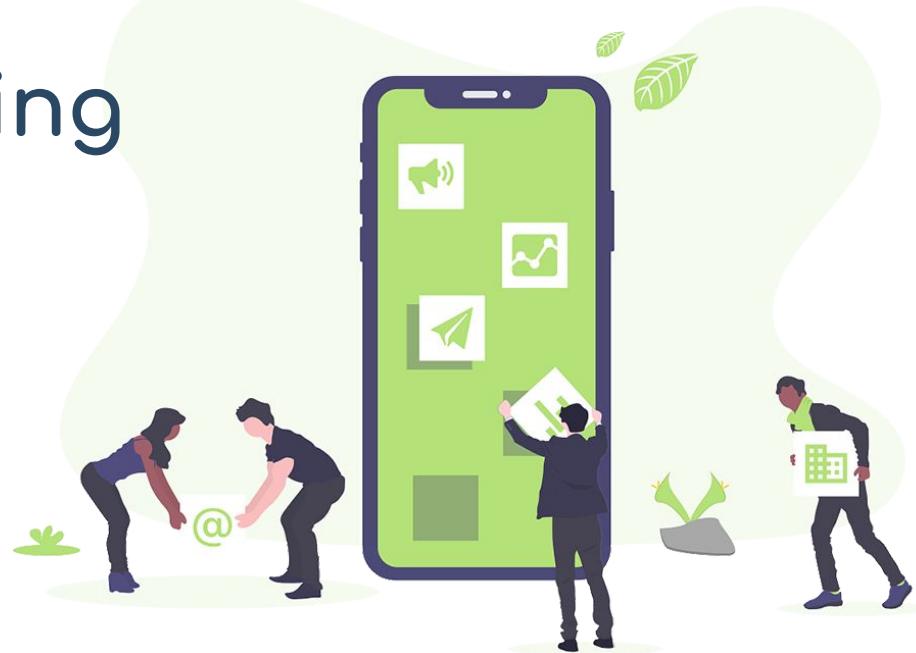




Digital Skills & Coding

Coding for Non-Coder
Beiersdorf

11.10.2021



Agenda

1. <Hello World>
2. Computational Thinking
3. Theorie & Begriffe
4. Programmiersprachen
5. Coding Robots
6. Warum Python
7. Ihr habt es geschafft!



Hello World Academy

Was machen wir?



Viele Menschen sind
durch die Digitalisierung verunsichert.

Das mangelnde Wissen führt zu Skepsis.

Digitalisierung ist mehr als

Lean Startup, Design Thinking
& Agiles Arbeiten.

Digitalisierung ist zuerst & immer:

Software. Hardware. Technik.

Wer diese Hintergründe versteht,
kann besser steuern, entscheiden und gestalten.



Um die digitale Welt zu verstehen,
hilft es, sie **zu erfahren**.

Selber
Machen 

Konzepte
Begreifen 

Digitalisierung
Verstehen

Um digitale Produkte, Prozesse und
Geschäftsmodelle zu gestalten, hilft es, die
technische Seite der Digitalisierung zu verstehen.



Eure Trainer:innen



Dr. rer. nat.
Joachim Krois

Areas of interest
Coding, Predictive
Modelling, Spatial
Data Analytics,
Artificial Intelligence
and Computer Vision



Annemieke
Frank

Areas of interest
Co-Learning
& Digital Competencies,
Coding, Design
Thinking,
Gamification



Let's stay connected

<https://etherpad.hello-world.academy/p/beiersdorf>



Agenda

1. <Hello World>
2. Computational Thinking
3. Theorie & Begriffe
4. Programmiersprachen
5. Coding Robots
6. Warum Python?
7. Ihr habt es geschafft!



“ To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability.

Jeannette M. Wing
Professor for Computer Science



“ Everybody should learn to program a computer, because it teaches you how to think.

Steve Jobs
former CEO Apple



“Programmiersprachen gehören in die Lehrpläne. Sie sind mindestens genauso wichtig wie Multiplizieren, Lesen und Fremdsprachen.”

Timothy Höttges
CEO Deutsche Telekom



Computational Thinking

Mr. Robot: The Bread Machine

1. Erteile die Befehle so, dass 3 normale Scheiben Brot abgeschnitten werden.
2. Notiere jeden Befehl im etherpad.



Computational Thinking

Mr. Robot: The Bread Machine

1. Geben Sie die Befehle so, dass 3 normale Scheiben Brot abgeschnitten werden.
2. Notieren Sie jeden Befehl im etherpad.
3. Denke wie ein Roboter: In Mural, bringe die Befehle in die richtige Reihenfolge



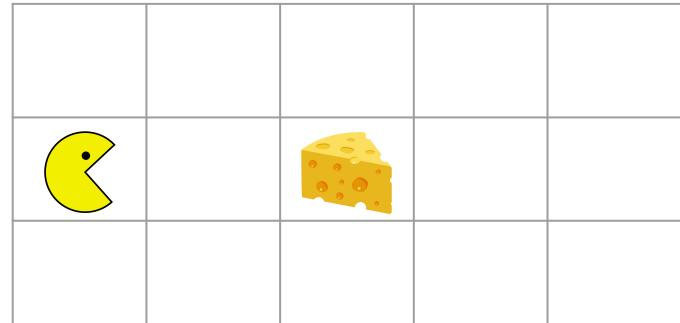
Computational Thinking

Wir schreiben unseren ersten
Algorithmus

Stelle dir vor du programmierst dein eigenes Spiel...

Das ist MacPac. MacPac versteht nur zwei Befehle:

GERADEAUS_EIN_SCHRITT und KÄSE_ESSEN



Computational Thinking

Wir schreiben unseren ersten Algorithmus

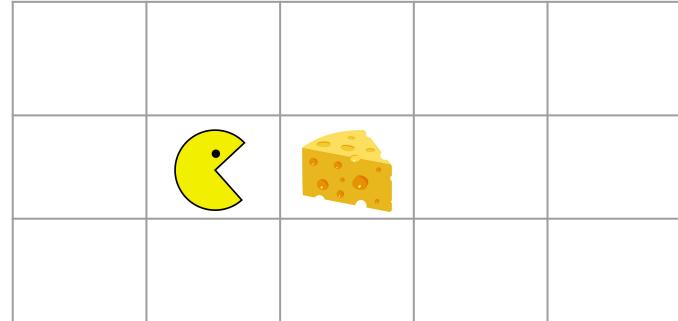
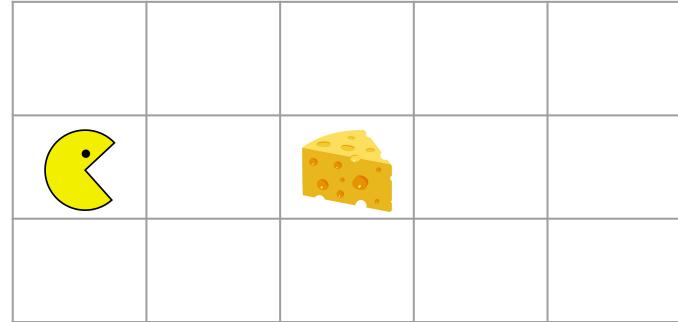
Stelle dir vor du programmierst dein eigenes Spiel...

Das ist MacPac. MacPac versteht nur zwei Befehle:

GERADEAUS_EIN_SCHRITT und KÄSE_ESSEN

Ausführung:

GERADEAUS_EIN_SCHRITT



Computational Thinking

Wir schreiben unseren ersten Algorithmus

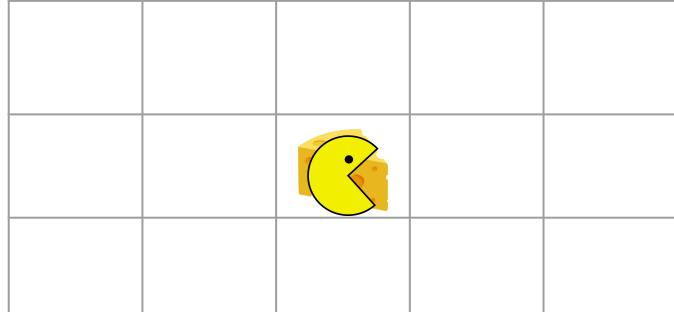
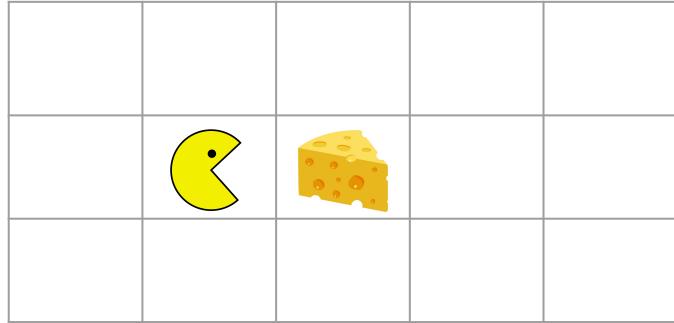
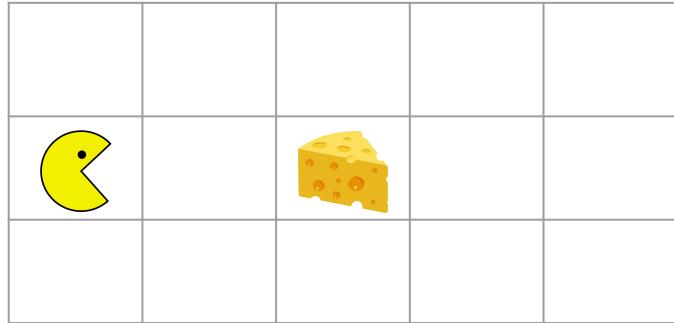
Stelle dir vor du programmierst dein eigenes Spiel...

Das ist MacPac. MacPac versteht nur zwei Befehle:

GERADEAUS_EIN_SCHRITT und KÄSE_ESSEN

Ausführung:

```
GERADEAUS_EIN_SCHRITT  
GERADEAUS_EIN_SCHRITT  
KÄSE_ESSEN
```



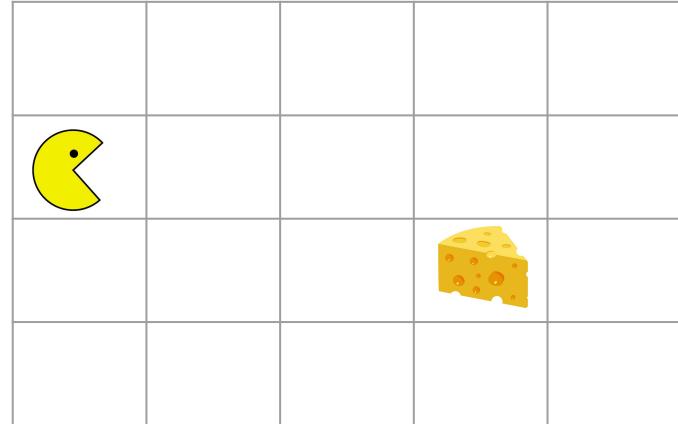
Computational Thinking

Wir schreiben unseren ersten Algorithmus

MacPac versteht nur drei Befehle:

GERADEAUS_EIN_SCHRITT
EINS_NACH_LINKS_DREHEN
KÄSE_ESSEN

Aufgabe 1:



Aufgabe: Schreibe im etherpad die Schritte auf, die MacPac vornehmen muss, um den Käse zu essen. Wichtig: Jeder Befehl bekommt seine eigene Zeile.

Computational Thinking

Wir schreiben unseren ersten Algorithmus

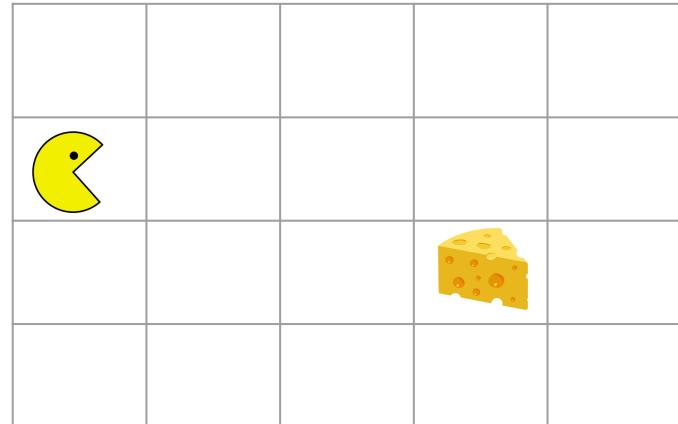
MacPac versteht nur drei Befehle:

```
GERADEAUS_EIN_SCHRITT  
EINS_NACH_LINKS_DREHEN  
KÄSE_ESSEN
```

Beispiel Ausführung:

```
GERADEAUS_EIN_SCHRITT  
GERADEAUS_EIN_SCHRITT  
GERADEAUS_EIN_SCHRITT  
EINS_NACH_LINKS_DREHEN  
EINS_NACH_LINKS_DREHEN  
EINS_NACH_LINKS_DREHEN  
GERADEAUS_EIN_SCHRITT  
KÄSE_ESSEN
```

Aufgabe 1:



Computational Thinking

Wir schreiben unseren ersten
Algorithmus

Beispiel Ausführung:

GERADEAUS_EIN_SCHRITT

GERADEAUS_EIN_SCHRITT

GERADEAUS_EIN_SCHRITT

EINS_NACH_LINKS_DREHEN

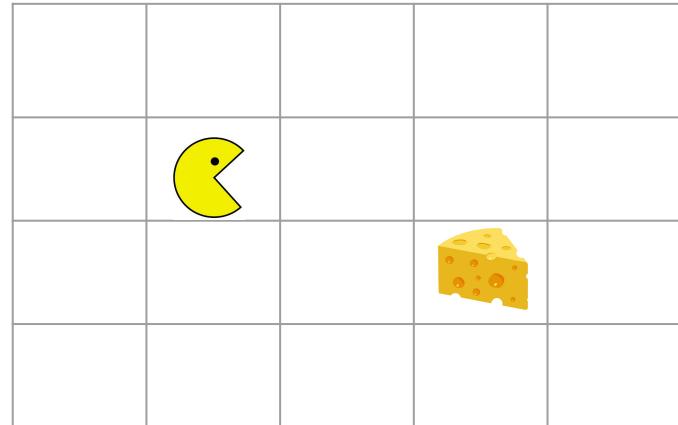
EINS_NACH_LINKS_DREHEN

EINS_NACH_LINKS_DREHEN

GERADEAUS_EIN_SCHRITT

KÄSE_ESSEN

Aufgabe 1:



Computational Thinking

Wir schreiben unseren ersten
Algorithmus

Beispiel Ausführung:

GERADEAUS_EIN_SCHRITT

GERADEAUS_EIN_SCHRITT

GERADEAUS_EIN_SCHRITT

EINS_NACH_LINKS_DREHEN

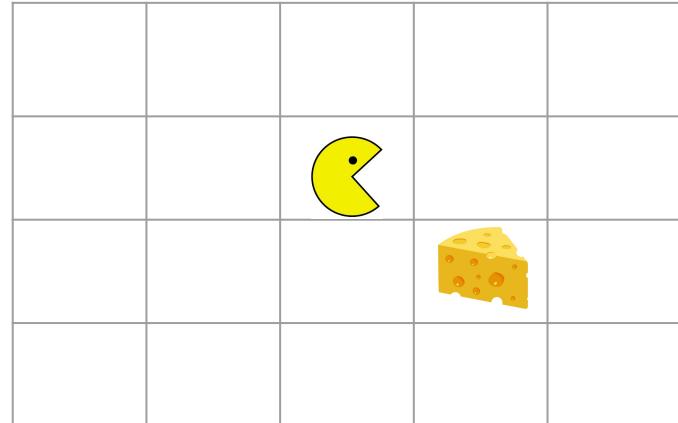
EINS_NACH_LINKS_DREHEN

EINS_NACH_LINKS_DREHEN

GERADEAUS_EIN_SCHRITT

KÄSE_ESSEN

Aufgabe 1:



Computational Thinking

Wir schreiben unseren ersten
Algorithmus

Beispiel Ausführung:

GERADEAUS_EIN_SCHRITT

GERADEAUS_EIN_SCHRITT

GERADEAUS_EIN_SCHRITT

EINS_NACH_LINKS_DREHEN

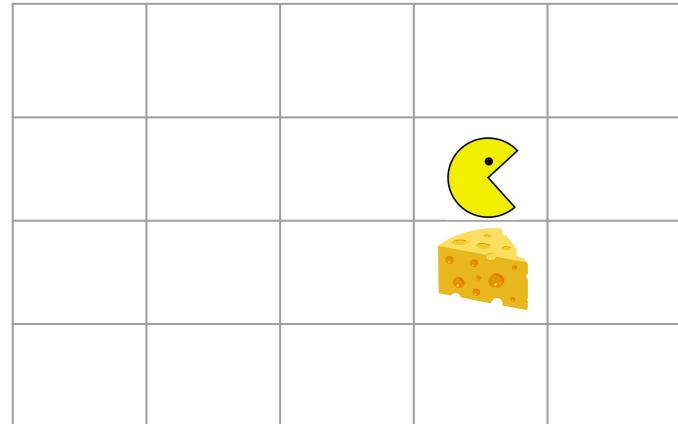
EINS_NACH_LINKS_DREHEN

EINS_NACH_LINKS_DREHEN

GERADEAUS_EIN_SCHRITT

KÄSE_ESSEN

Aufgabe 1:



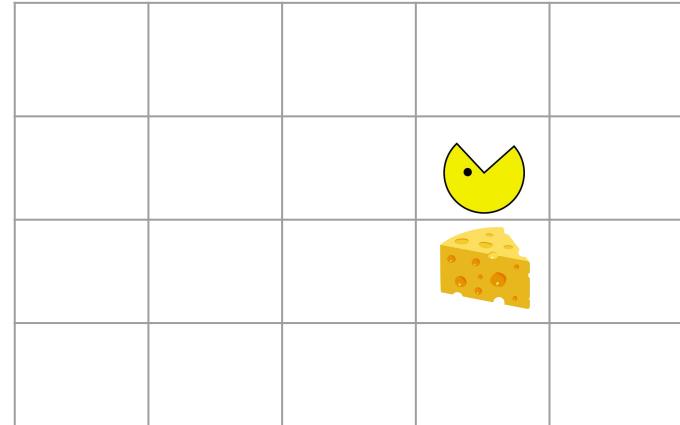
Computational Thinking

Wir schreiben unseren ersten
Algorithmus

Beispiel Ausführung:

```
GERADEAUS_EIN_SCHRITT  
GERADEAUS_EIN_SCHRITT  
GERADEAUS_EIN_SCHRITT  
EINS_NACH_LINKS_DREHEN  
EINS_NACH_LINKS_DREHEN  
EINS_NACH_LINKS_DREHEN  
GERADEAUS_EIN_SCHRITT  
KÄSE_ESSEN
```

Aufgabe 1:



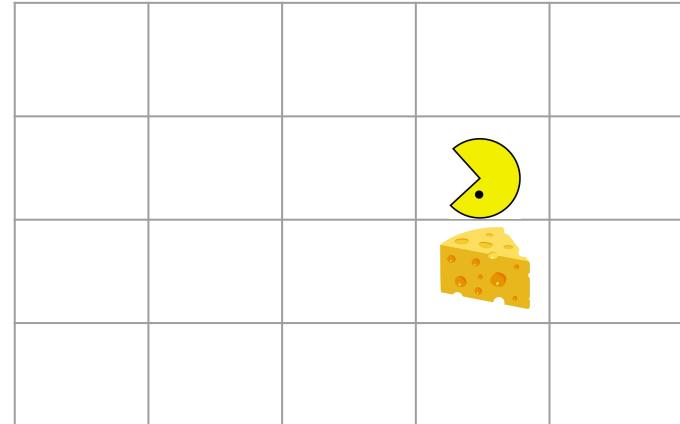
Computational Thinking

Wir schreiben unseren ersten
Algorithmus

Beispiel Ausführung:

```
GERADEAUS_EIN_SCHRITT  
GERADEAUS_EIN_SCHRITT  
GERADEAUS_EIN_SCHRITT  
EINS_NACH_LINKS_DREHEN  
EINS_NACH_LINKS_DREHEN  
EINS_NACH_LINKS_DREHEN  
GERADEAUS_EIN_SCHRITT  
KÄSE_ESSEN
```

Aufgabe 1:



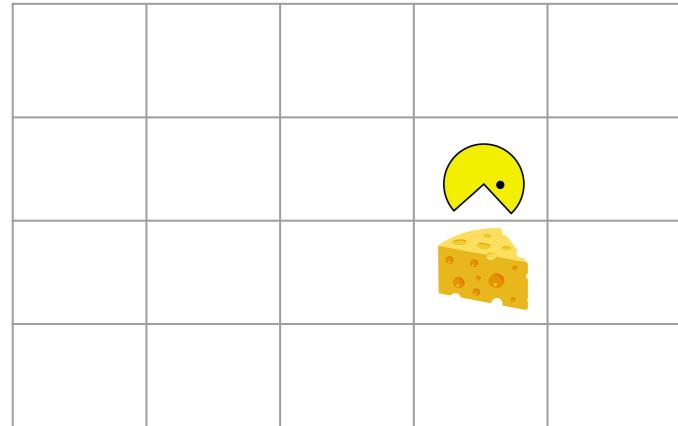
Computational Thinking

Wir schreiben unseren ersten
Algorithmus

Beispiel Ausführung:

```
GERADEAUS_EIN_SCHRITT
GERADEAUS_EIN_SCHRITT
GERADEAUS_EIN_SCHRITT
EINS_NACH_LINKS_DREHEN
EINS_NACH_LINKS_DREHEN
EINS_NACH_LINKS_DREHEN
GERADEAUS_EIN_SCHRITT
KÄSE_ESSEN
```

Aufgabe 1:



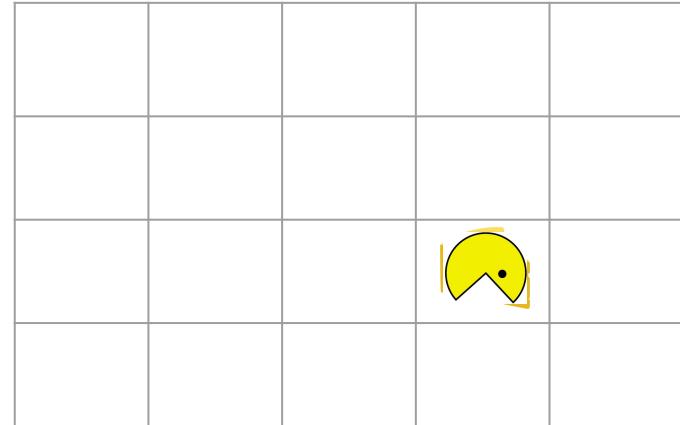
Computational Thinking

Wir schreiben unseren ersten
Algorithmus

Beispiel Ausführung:

```
GERADEAUS_EIN_SCHRITT  
GERADEAUS_EIN_SCHRITT  
GERADEAUS_EIN_SCHRITT  
EINS_NACH_LINKS_DREHEN  
EINS_NACH_LINKS_DREHEN  
EINS_NACH_LINKS_DREHEN  
GERADEAUS_EIN_SCHRITT  
KÄSE_ESSEN
```

Aufgabe 1:



Computational Thinking

10 Minuten

Wir schreiben unseren ersten Algorithmus

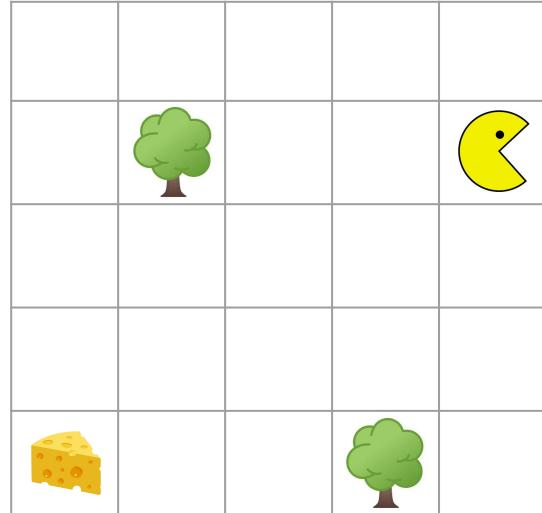
MacPac versteht nur drei Befehle:

```
GERADEAUS_EIN_SCHRITT  
EINS_NACH_LINKS_DREHEN  
KÄSE_ESSEN
```

Jetzt kannst du auch deine eigenen Befehle definieren, indem du die alten Befehle kombinierst. Zum Beispiel:

```
DREHUNG_180 =  
EINS_NACH_LINKS_DREHEN + EINS_NACH_LINKS_DREHEN
```

Aufgabe 2:



Aufgabe: Teamarbeit, welches Team schreibt den kürzesten Algorithmus?

Agenda

1. <Hello World>
2. Computational Thinking
3. Theorie & Begriffe
4. Programmiersprachen
5. Coding Robots
6. Warum Python
7. Ihr habt es geschafft!



Theorie & Begriffe

Wir spielen ein Spiel

Wie gut kennt ihr euch mit IT-Begriffen aus?

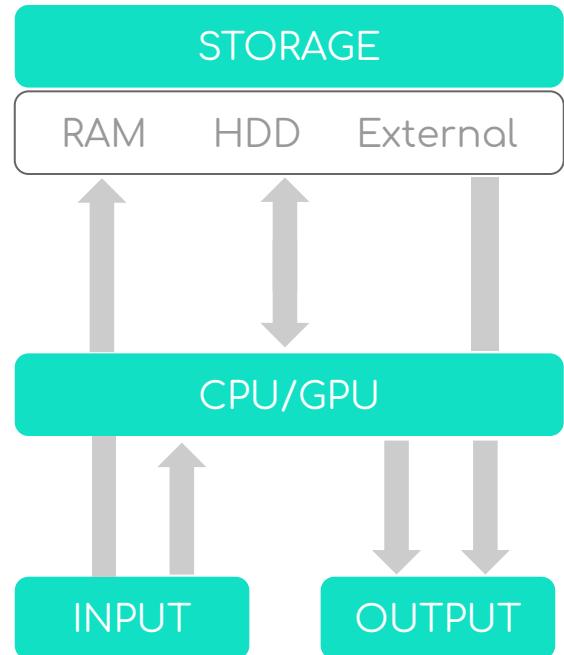
Aufgabe:

- In Teams trefft ihr euch auf Mural..
- Wir spielen 3 Runden. Jedes Team erhält jede Runde 5 Definitionen.
- In der Mitte vom Mural Board stehen immer die dazugehörigen Begriffe.
- Ihr müsst jetzt die Definitionen mit den Begriffen in der Mitte matchen.
- Welches Team am schnellsten ist gewinnt :-)



HARDWARE

- Hardware sind die physischen Bestandteile eines Computer
- In der CPU (Central Processing Unit, Prozessor) wird Software ausgeführt
- Im Storage (Speicher) werden Daten gesichert
- Input: Maus, Tastatur, Touchscreen ...
- Output: Monitor, Ton, Vibration ...
Analogie: Hardware als Körper



SOFTWARE

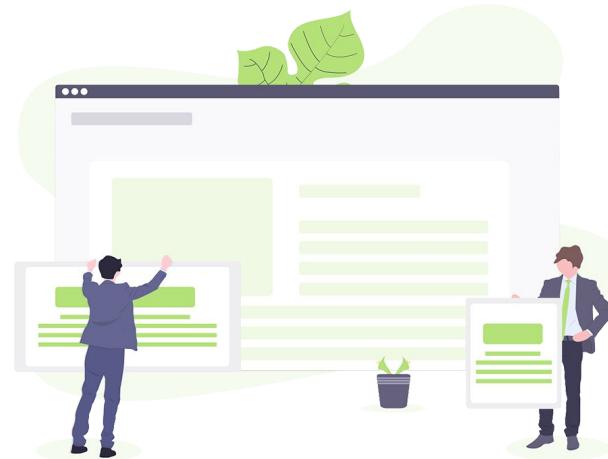
- **Software** bestimmt, welche Befehle Hardware ausführt
- **Daten** sind Informationen, die im Storage gespeichert werden
- **Code** ist die Gesamtheit von textbasierten Befehlen



FUNKTION

- Eine **Funktion** sind Wiederverwendbare Sammlung von Befehlen

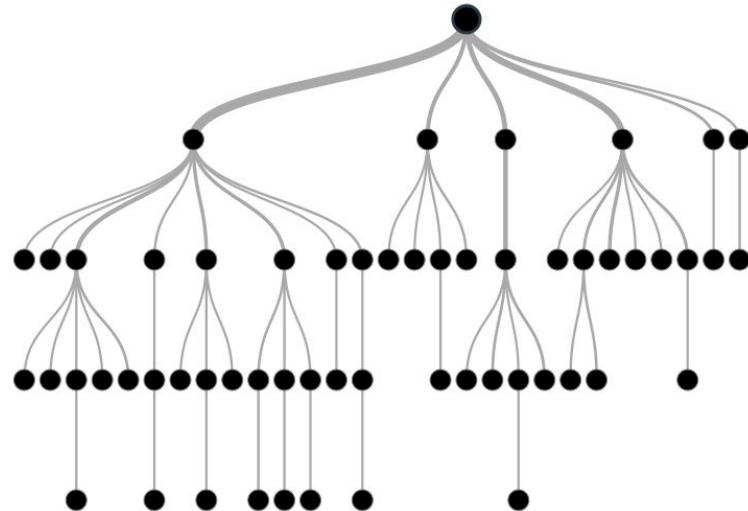
```
function roll (time, speed, dir) {  
    setDirection(dir);  
    setSpeed(speed);  
}  
  
function order(type, client) {  
    var pizza = bake(type);  
    deliver(pizza, client);  
    ...  
}
```



ABFRAGE

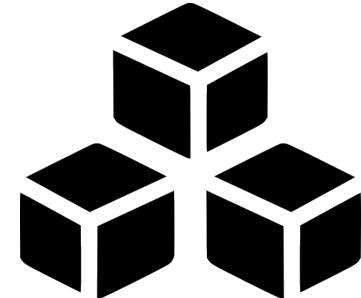
- Von **Abfrage** spricht man, wenn Befehle ausgeführt werden, wenn eine Voraussetzung erfüllt ist.

```
if (colour == green) {  
    Sound(Boing);  
}  
if (Waiter.ready == Yes) {  
    order();  
}
```



BIBLIOTHEK

- Sammlung von Funktionen und Befehlen



FRAMEWORK

- Sammlung von Funktionen, Befehlen und Regeln



SOFTWARE PHASEN

- In der **Entwicklung** wird die Software programmiert (Feld der Programmierer)
- In der **Laufzeit** wird die Software ausgeführt (Feld der Anwender)
- Analogie: Auto bei der Herstellung & Reparatur in der Entwicklung und bei der Fahrt in der Laufzeit



FRONTEND & BACKEND

- Frontend und Backend beschreiben Schichten von IT-Systemen
- Frontend ist näher am Anwender und an Dateneingaben, kann Logik enthalten
- Backend ist näher an der Datenverarbeitung
- Begriffspaar ist kontextbezogen
- Analogie: Mimik als Frontend, Gedanken als Backend



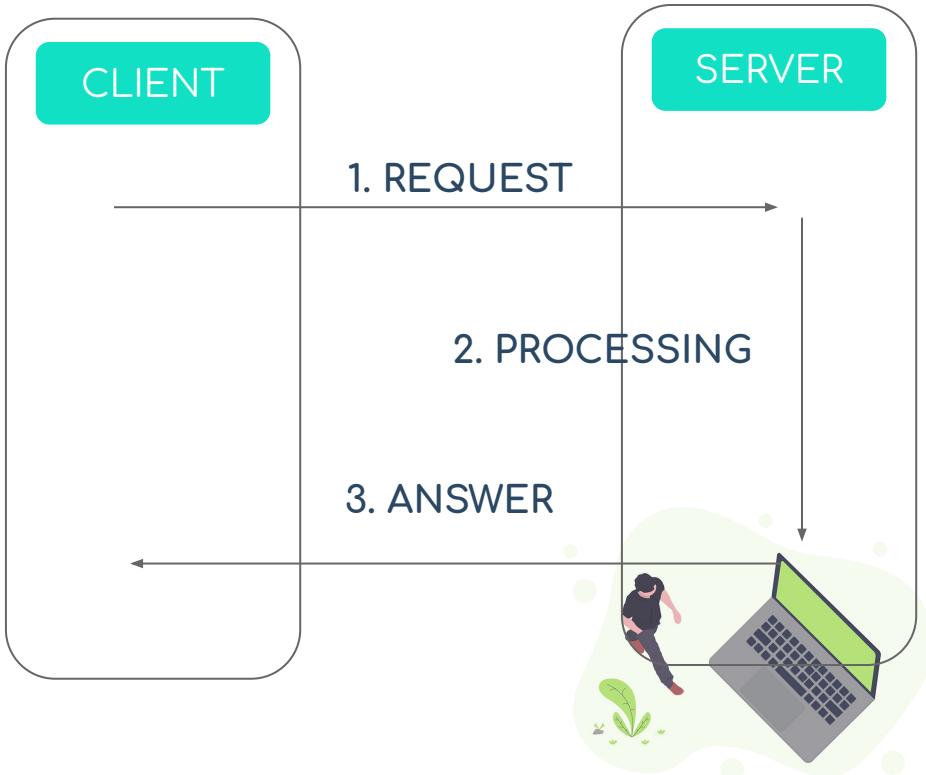
INTERFACE

- Interfaces sind Schnittstellen über die ein System seiner Umwelt definierte Funktionen bereit stellt.
- Graphische Interfaces (auch GUI, Graphic User Interface) werden von Menschen genutzt.
- API (Application Programming Interfaces) werden von anderen Programmen genutzt.
- Analogie: Stromkabel in Steckdose, Herdplatte und Topf und Deckel



CLIENT & SERVER

- Client-Server beschreibt die Aufgabenverteilung zweier Software-Systeme zueinander.
- Ein physisches Gerät kann gleichzeitig Client und Server sein.
- Das Begriffspaar ist kontextbezogen.
- Analogie: Pizzakunde ist Client, Mitarbeiter ist Server.



Agenda

1. <Hello World>
2. Computational Thinking
3. Theorie & Begriffe
4. Programmiersprachen
5. Coding Robots
6. Warum Python
7. Ihr habt es geschafft!



Programmiersprachen



Ada Lovelace
(1815-1852)

Entwickelte gemeinsam mit Babbage die Vorstufe zur ersten Programmiersprache



Grace Hopper
(1906-1992)

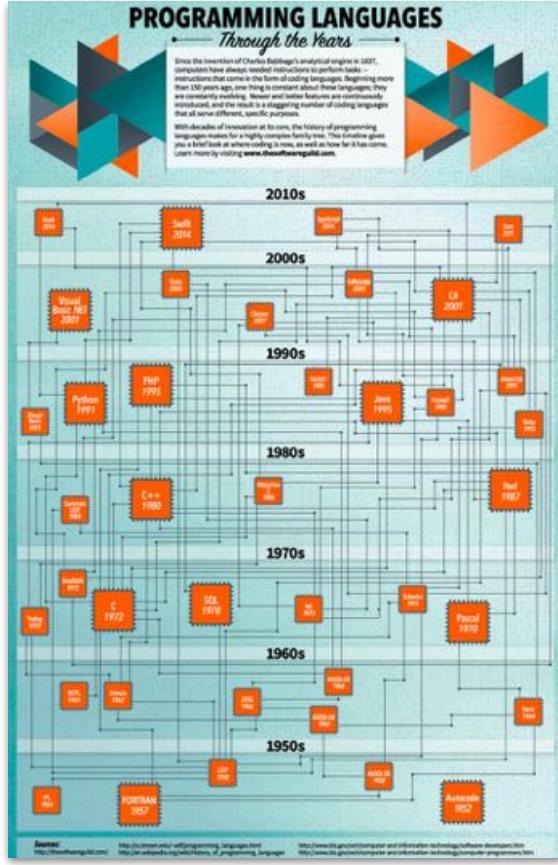
An der Entwicklung des ersten Compilers beteiligt. Dieser übersetzte Quellcode in eine für den Prozessor verständliche Maschinensprache.



Margret Hamilton
(1936)

Leitete das Software-Team der Apollo-11-Mission. Und ganz nebenbei etabliert sie Prinzipien des Programmierens, die bis heute gelten.

Coding Sprachen



Mittagspause

Lasst es euch schmecken



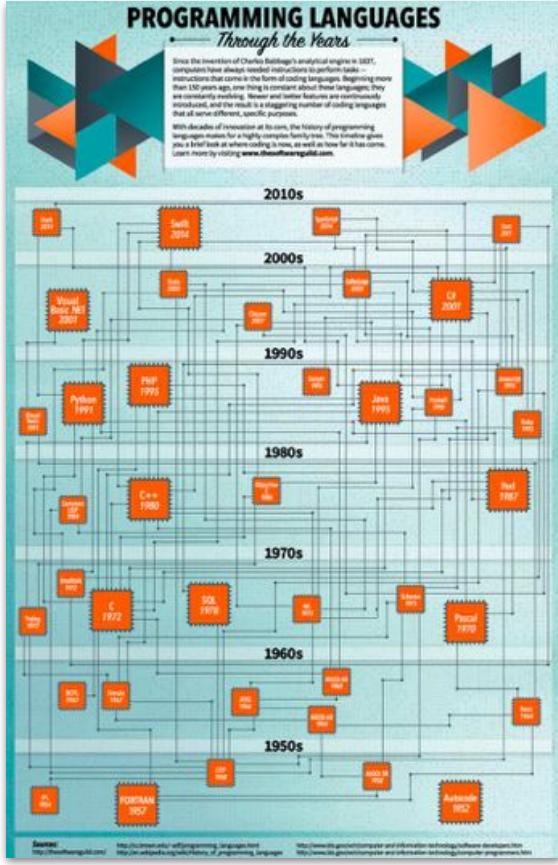
Quiz

Habt ihr auch schön aufgepasst ;-)

- Gehe auf [www.kahoot.it](http://wwwkahootit)
- Und gebe folgenden PIN ein. Schnelligkeit gewinnt!

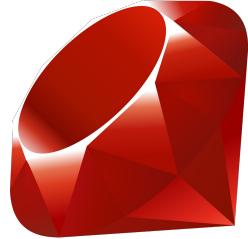


Programmiersprachen



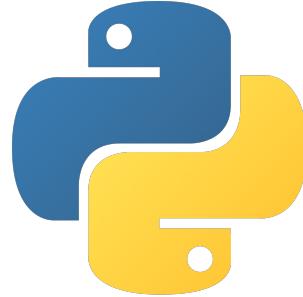
RUBY

```
MyLittleVar = "Hello World!"  
5.times{puts MyLittleVar}
```



PYTHON

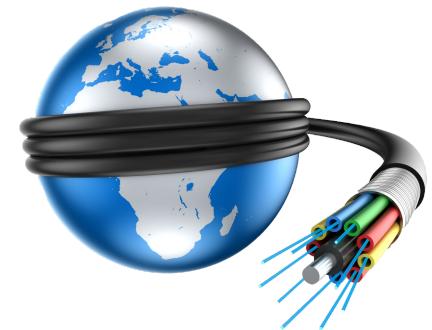
```
MyLittleVar = "Hello World!"  
for _ in range(5):  
    print(MyLittleVar)
```



NETFLIX

JAVASCRIPT

```
var MyLittleVar = "Hello World!";  
  
for (var x = 1; x <= 5; x++)  
  
{  
  
    alert(MyLittleVar);  
  
};
```



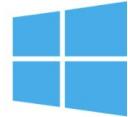
JAVA

```
class MyExample {  
    public static void main(String[] args)  
    {  
        String MyLittleVar = "Hello World!";  
        for (int x = 0; x < 5; x++)  
        {  
            System.out.println(MyLittleVar);  
        }  
    }  
}
```



C

```
#include<stdio.h>
int main() {
    char MyLittleVar[] = "Hello
World\n";
    int x = 0;
    for (x = 0; x < 5; x++)
    {
        printf("%s", MyLittleVar);
    }
    return 0;
}
```



ASSEMBLER

```
section .data
    MyLittleVar  db 'Hello World!', 10
    length equ $ - MyLittleVar;

section .data
    start:
        mov cx, 5 ; fill cx-register with 5
    loop:
        mov eax, 4 ; write(stdout, hello, length)
        mov ebx, 1
        mov ecx, MyLittleVar
        mov edx, length
        int 80h

        loop schleife ; jump to 'loop' as long as cx > 0 and
decrease cx by 1
        mov ebx, 0 ; Call: exit
        mov eax, 1
        int 80h
```



MACHINE CODE

```
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010  
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010  
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010  
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010  
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010  
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010  
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010  
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010  
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010  
01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100 0001010
```

=

Hello World

Hello World

Hello World

Hello World

Hello World



Coding & Languages

Jetzt seid ihr gefragt



Python Online Compiler

```
main.py
1 # Online Python compiler (interpreter) to run Python online.
2 # Write Python 3 code in this online editor and run it.
3 print("Hello world")
```

- Clicke auf folgenden Link:
<https://www.programiz.com/python-programming/online-compiler/>
- Programmiere selbst!
- Code snippets
<https://gist.github.com/eotp/89d1c14a854b26a19f11ad8a5c3403f8>

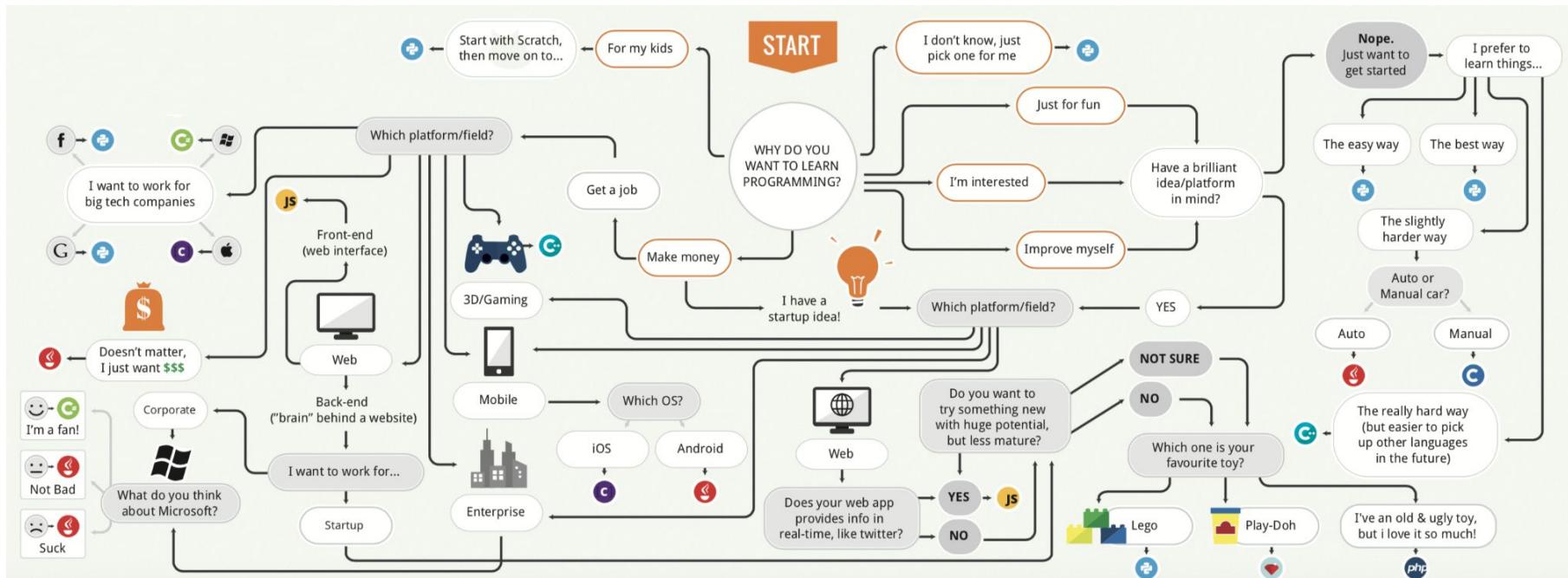


Coding for Non-Coders



Programmiersprachen

Mit welcher Sprache fange ich an?



JS JAVASCRIPT

C# C#

RUBY

OBJ OBJECTIVE-C

PYTHON

CAFFEE JAVA

C

PHP

C++

Coding for Non-Coders

Agenda

1. <Hello World>
2. Computational Thinking
3. Theorie & Begriffe
4. Programmiersprachen
5. Coding Robots
6. Warum Python
7. Ihr habt es geschafft!



Coding Robots

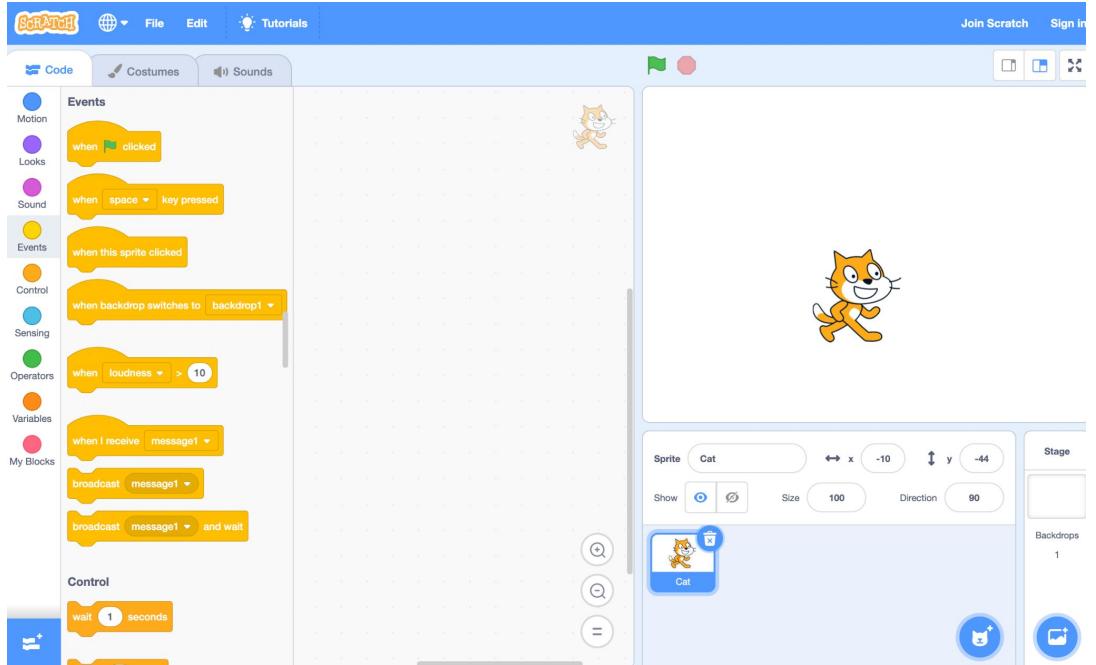
Scratched-based Coding



- Scratch ist eine blockbasierte visuelle Programmiersprache
- Ursprünglich vom MIT Media Lab entwickelt.
- Die offizielle Website verzeichnet mehr als 73 Millionen gemeinsame Projekte von über 68 Millionen Nutzern und fast 38 Millionen monatliche Website-Besuche
- Scratch ist quell offen und wurde in Hunderten von verschiedenen Editoren angepasst.
- Die Scratch-Oberfläche ist in drei Hauptbereiche unterteilt: einen Bühnenbereich, eine Blockpalette und einen Codierbereich

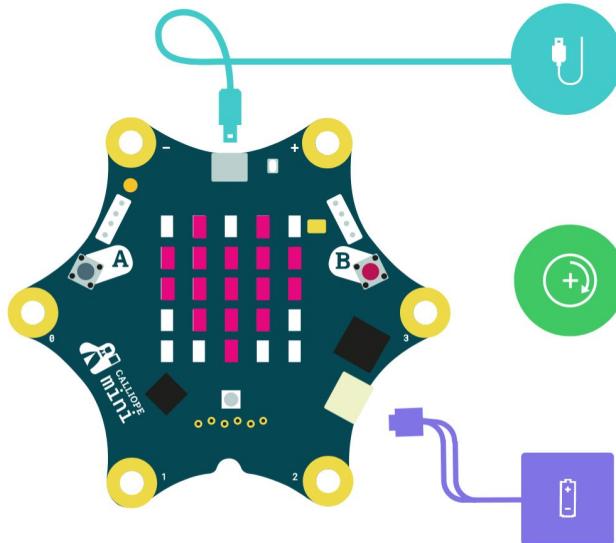
Coding Robots

Scratched-based Coding



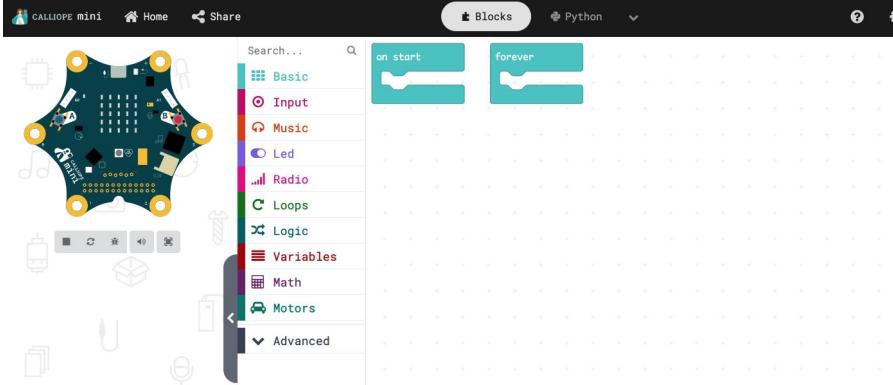
Coding Robots

Einführung: Calliope Mini



Coding Robots

Einführung: Calliope Mini



Coding Robots

Einführung: Calliope Mini

Klicke auf die Schaltfläche Download, um das Programm herunterzuladen.

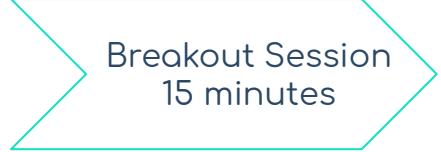
Die Hex-Datei wird zunächst in deinem Download-Ordner gespeichert.

Ziehe dann die Hex-Datei auf das Laufwerk mit dem Namen MINI.



Coding Robots

Einführung: Calliope Mini



Breakout Session
15 minutes

Los geht's

Schritt 1: Schreibe deinen Namen.

Schritt 2: Wiederhole deinen Namen zweimal.

Schritt 3: Lass deinen Namen nur erscheinen, wenn du den Calliope schüttelst.

Schritt 4: Lass ein Herz erscheinen, wenn du Knopf A drückst.

Schritt 5: Schalte die blaue LED ein, wenn du den Knopf B drückst.

Schritt 6: Füge einen Titelsong hinzu



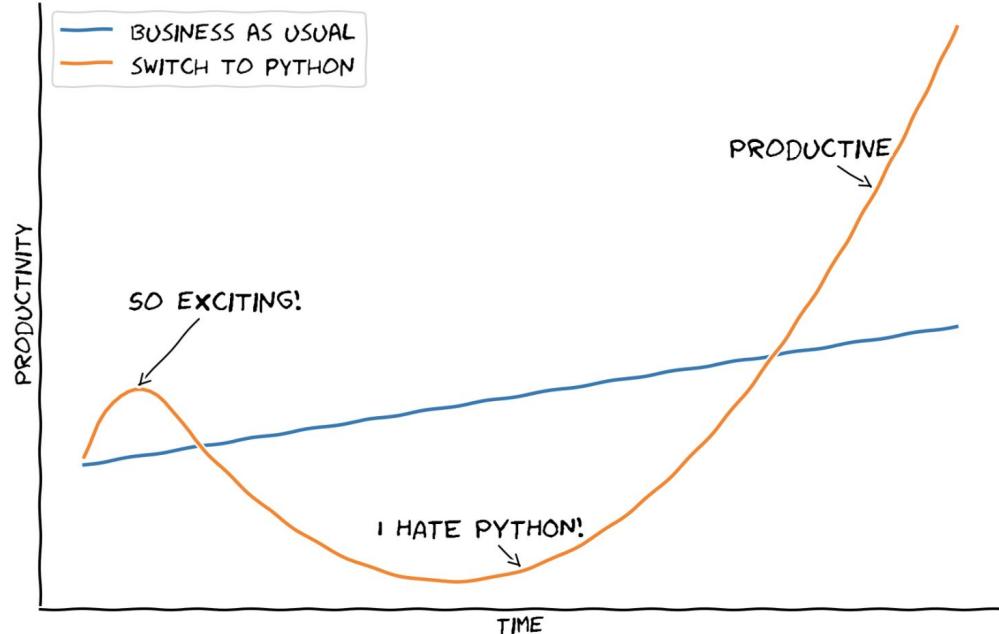
Agenda

1. <Hello World>
2. Computational Thinking
3. Theorie & Begriffe
4. Programmiersprachen
5. Coding Robots
6. Warum Python
7. Ihr habt es geschafft!



Warum Python?

Eine leistungsstarke Allzwecksprache mit einer hervorragenden Syntax ("ausführbarer Pseudocode")



Warum Python?

Interoperabilität mit anderen Sprachen¶

[...] Ich wollte nie, dass Python die Hauptsprache für Programmierer wird.

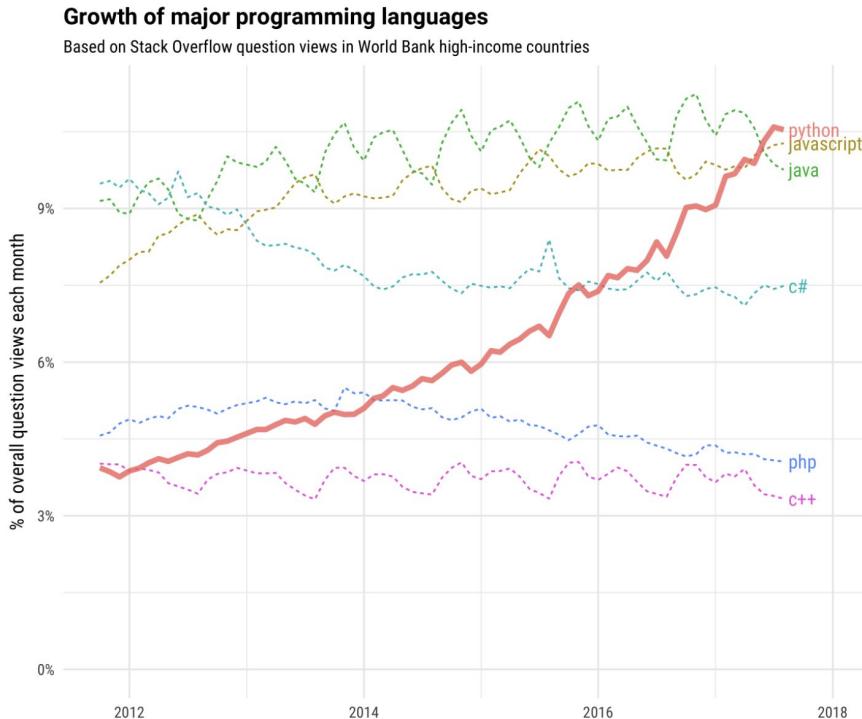
[...] die Lücke zwischen der Shell und C zu schließen [...]

Guido van Rossum
Ehemaliger "Wohlwollender Diktator auf Lebenszeit (BDFL)".



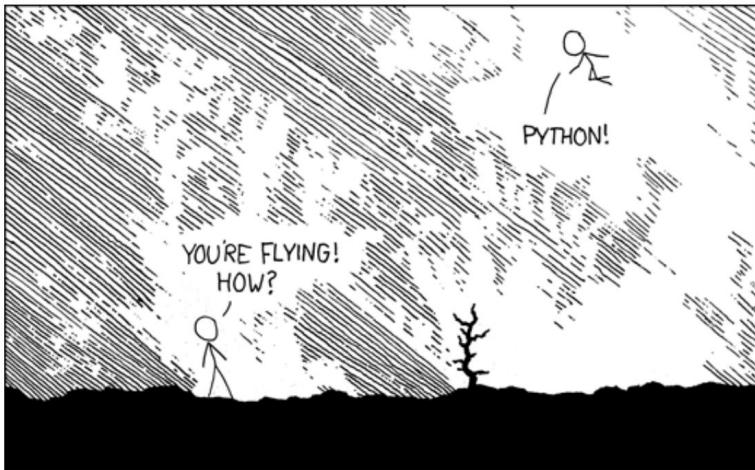
Warum Python?

Offene und ermutigende Gemeinschaft



Warum Python?

Inklusive Batterien und Module von Drittanbietern



I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
/ HELLO WORLD IS JUST
print "Hello, world!"

I DUNNO...
DYNAMIC TYPING?
WHITESPACE?
/ COME JOIN US!
PROGRAMMING IS FUN AGAIN!
IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?

I JUST TYPED
import antigravity
THAT'S IT?
/ ... I ALSO SAMPLED
EVERYTHING IN THE
MEDICINE CABINET FOR COMPARISON.
/ BUT I THINK THIS
IS THE PYTHON.

Warum Python?

Inklusive Batterien und Module von Drittanbietern



Wie spreche ich Python?

Text editors

Ein Texteditor ist eine Art Computerprogramm, das einfachen Text bearbeitet. Sie werden verwendet, um Dateien wie Konfigurationsdateien, Dokumentationsdateien und den Quellcode von Programmiersprachen zu ändern.

Plain text

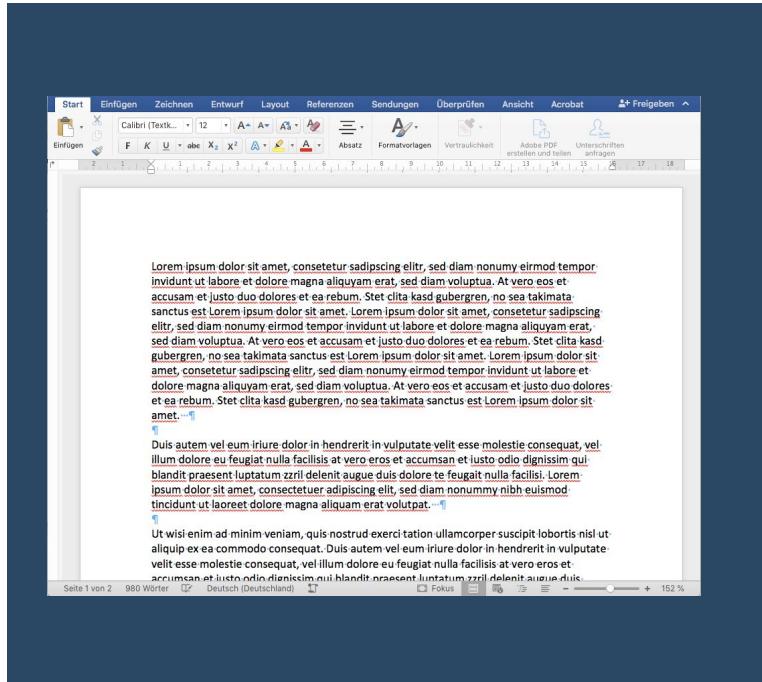
Plain text exclusively consists of character representation. Plain text contains no other information about the text itself, not even the character encoding convention employed.

Rich text

Rich text may contain metadata, such as character and paragraph formatting data as well as page specification data. Rich text can be very complex. Rich text can be saved in binary format (e.g. .doc), text files adhering to a markup language (e.g. Markdown or HTML), or in a hybrid form of both (e.g. Office Open XML).

Wie spreche ich Python?

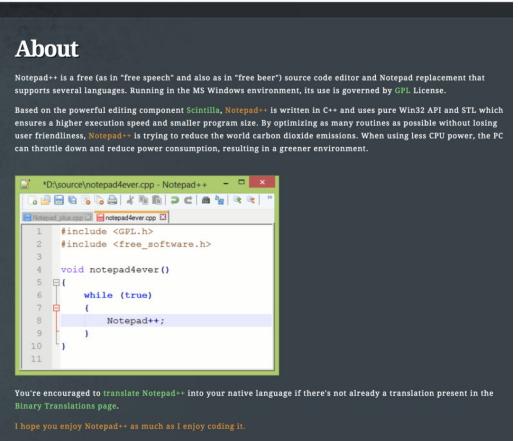
Text editors



```
1 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor  
invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et  
accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata  
sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing  
elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,  
sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd  
gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit  
amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore  
et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores  
et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit  
amet....  
2  
3 Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel  
illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui  
blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem  
ipsum dolor sit amet, consetetur adipiscing elit, sed diam nonumy nibh euismod.  
tincidunt ut laoreet dolore magna aliquam erat volutpat....  
4  
Ut wisi enim ad minim veniam, quis nostrud exercitation ullamcorper suscipit lobortis nisl ut  
aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate  
velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et  
accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore  
te feugait nulla facilisi. Lorem ipsum dolor sit amet, consetetur
```

Python

Text editors



The screenshot shows the Atom text editor's website. At the top, there are navigation links: 'Packages', 'Themes', 'Documentation', 'Blog', and 'Discuss'. On the far right, there are 'Sign In' and a 'Sign Up' button. The main visual is a large circular graphic featuring a stylized atomic model in the center, surrounded by concentric rings of orange and yellow. To the left of this graphic is a cartoon illustration of a green, multi-eyed alien with a small antenna. To the right, there's a box with the word 'ATOM' in large letters, followed by '1.57.0' and 'Release notes'. Below that is another box labeled 'macOS' with the note 'For macOS 10.10 or later'. At the bottom right is a prominent yellow 'Download' button with a downward arrow icon. At the very bottom of the page, there's a footer with the text 'By downloading, you agree to the Terms and Conditions.' and links for 'Other platforms', 'Try Atom Beta', and 'Try Atom Nightly'.

The screenshot shows the Sublime Text interface. The left pane displays the file structure of a GitHub repository, including files like README.md, CONTRIBUTING.md, LICENSE, and package-lock.json. The right pane shows the content of CONTRIBUTING.md, which discusses Svelte as a new way to build web applications and provides links for contributing, creating issues, and joining the community. A status bar at the bottom indicates "F11" and "File / Open".

Sublime Text

Dark | Light

DOWNLOAD FOR MAC Sublime Text 4 (Build 4107) See What's New

Linux Mac Windows

Sublime Text

README.md CONTRIBUTING.md

1 <p> Svelte is a component-based web application framework built on top of the Vite build tool. It's a compiler that takes your declarative component descriptions and turns them into JavaScript that surgically updates the DOM.

2 Svelte is a new way to build web applications. It's a compiler that takes your declarative component descriptions and turns them into JavaScript that surgically updates the DOM.

3 The Open Source Guides (<https://openguides.org>) are maintained by individuals, communities, and companies. These resources are open to anyone to contribute to, read, run and contribute to open source projects. Consider contributing to one of these guides; it will be very useful!

4 How to Contribute to Open Source! (<https://openguides.org/contributing.html>)

5 Building Welcoming Communities (<https://openguides.org/building-community.html>)

6 Get Involved

7 There are many ways to contribute to Svelte, and most of them don't require writing any code. Here's a few ideas to get started:

8 Start using Svelte. Go through the [tutorial](https://svelte.dev/tutorial), [the guide](https://svelte.dev/guide), [the API reference](https://svelte.dev/api), and everything else. Report bugs, always looking for improvements. Let us know by opening an issue or pull request.

9 Look through the open issues (<https://github.com/sveltejs/svelte/issues>). Provide workarounds, ask for clarification, or suggest improvements.

10 F11

Wie spreche ich Python?

Integrated Development Environments (IDE)



Coding for Non-Coders



NetBeans



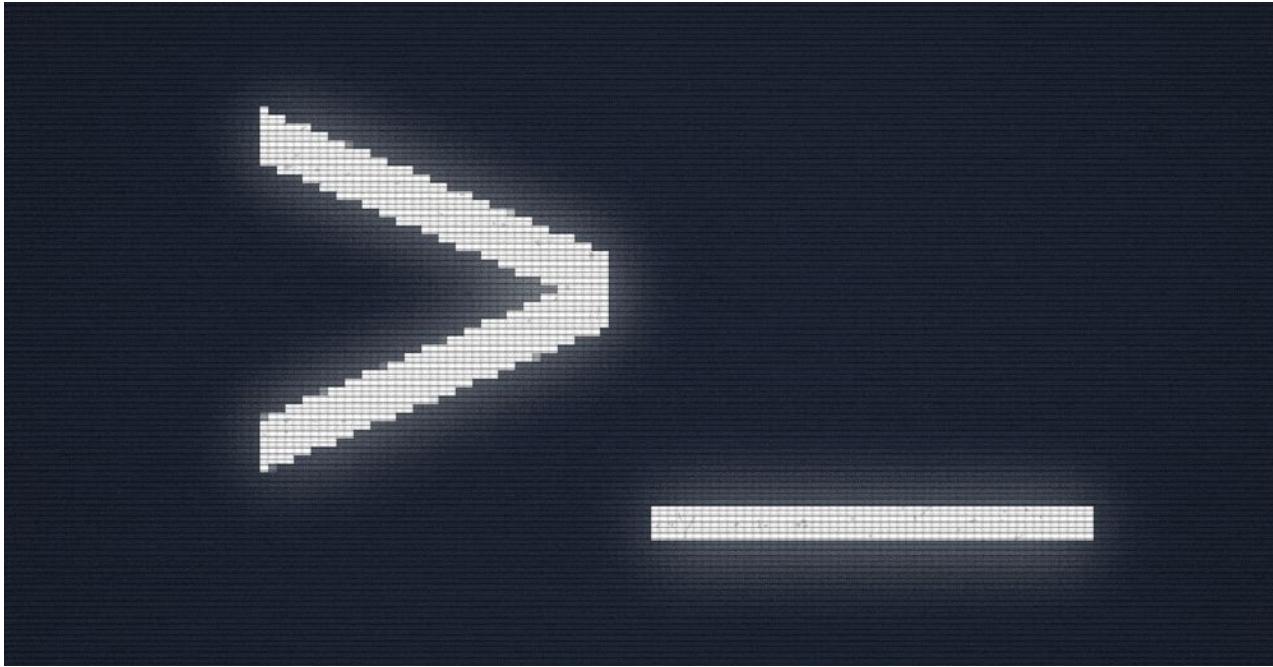
Wohlverdiente Kaffeepause

Wir sehen uns in 15 Minuten wieder.



Hello world!

Wir lernen unsere command line kennen.



Hello world!

Der Python interpreter

```
bash
bash-3.2$
```

```
bash
bash-3.2$ bash-3.2$ python
Python 3.8.5 | packaged by conda-forge | (default, Jul 31 2020, 02:18:36)
[Clang 10.0.1 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Hello world!

Script mode

```
bash  
bash-3.2$
```

```
bash  
bash-3.2$ python chatbot.py
```

My chatbot

```
1  import time
2  print("Hi, my name is Vincent.")
3  time.sleep(1)
4  print("What's your name?")
5  user = input()
6  if user.lower() == "joachim":
7      print("Good morning my creator!")
8      print('')
9  elif user.lower() == 'mieke':
10     print("Oh, what a pleasure! It feels great to have the Hello World Academy in the house.")
11     time.sleep(1)
12     print("Is Joachim somewhere around too?")
13     answer = input()
14     if answer.lower() == 'yes':
15         print('Great, a strong team!')
16     elif answer.lower() == 'no':
17         print('OK, if you see him make sure he knows that we miss him.')
18     else:
19         print("Well, did not get that, but that's fine. See you buddy!")
20     print('')
21 else:
22     print(f"Nice to meet you {user}.")
23     print('')
24 time.sleep(1)
```



Jupyter Lab



<https://mybinder.org/v2/gh/hello-world-academy/beiersdorf-remote/day1?urlpath=lab/>

ABSCHLUSS

- Haben wir eure Erwartungen getroffen?
- Was fandt ihr besonder gut?
- Was können wir besser machen?





Annemieke Frank
Dr. Joachim Krois

www.hello-world.academy
hello@hello-world.academy