CEDUS-WORKSHOP

TAG 1: EINFÜHRUNG IN SCRATCH & PYTHON

HEUTIGE AGENDA

- Einführung in Scratch - euer eigenes Spiel
- Algorithmen & Algorithmisches Denken
- Python
  1. Basics: Variablen, Funktionen, Kontrollstrukturen
  2. Advanced: Listen, Dictionaries, Bibliotheken

Name: Lea

Alter: 26 Jahre

Beruf: Maschinenbau-Studentin

Hobbies: Tanzen, Hochseilgärten

Name: Philipp

Alter: 25 Jahre

Beruf: Informatik-Student

Hobbies: Joggen,
Fußball,
Computer/Brettspiele,
Kochen

EINFÜHRUNG IN SCRATCH

WAS IST SCRATCH?

- Programmier-Lernplattform für Kinder ab 10 Jahren
- Spiele entwickeln, Animationen erstellen, Stories erzählen
- Programmieren durch Zusammenstecken von Anweisungs-Puzzleblöcken

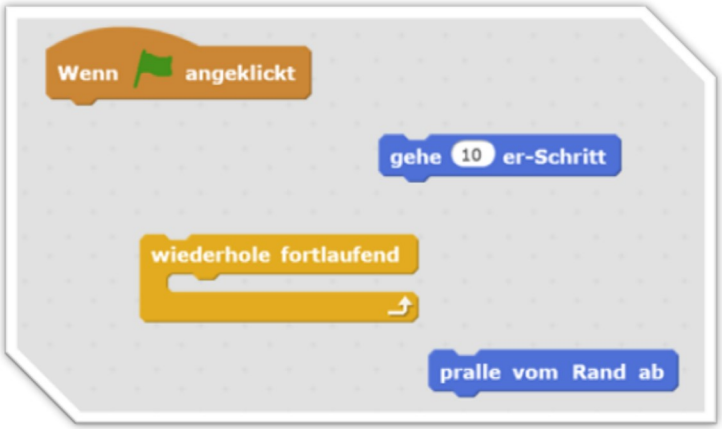
Offizielle Seite

SOFTWARE WALKTHROUGH + DEMO DES SPIELS

TEILAUFGABE 1: SPIELFIGUR

- Spielfigur soll sich vorwärts bewegen

- Trifft sie auf eine Wand, soll sie von dieser abprallen



TEILAUFGABE 2: STEUERUNG

- Die Spielfigur soll mithilfe einer Taste (z.B. A) steuerbar sein

- Wird die Taste gedrückt, dreht sich die Figur (z.B. 15°)

TEILAUFGABE 3: PUNKT SAMMELN

- Zusätzliches Objekt als "Punkt" einfügen

- Beim Einsammeln erhält Punkt neue Position

TEILAUFGABE 4: EIN ZWEITER SPIELER

- Fügt einen zweiten Spieler ein, der über eine andere Taste steuerbar ist

- Tipp: "Copy&Paste"

TEILAUFGABE 5: PUNKTE ZÄHLEN

- Fügt 2 Variablen ein, die die Punkte der beiden Spieler zählen

TEILAUFGABE 6: "GAME OVER"-SCREEN

- Hat ein Spieler X Punkte, soll das Spiel beendet werden

- Game-Over-Screen + Spiel soll anhalten

ALGORITHMISCHES DENKEN

WAS IST EIGENTLICH EIN ALGORITHMUS?

```
def euclid(a,b):
 if b = 0:
     return b
 else:
     return euclid(b,a%b)
```

Ein Algorithmus ist eine eindeutige Handlungsvorschrift zur Lösung eines Problems oder eine Klasse von Problemen. Algorithmen bestehen aus endlich vielen, wohldefinierten Einzelschritten. Damit können sie zur Ausführung in ein Computerprogramm implementiert, aber auch in menschlicher Sprache formuliert werden.

http://www.wikipedia.org/Algorithmus

ALGORITHMISCH DENKEN HEISST..

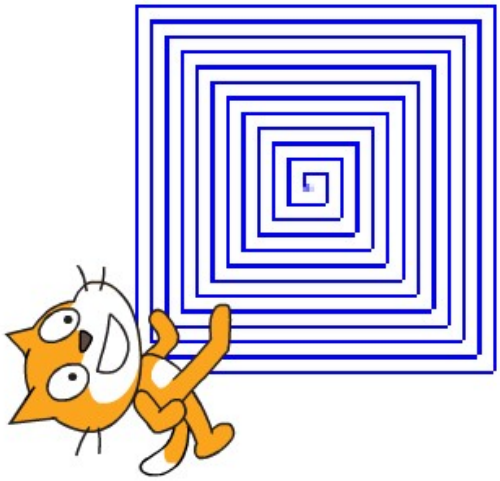
- ..systematisch Lösungen zu Problemen entwickeln
- ..verstehen, wie/wo/wieviel Einfluss Algorithmen auf unseren Alltag nehmen
- ..oder wo sie noch Einfluss nehmen könnten... ;)

Toller Videobeitrag zum Thema: bpb - Algorithmisches Denken verstehen...

DEMO: ERWEITERTE FUNKTIONEN SCRATCH

AUFGABE: SPIRALE ZEICHNEN

1. Schreibt ein Programm, dass eine Figur eine Spirale zeichnen lässt



2. Baut dann einen Block, mit dem ihr Spiralen zeichnen könnt

PYTHON

BASICS

VARIABLEN, KONTROLLSTRUKTUREN, FUNKTIONEN

WAS IST PYTHON?

- interpretierte Programmiersprache
- Plattformunabhängig, Open Source
- Anwendungsbereiche insbesondere:
  - Automatisierung
  - Data Science/Machine Learning
  - Physical Computing

VORTEILE VON PYTHON?

- Fokus auf schnellen Ergebnissen und Lesbarkeit(!)
- Vielzahl an Bibliotheken für alle möglichen Anwendungsbereiche
- Große und aktive Community

Laut TIOBE-Index ist Python aktuell Platz 3 der beliebtesten Programmiersprachen hinter Java und C

ZUM VERGLEICH..

PYTHON:

```
print("Hello World!")
```

JAVA:

```
public class HelloWorld {
public static void main(String[] args){
        System.out.println("Hello World!");
}
}
```

HELLO WORLD!

- **`print()`** druckt etwas auf der Standardausgabe aus

```
print("Hello World!")
```

```
print(5+7)
print(5 < 6)
```

```
print("5^2 ist", 5**2)
```

EINGABEN LESEN MIT `input()`

- input() ermöglicht das Speichern von Benutzereingaben
- Nützlich für einfache Scripte

```
name = input("Wie heißt du?")
print("Grüß dich, ", name)
```

VARIABLEN

- Variablen ermöglicht Daten zwischenzuspeichern
- **`Name = Wert`**
- Daten besitzen einen Datentyp:
  - Ganzzahl (Integer): **`int`** (5,7,123 ....)
  - Fließkommazahl (Float): **`float`** (5.33, 8.4, 9.0)
  - Text/Zeichenlisten (String): **`str`**
    ("Hallo", "x3&32sp\n", "17")
  - Wahrheitswerte: **`Boolean`** (**`True/False`**)

DRILL TIME!

KONTROLLSTRUKTUREN

- Programm läuft normal sequentiell ab
- Kontrollstrukturen: steuern Programmfluss
- 2 Arten von Kontrollstrukturen:
  - Bedingungen: **`if/elif/else`**
    "Unter welcher **Bedingung** soll dieser Code laufen?"
  - Schleifen: **`for/while`**

  "**Wie oft/wie lange** soll dieser Code laufen?"

BEDINGUNGEN

- "Wenn X, dann Y:"
- **`if/elif/else`**
- Code, der zu Bedingung gehört, wird eingerückt

```
alter = int(input("Wie alt bist du?: "))

if alter >= 18:
        print("Du darfst wählen gehen!")
else:
        print("Du darfst noch nicht wählen
```

```
anzahlTage = 24
if anzahlTage <= 7:
        print("Gebühren: 1€")
elif anzahlTage <= 14:
        print("Gebühren: 5€")
elif anzahlTage <= 30:
        print("Gebühren: 10€")
else:
        print("Gebühren: 20€")
```

SCHLEIFEN

- dienen dazu Anweisungen wiederholt auszuführen
- 2 Arten von Schleifen
  - for-Schleife
  - while-Schleife
- Unterschied zwischen den Schleifen rein semantisch

FOR-SCHLEIFE

Schleife, die in den meisten Fällen eine feste Anzahl Durchläufe hat

Beispiel:

```
for i in range(20):
        print(i)
```

WHILE-SCHLEIFE

Schleife, die solange läuft bis eine Abbruchbedingung eintritt

```
i = 0
while i < 20:
        print(i)
        i += 1
```

FUNKTIONEN

- Gruppe von Anweisungen - mit Funktionsnamen aufrufbar
  - können zusätzliche Daten übergeben bekommen (**Parameter**)
  - Kann Ergebnisse zurückliefern: **`return`**
- Python bringt Funktionen mit (built-in-Funktionen)
- Funktionen können selbst **`def`**iniert werden

BUILT-IN-FUNKTIONEN

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Dokumentation: built-in-Funktionen

EIGENE FUNKTION DEFINIEREN

- Mithilfe des Keywords **`def`** können eigene Funktionen definiert werden
- Parameter in runden Klammern
- Optional: Daten mit **`return`** zurückgeben

```
def greeting(name):
        print("Hello", name)
```

```
def potenz(basis,exponent):
        return basis**exponent
```

BEISPIEL: `print()`-FUNKTION

```
print(*objects, sep='', end='\n', ...)
```

- Parameter
  - **`*objects`**: Liste an auszugebenden Objekten
  - **`sep`**: Trennzeichen (Standard: "")
  - **`end`**: Abschlusszeichen (Standard: "\n")

```
print("Ananas","Kiwi","Mango")
```

```
print("Ananas","Kiwi","Mango",sep="---")
```

```
print("Ananas", end="\n...\n")
print("Kiwi", end="\n***\n")
print("Mango", end="\n___\n")
```

DRILL TIME!

FORTGESCHRITTENE THEMEN

DATENSTRUKTUREN, BIBLIOTHEKEN

DATENSTRUKTUREN

- Programme arbeiten oft mit großen Datenmengen
  - Tabellen
  - Ordner..
- Tausende Variablen werden unhandlich..
- (Strings): Zeichen*listen*
- Listen:**`list`**
- Tupel: **`tuple`**
- Dictionaries: **`dict`**

STRINGS

- Zeichenketten - Liste von darstellbaren Zeichen
- Strings kommen überall vor..:
  - Benutzernamen
  - Passwörter
  - Blogtext
  - Chatnachrichten

STRING-FUNKTIONEN

- Zeichen/Teile ersetzen/formatieren
- Auf bestimmte Zeichen prüfen
  - Groß/Kleinbuchstaben
  - Sonderzeichen
  - Schlüsselwörter
- wiederkehrende Aufgaben -> Funktionen

STRINGS: HÄUFIGE FUNKTIONEN

- **`upper()` `lower()`**
  String klein/groß machen
- **`replace("ä","ae")`**
  Ersetze ä durch ae
- **`isupper()` `islower()`**
  prüft String auf bestimmte Eigenschaften

STRINGS: HÄUFIGE FUNKTIONEN

- **`startswith("+49")` `endswith("ing")` `contains("lol")`**
  überprüft ob bestimmte Zeichen am Start/Ende /zwischendrin vorkommen
- **Tipp: Autovervollständigung nutzen (TAB-Taste)**

Dokumentation: Strings (Google Python Kurs)

LISTEN

- ermöglicht Listen von Daten zu speichern
- `my_list = ["beer", 5.0, 3, True]`
- Daten können hinzugefügt/entfernt/abgefragt werden
- Kann Daten unterschiedlicher Typen enthalten, meist jedoch nur 1 Typ

LISTEN: FUNKTIONEN

- Für eindimensionale Daten (~ Excelzeile)
- vordefinierte Funktionen zB. für:
  - Listen sortieren **`sort()`**
  - Kommt X vor? **`contains()`**
  - Wie oft kommt X vor? **`count()`**
  - **Tipp: Autovervollständigung nutzen (TAB-Taste)**

Dokumentation: Listen (Google Python Kurs)

SLICING

- Teile aus Datenstruktur entnehmen
- Use Cases z.B.:
  - Entferne den letzten Nutzer aus der Warteschlange
  - Gib mir die ersten/letzten 10 Einträge `[:10]`
  - Teile in N gleich/verschieden große Teile
- Indexierung beginnt bei 0
- Notation `[start:stop:step]`

DICTIONARIES

- Datenstruktur die eine Art Tabelle darstellt
- Schlüssel-Wert-Paare
  - Schlüssel muss eindeutig sein
  - Wert kann mehrmals vorkommen

Dokumentation: Dictionaries(Google Python Kurs)

JSON

- **J**ava**S**cript **O**bject **N**otation
- JavaScript-Dictionary
- Beliebtes Format zum Austausch von Datenobjekten

Youtube Trends als JSON-Antwort

BIBLIOTHEKEN

- Codepakete zu einem bestimmten Themenbereich
- Ähnlich zu Plugins aus Wordpress
- Morgen lernen wir kennen:
  - **`pandas`** - DataScience-Bibliothek
- Vor der Benutzung müssen diese installiert und importiert werden

ALLES RUND UM ZUFALL - `random`

- Bibliothek die alles rund um Zufall mitbringen
- Zufallszahlen erzeugen: **`randint(start,stop)`**
- Zufällige Elemente wählen: **`choice()` `sample()`**
- Listen mischen: **`shuffle()`**

Dokumentation: random-Bibliothek

KOMBINATORIK LEICHT GEMACHT - `itertools`

- Bibliothek die bei kombinatorischen Problemen hilft
- "Bastel mir alle Kombinationen aus diesen 5 Buchstaben"
- "Wieviele mögliche Pokerhände gibt es?"
  - **`product(list,repeat=)`**
  - **`permutations(list,n)`**
  - **`combinations(list,n)`**

Dokumentation: itertools-Bibliothek

FRAGEN? FEEDBACK?

BIS MORGEN!