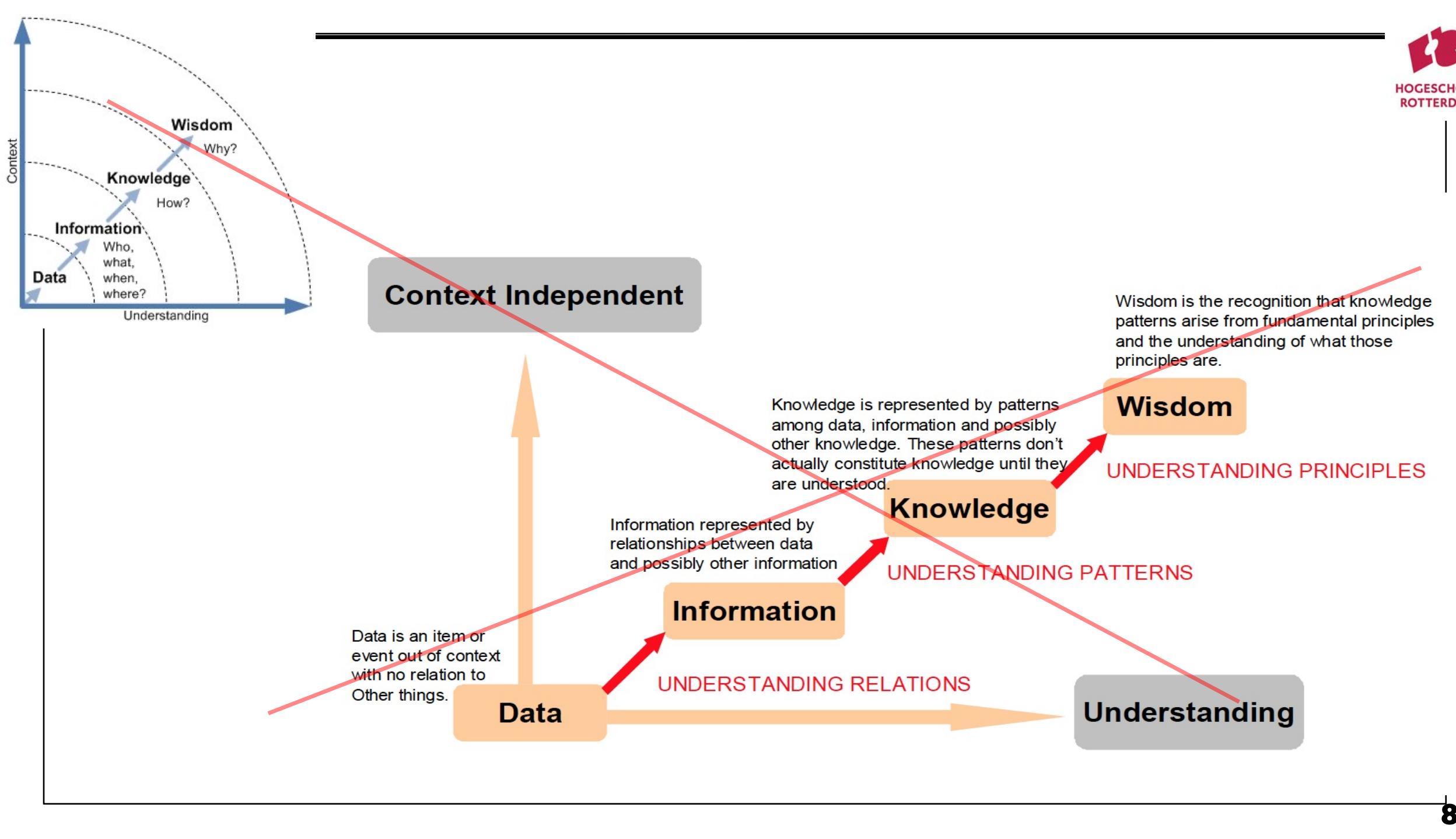
Nominal Data

Quantitative Data

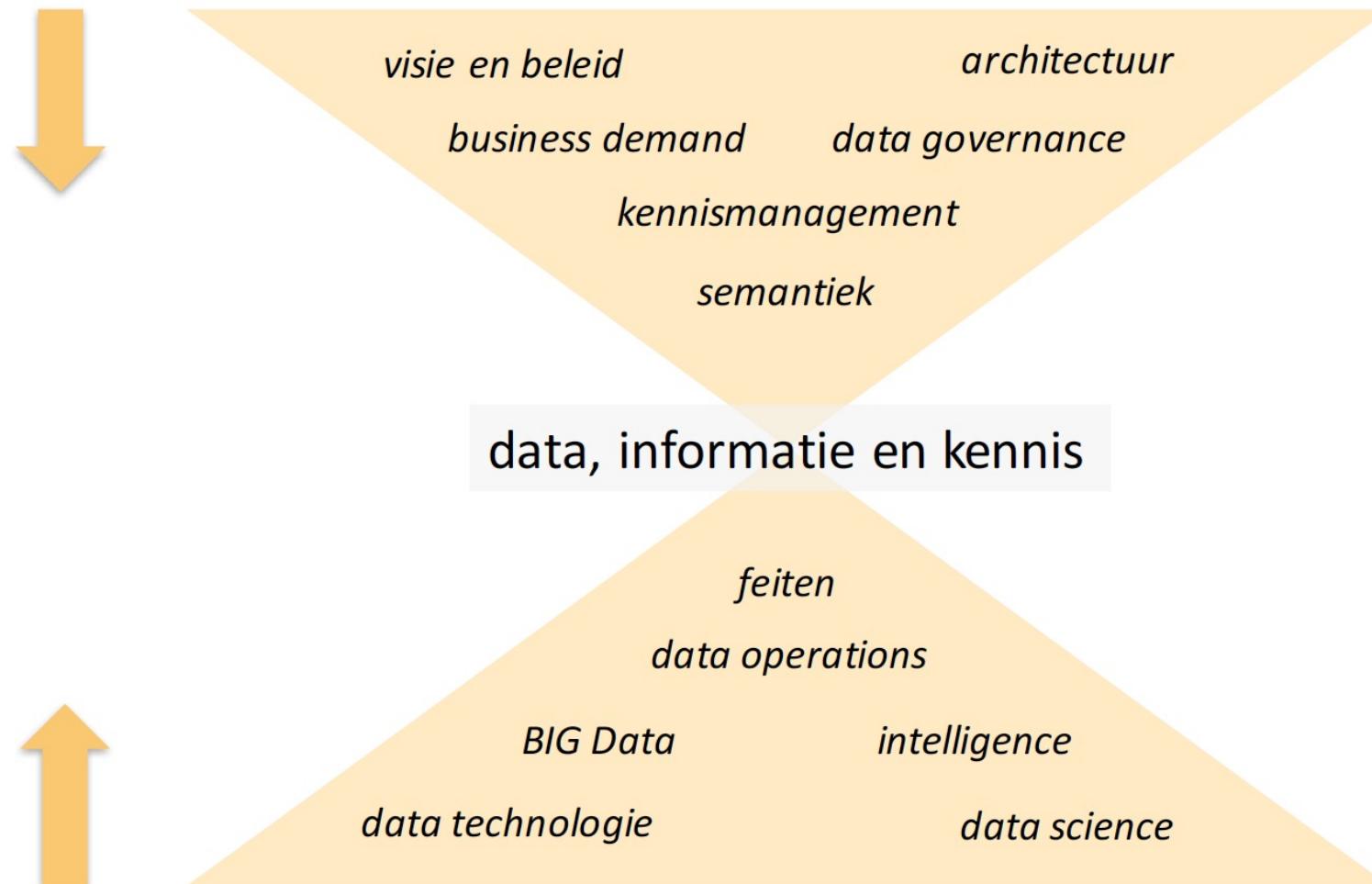
Qualitative Data



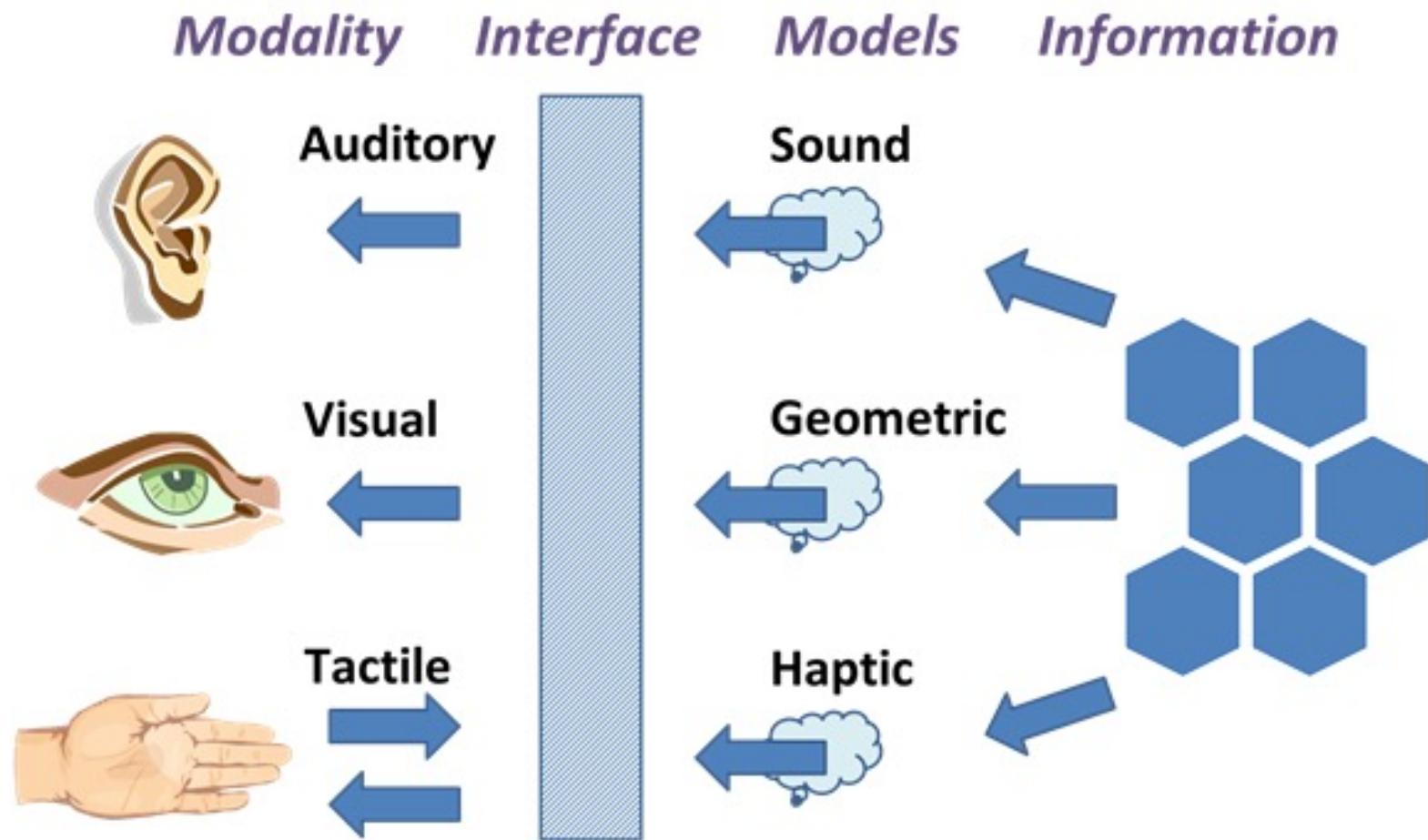
sector	employees	empcat
Finance	23	11-50 employees
Government	3	2-5 employees
Government	10	6-10 employees
Government	20	11-50 employees
Government	37	11-50 employees



Data, informatie en kennis is wat ons verbindt



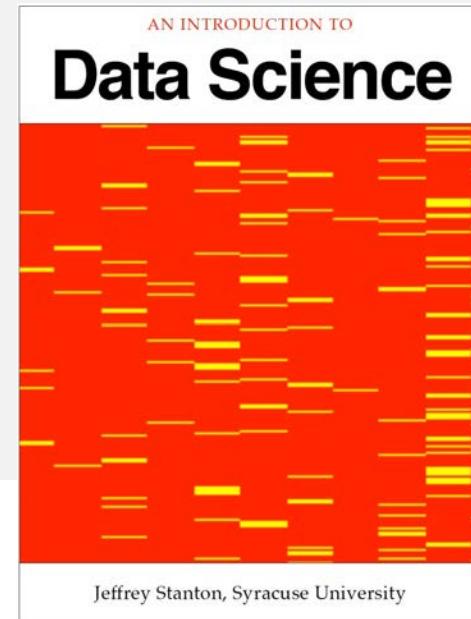
<http://intelligence.agconnect.nl/content/van-data-naar-informatie>



DATA-DRIVEN: Data versus Information

Data [gegevens]

Raw Facts
No Context
Numbers
Symbols

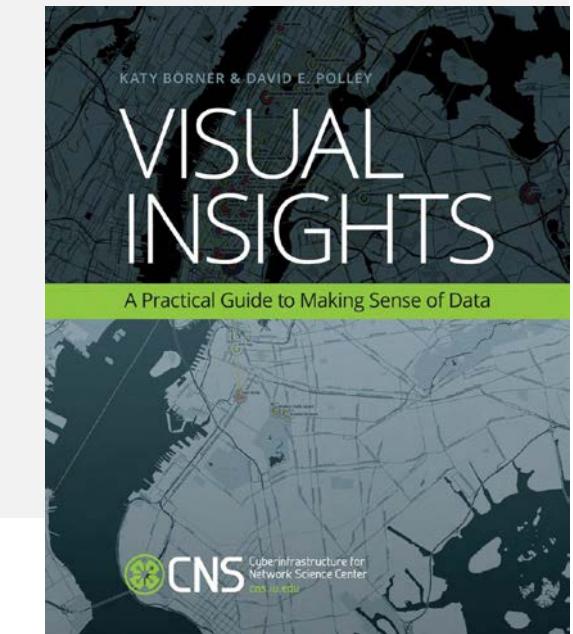


Information

Data with structure = processed data

Value-added to Data
through

- Summarisation
- Organisation
- Analysing



TYPES OF DATA: Structured vs Unstructured [organized vs unorganized]

Data Structuring

Structured (organized) data:

This is data that can be thought of as observations and characteristics. It is usually organized using a table method (rows and columns).

Structured Data

High Degree of organization, such as a relational database

Column	Value
Patient	Joe Brown
Date of Birth	02/13/1972
Date Admitted	02/05/2014

Unstructured Data

Information that is difficult to organize using traditional mechanisms

Unstructured (unorganized) data:

This data exists as a free entity and does not follow any standard organization hierarchy.

"The patient came in complaining of chest pain, shortness of breath, and lingering headaches...smokes 2 packs a day... family history of heart disease...has been experiencing similar symptoms for the past 12 hours...."

DATA STRUCTURING: Generalized Form of a Data Table

Data Table [DATA MATRIX]

A generalized version of the data table is shown.

This table can represent any number of observations described over multiple variables.

This table describes a series of observations (from o_1 to o_n) where each observation is described using a series of variables (from x_1 to x_p). A value is provided for each variable of each observation.

	Variables					
Observations	x_1	x_2	x_3	...	x_p	
o_1	x_{11}	x_{12}	x_{13}	...	x_{1p}	
o_2	x_{21}	x_{22}	x_{23}	...	x_{2p}	
o_3	x_{31}	x_{32}	x_{33}	...	x_{3p}	
...
o_n	x_{n1}	x_{n2}	x_{n3}	...	x_{np}	

Most data that exists in text form, including server logs and Facebook posts, is unstructured

Scientific observations, as recorded by careful scientists, are kept in a very neat and organized (structured) format: THE DATA TABLE

A genetic sequence of chemical nucleotides [ACGTATTGCA] is unstructured even if the order of the nucleotides matters

DATA STRUCTURING: Observations versus Variables

Data Table [DATA MATRIX]

A generalized version of the data table is shown. This table can represent any number of **observations** described over multiple **variables**.

This table describes a series of observations (from o₁ to o_n) where each observation is described using a series of variables (from x₁ to x_p). A value is provided for each variable of each observation.

Patient ID	Treated	Age	Outcome	Random
1	Yes	Young	Positive	0.24
2	No	Young	Positive	0.85
3	Yes	Old	Negative	0.64
4	No	Old	Negative	0.70
5	No	Old	Negative	0.87
6	No	Old	Negative	0.72
7	No	Old	Negative	0.86
8	No	Young	Negative	0.16
9	No	Young	Positive	0.17

Observations	Variables				
	x ₁	x ₂	x ₃	...	x _p
o ₁	x ₁₁	x ₁₂	x ₁₃	...	x _{1p}
o ₂	x ₂₁	x ₂₂	x ₂₃	...	x _{2p}
o ₃	x ₃₁	x ₃₂	x ₃₃	...	x _{3p}
...
o _n	x _{n1}	x _{n2}	x _{n3}	...	x _{np}

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

variables

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

observations

STRUCTURED DATA FORMATS

- CSV, comma separated values

```
sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
```

- hierarchies, e.g., Newick tree format

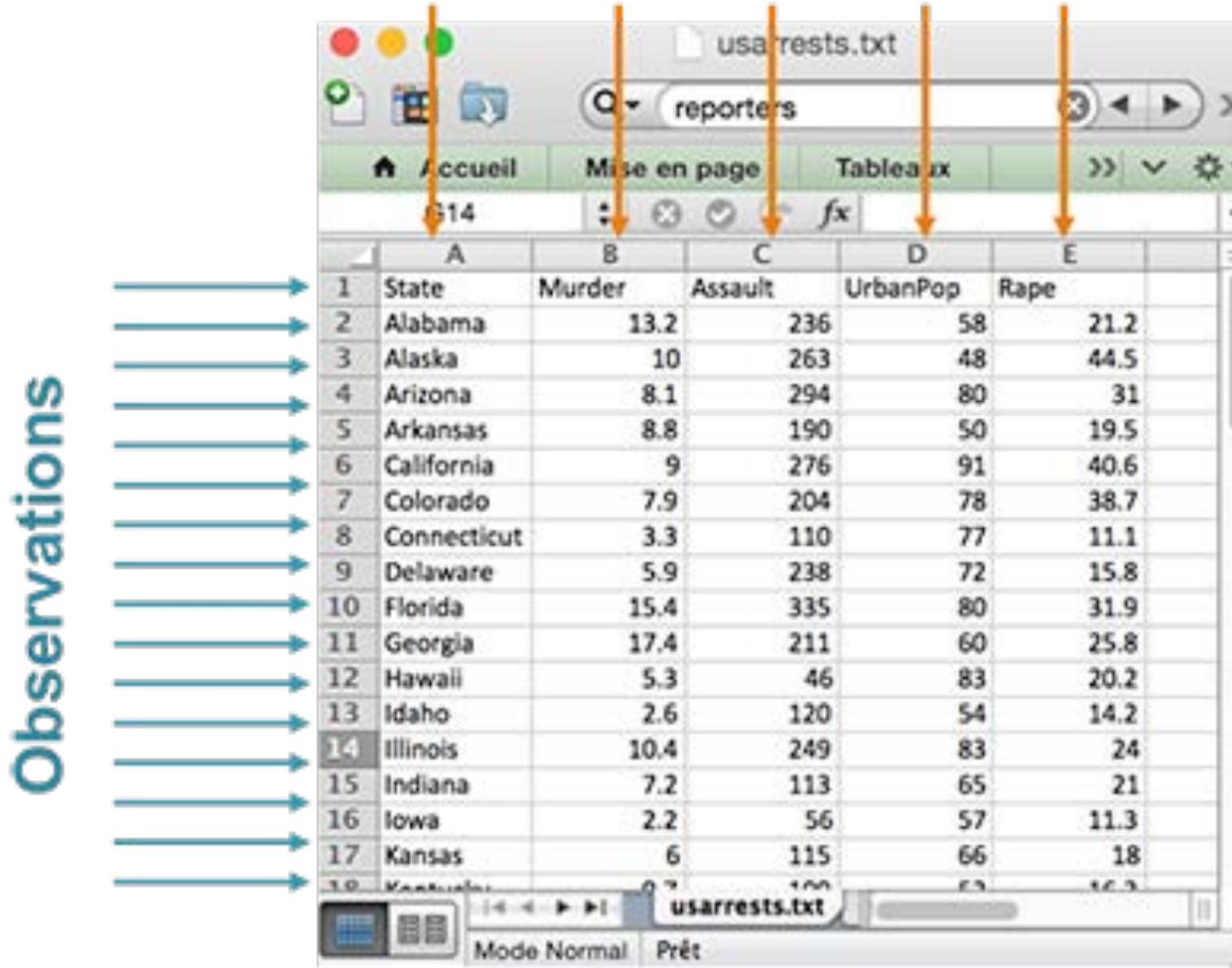
```
(A,B,(C,D)E)F;
```

- networks, e.g., GraphViz (DOT)

```
digraph graphname {
    a -> b -> c;
    b -> d;
}
```

Tidy data

Variables



Observations

	A	B	C	D	E
1	State	Murder	Assault	UrbanPop	Rape
2	Alabama	13.2	236	58	21.2
3	Alaska	10	263	48	44.5
4	Arizona	8.1	294	80	31
5	Arkansas	8.8	190	50	19.5
6	California	9	276	91	40.6
7	Colorado	7.9	204	78	38.7
8	Connecticut	3.3	110	77	11.1
9	Delaware	5.9	238	72	15.8
10	Florida	15.4	335	80	31.9
11	Georgia	17.4	211	60	25.8
12	Hawaii	5.3	46	83	20.2
13	Idaho	2.6	120	54	14.2
14	Illinois	10.4	249	83	24
15	Indiana	7.2	113	65	21
16	Iowa	2.2	56	57	11.3
17	Kansas	6	115	66	18
18	Massachusetts	12.5	296	88	32.0

KINDS OF DATA

- STRUCTURED DATA
 - lists
 - $n \times p$ tables, arrays
 - hierarchies
 - (e.g., organization chart)
 - networks
 - (e.g., travel routes,
 - hypertext = links)
- Generic data-interchange formats:
XML, JSON
- UNSTRUCTURED DATA
 - text
 - images
 - video
 - sound
- Often can be made structured by, e.g., parsing language, segmenting images, etc.

BIG DATA

- A crucial part of the rise of Data Science is the steep increase in the amount and availability of data
- Big Data refers not only to the quantity but also to the quality of the data:
 - VOLUME: lots of it
 - VELOCITY: fast (streaming)
 - VARIETY: all kinds, not nice and “clean”
 - VERACITY: can it be trusted?

JSON

- Similar to XML but simpler

```
{  
    "firstName": "John",  
    "lastName": "Smith",  
    "isAlive": true,  
    "age": 25,  
    "address": {  
        "streetAddress": "21 2nd Street",  
        "city": "New York",  
        "state": "NY",  
        "postalCode": "10021-3100"  
    },  
    "phoneNumbers": [  
        {  
            "type": "home",  
            "number": "212 555-1234"  
        },  
        {  
            "type": "office",  
            "number": "646 555-4567"  
        },  
        {  
            "type": "mobile",  
            "number": "123 456-7890"  
        }  
    "children": [],  
    "spouse": null  
}
```

XML

- same example:

```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumber>
    <type>home</type>
    <number>212 555-1234</number>
  </phoneNumber>
  <phoneNumber>
    <type>fax</type>
    <number>646 555-4567</number>
  </phoneNumber>
  <gender>
    <type>male</type>
  </gender>
</person>
```

PARSING

- Given a known grammar, unstructured text data can be parsed
- “It ain’t over till the fat lady sings”
((it, (ain't, over)), (till, ((the, (fat, lady)), sings)))
- Similarly, images can be segmented into parts

PARSING

- Similarly, images can be segmented into parts



(a) Detection



(b) Instance Segmentation



(c) Human Parsing

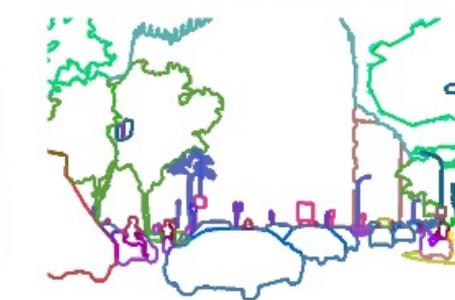
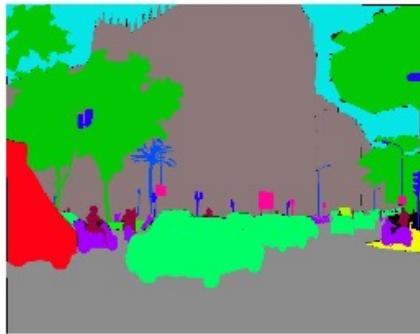


(d) Multi-Human Parsing



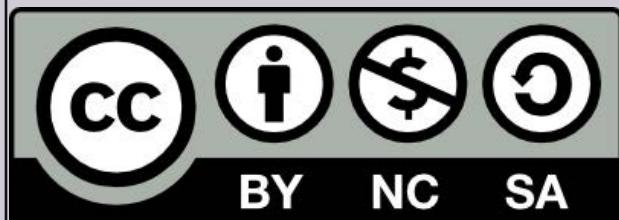
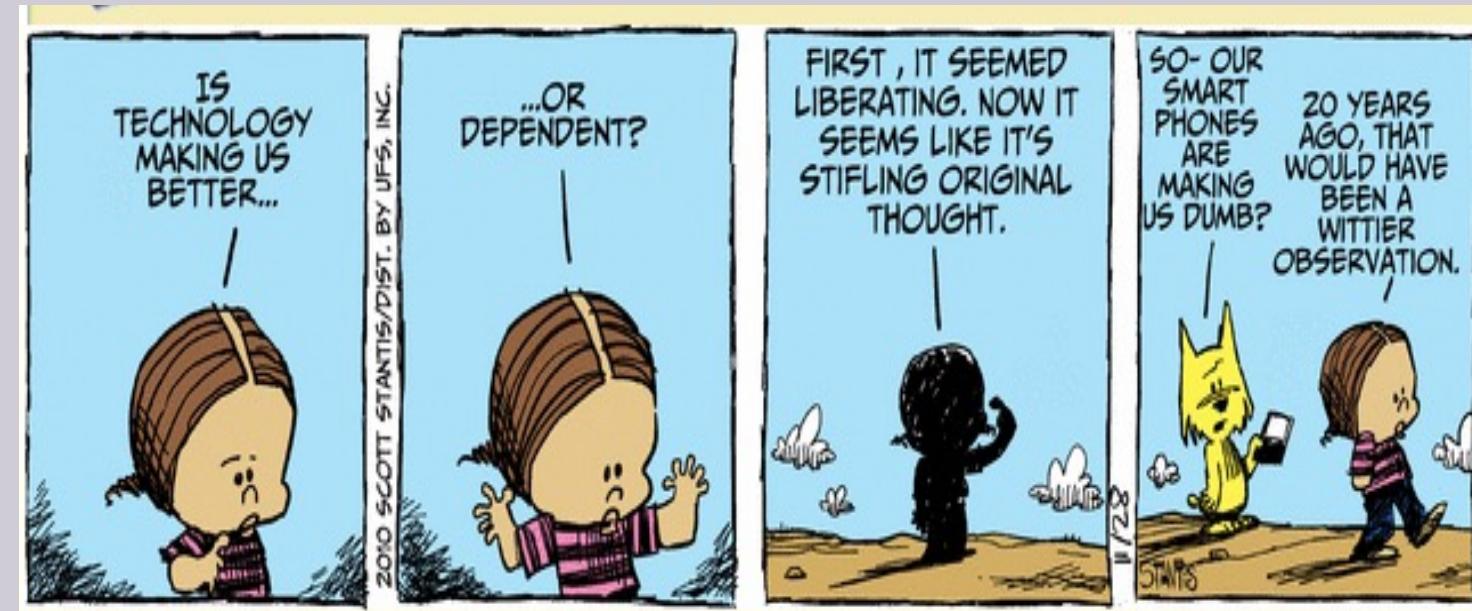
PARSING

- Similarly, images can be segmented into parts



<http://creativecommons.org/licenses/by-nc-sa/3.0/>

These materials are licensed under a Creative Commons Attribution-Share-Alike license. You can change it, transmit it, show it to other people. Just always give credit to RFvdW.



This seminar was developed by:
Rob van der Willigen
LivingLab AI & Ethics
Hogeschool Rotterdam
November 2021

Creative Commons License Types		
	Can someone use it commercially?	Can someone create new versions of it?
Attribution	OK (green thumbs up)	OK (green thumbs up)
Share Alike	OK (green thumbs up)	Yup, AND they must license the new work under a Share Alike license.
No Derivatives	OK (green thumbs up)	NOT OK (red thumbs down)
Non-Commercial	NOT OK (red thumbs down)	Yup, AND the new work must be non-commercial, but it can be under any non-commercial license.
Non-Commercial Share Alike	NOT OK (red thumbs down)	Yup, AND they must license the new work under a Non-Commercial Share Alike license.
Non-Commercial No Derivatives	NOT OK (red thumbs down)	NOT OK (red thumbs down)

SOURCE
<http://www.masternewmedia.org/how-to-publish-a-book-under-a-creative-commons-license/>

