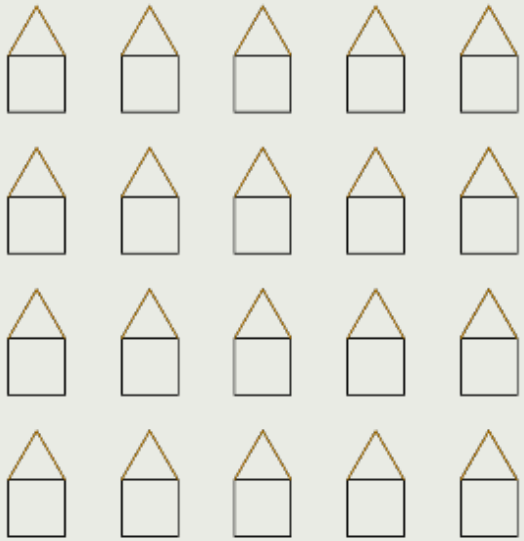
ETH zürich

ABZ AUSBILDUNGS- UND BERATUNGSZENTRUM FÜR INFORMATIKUNTERRICHT

Allgemeinbildender Programmierunterricht – tut auch der Mathematik gut!

Vortragsreihe der ETH Zürich und der PH Zürich:

Vom Kindergarten bis zur Hochschule – Mathematik im Unterricht heute



Giovanni Serafini

Agenda

1. Begrüssung und Einführung (5‘)
2. Programmierunterricht im Spiralcurriculum (20‘)
3. Der modulare Entwurf (20‘)
4. Die Denkweise der Informatik (15‘)
5. Zusammenfassung und Abschluss
6. Diskussion (30‘)

1. Begrüssung und Einführung

Who are we?

- ETH Zürich, Departement Informatik,
  Professur für Informationstechnologie und Ausbildung
- **Leitung**: Prof. Dr. Juraj Hromkovic
- Giovanni Serafini
  - Dozent für Didaktik der Informatik
  - SVIA-Vorstand
  - Leitungskomitee des "Niklaus Wirth Young Talent Computer Science Award" von SI, SVIA & ABZ.
  - Ausbildung der PrimaLogo-Lehrpersonen

Ausbildungs- und Beratungszentrum für Informatikunterricht der ETH Zürich

- **Leitung:** Prof. Dr. Juraj Hromkovic
- **Aufgaben:** Unterstützt Schulen und Lehrkräfte der Volks- und Maturitätsschulen, die ihren Informatikunterricht entsprechend auf- oder ausbauen möchten.
- **Aus dem Angebot:**
  - Aus- und Weiterbildungskurse für Lehrpersonen
  - Entwickeln von Unterrichtsmaterialien
  - Workshops für Schüler (und Lehrpersonen)
  - Schweizer Tag für den Informatikunterricht (STIU)
- **Webseite:** www.abz.inf.ethz.ch

Materialien zum Vortrag

- Logo-Programmierumgebung, Slides, Unterlagen, zusätzliche Literatur:
  - https://tinyurl.com/xlogo-2017
- Programmierumgebung: **XLogoOnline**
  - Offline-Version für Windows, macOS, Linux
  - (Online-Version: http://ethz.ch/xlogo)
- „All about Logo“ auf der Webseite des ABZ:
  - http://abz.inf.ethz.ch/logo
- „All about Python and **TigerJython**“:
  - http://www.tigerjython.ch/

2. Programmierunterricht im Spiralcurriculum

Prämisse

- Schwerpunkt des Vortrags: Programmierunterricht
  - Der Programmierunterricht ist ein wesentlicher, auch wenn natürlich nicht der einzige, Teil des Informatikunterrichts.
  - "Programmieren ist nicht alles, aber ohne programmieren ist alles nichts."
- Die Schule hat den Auftrag, den Kindern und den Jugendlichen "**etwas fürs Leben**" beizubringen. Dies gilt selbstverständlich auch für den Informatikunterricht.
- Der Risiko ist gross, sich von der kurzlebigen Technologie "verführen" zu lassen, oder auf die Features einer Programmiersprache zu fokussieren.

(Fachdidaktische) Kernfragen

- Was macht den allgemeinbildenden Programmierunterricht aus?
- Wie gestaltet man allgemeinbildenden Programmierunterricht?
- Der Vortrag zeigt dies exemplarisch und reflektiert die fachdidaktische Vision und die Aktivitäten des ABZ im Kontext des allgemeinbildenden Programmierunterrichts.

Schulprojekt für Kinder und deren Klassenlehrpersonen

- Typische Altersstufe: 5. und 6. Schuljahr (Primarschule)
- Unterricht: Vor Ort, an der Primarschule.
- Arbeitsmaterial (pro Kind):
  - Heft „Programmieren mit LOGO“
  - Computer mit XLogoOnline
- Zeitrahmen: 20 Lektionen (10 Einheiten à je 2 Lektionen)
- Die Klassenlehrperson assistiert. Wir unterrichten.

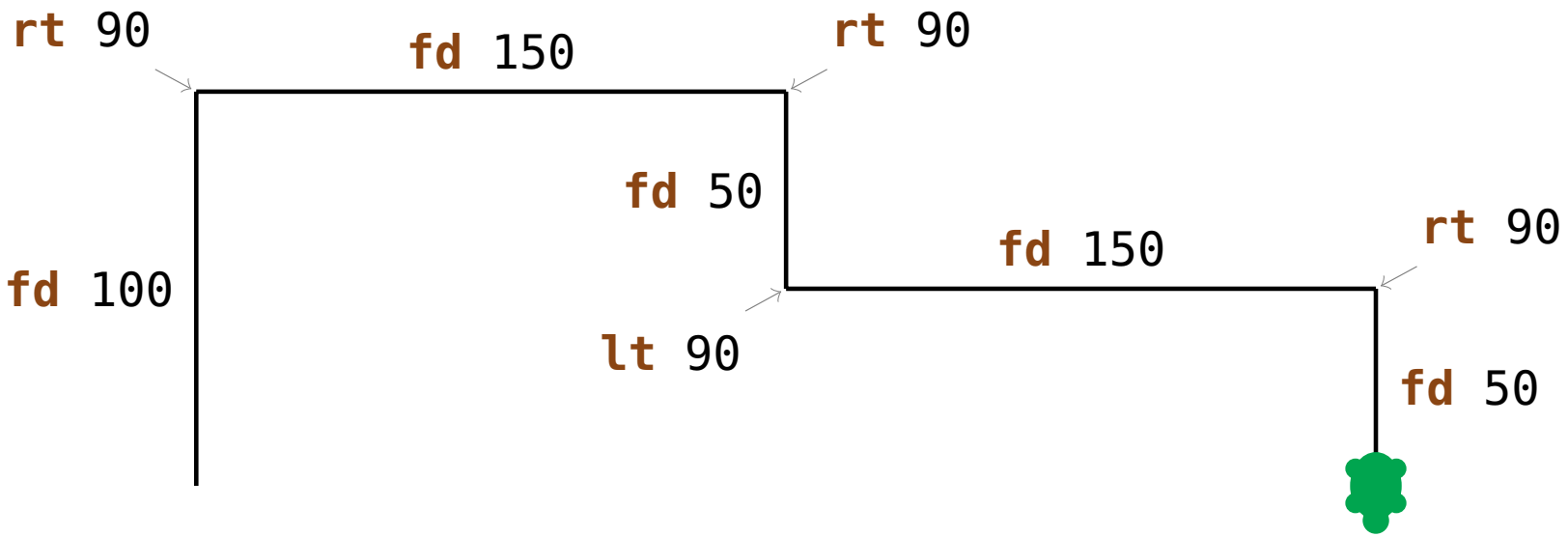
Beispiel 2.1: Eine Schildkröte bewegt sich auf den Bildschirm

(Aufgabe 1 im Heft): Tippe das folgende Programm ab und führe es aus:

```
fd 100
rt 90
fd 150
rt 90
fd 50
lt 90
fd 150
rt 90
fd 50
```

Beispiel 2.1: Fortsetzung

Hast du folgendes Bild gezeichnet?



Beispiel 2.2: Ein Quadrat

- Lektion 1: Zeichne ein Quadrat:

```
fd 100 rt 90
fd 100 rt 90
fd 100 rt 90
fd 100 rt 90
```

- Lektion 2: Ein Quadrat mit der Schleifen-Kontrollstruktur („repeat“)

```
repeat 4 [fd 100 rt 90]
```

ohne Variablenbegriff!

Beispiel 2.3: Ein neues Wort

- Lektion 3: Programme mit Namen

```
to QUADRAT
  repeat 4 [fd 100 rt 90]
end
```

Kinder und Programmierunterricht

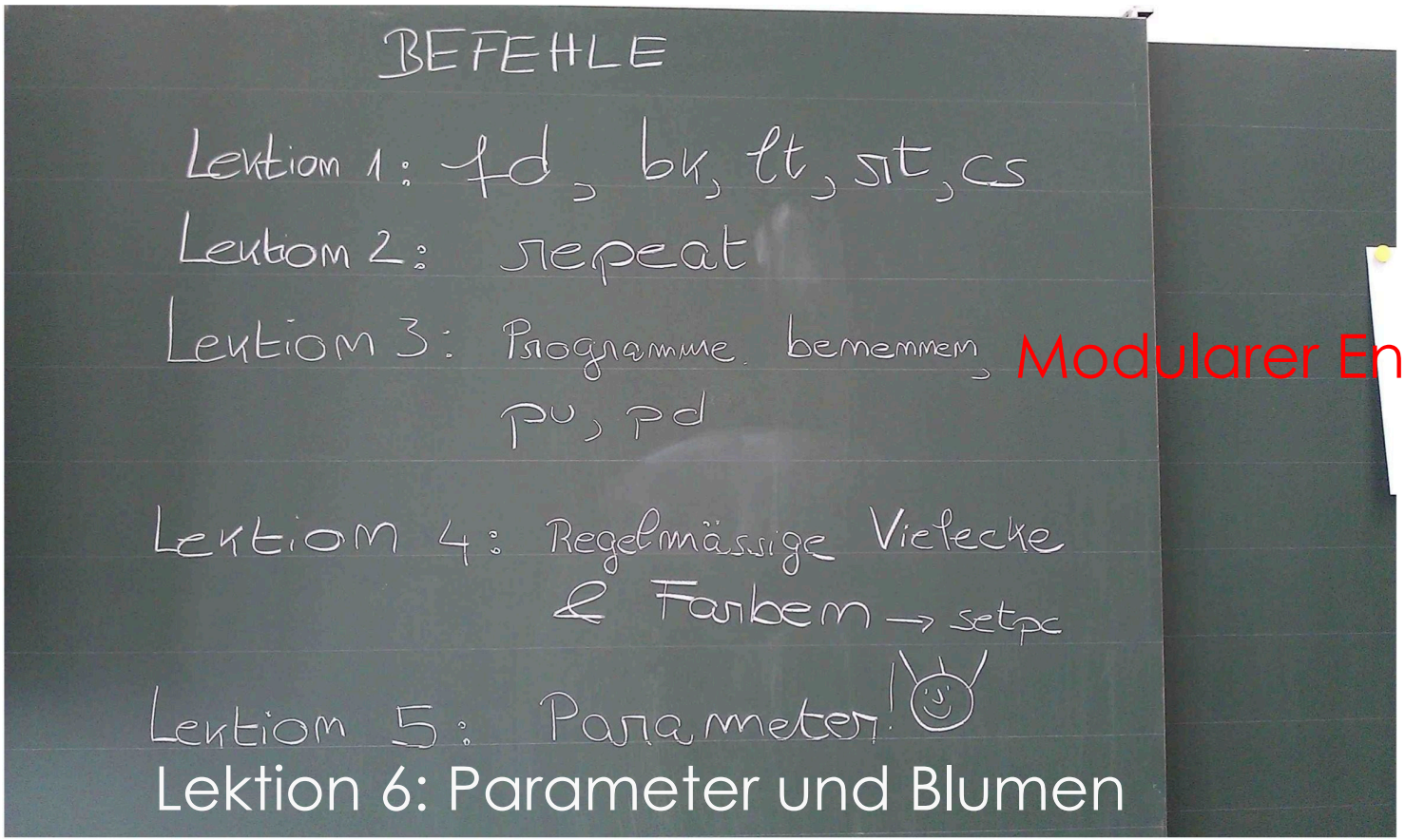
- Wir setzen die Programmiersprache Logo ein.
- Logo wurde am Ende der 1960er am MIT von Seymour Papert entwickelt.

http://dailypapert.com/wp-content/uploads/2011/01/best-turtle-with-citation.jpg (6.10.2011)

Die didaktischen Vorteile von Logo

- Logo wurde explizit für den Unterricht konzipiert.
- Die Kinder schreiben ihres erstes lauffähiges Programm innerhalb weniger Minuten.
- Die erste Kontrollstruktur (Schleife, mit „repeat“) kann ohne Variablenbegriff eingeführt werden.
- Der modulare Entwurf lässt sich sehr elegant umsetzen.
- Das Feedback erfolgt „visuell“. Die Fehlersuche ist „eine spannende Detektivarbeit“. Sie erfolgt zudem spontan.
- Die einfache Textsprache erzieht zur Genauigkeit, dies syntaktisch sowie semantisch.

Curriculum für die Primarschule



Modularer Entwurf

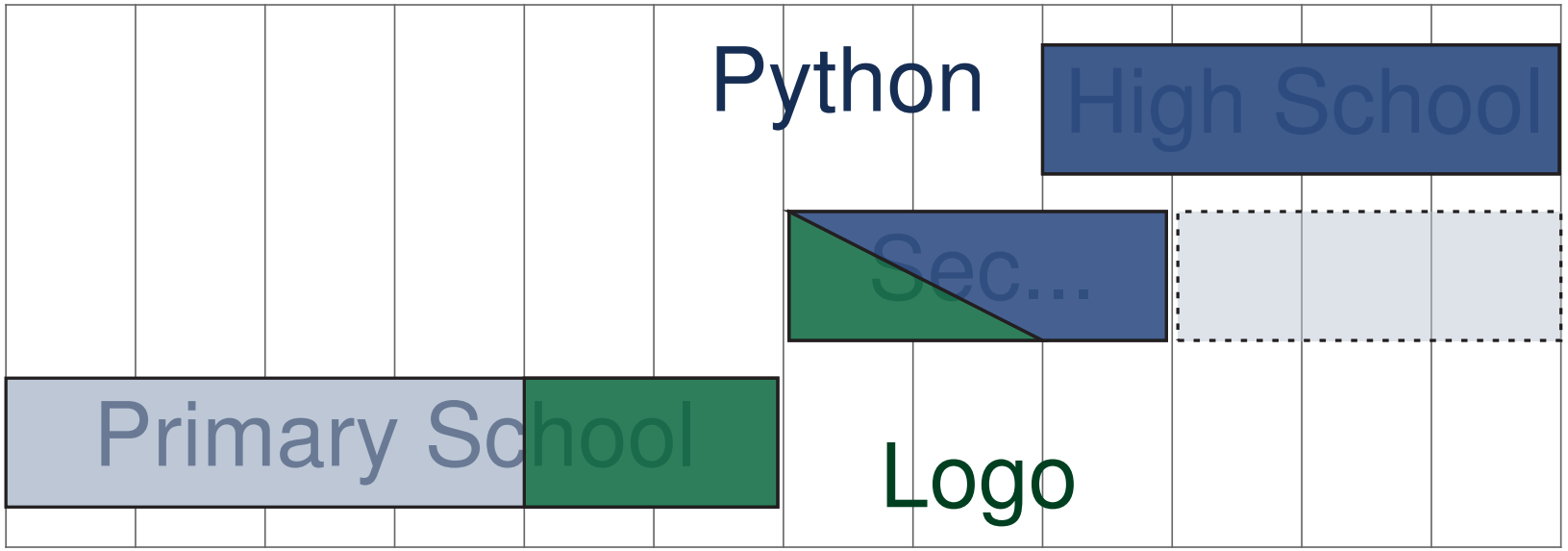
Lektion 6: Parameter und Blumen

Lektion 7: Animationen

Curriculum für die Primarschule

- Programm als Folge einfacher Grundbefehle
- Schleifen als Teil der strukturierten Programmierung
- Unterprogramme als Module zum Bau komplexer Programme
- Konstante Parameter als Grundkonzept des Programmierens
- Lokale und globale Parameter und Übertragung von Daten zwischen Programmen
- Programmierung von Animationen
- ~~Variablen als Grundkonzept des Programmierens~~

Und nach der Primarschule?



Curriculum für die Sek I&II am Beispiel von Logo

- Variablen als Grundkonzept des Programmierens 
- Lokal und globale Variablen
- Verzweigungen von Programmen und **while**-Schleifen
- Rekursion

Lektion 6: Parameter und Blumen

Der Unterricht in der Sek I & II

- Wir haben uns für die Programmiersprache Python und die Programmierumgebung **TigerJython** entschieden.
- **TigerJython** wurde z.T. bei uns am ABZ von meinem Kollegen Tobias Kohn entwickelt.
- Python hat sich in unserer Arbeit als eine natürliche Folgesprache zu Logo für den Programmierunterricht mit älteren SchülerInnen erwiesen.

3. Der modulare Entwurf

Beispiel 3.1: Modularer Entwurf – ganz einfach

- Ausgangslage & Vorwissen:
  - Lektion 2 des Logo-Hefts
  - Programme mit Namen noch nicht bekannt.
- Wir wollen das folgende Bild mit Hilfe des Befehls **repeat** zeichnen.

(Beispiel am Hellraumprojektor)

Beispiel 3.2: Modularer Entwurf – ein Schritt weiter

- Ausgangslage & Vorwissen:
  - Lektion 3 des Logo-Hefts
  - Programme mit Namen sind nun bekannt.
- Wir wollen das folgende Bild mit Hilfe eines **Unterprogramms** zeichnen.

(Beispiel am Hellraumprojektor)

Beispiel 3.3: Modularer Entwurf – Parameter und noch mehr

- Ausgangslage & Vorwissen:
  - Lektion 6 des Logo-Hefts. Konzept des Parameters bereits bekannt.
  - Übergabe von Parameterwerten an Unterprogramme wird gerade gelernt.
- Wir wollen das folgende Bild mit Hilfe eines **Unterprogramms** zeichnen.
- Die Länge der Linien wird mit dem Parameter **:GR** angegeben.

(Beispiel am Hellraumprojektor)

Beispiele 3.1 – 3.3: Reflexion

- Beispiel 3.3: [5./6. Klasse]
  - Lektion 6 des Logo-Hefts. Konzept des Parameters bereits bekannt.
  - Übergabe von Parameterwerten an Unterprogramme wird gerade gelernt.
- Beispiel 3.2: [4. Klasse]
  - Lektion 3 des Logo-Hefts.
  - Programme mit Namen sind nun bekannt.
- Beispiel 3.1 [3. Klasse]
  - Lektion 2 des Logo-Hefts
  - Programme mit Namen noch nicht bekannt.

Leitidee des Programmierunterrichts des ABZ

- Die primäre Zielsetzung des Unterrichts besteht darin, den Kindern **eine Sprache** beizubringen.
- Eine **spezifische Programmiersprache** (egal welche) mit all ihren speziellen Features ist immer ein Produkt der Wissenschaft bzw. der Arbeit, die jemand anderes geleistet hat.
- Wir möchten, dass die Kinder lernen, wie sie schrittweise den **Wortschatz** (der Schildkröte) erweitern können, indem sie ihr **neue Wörter** und deren **Bedeutung** beibringen, welche von ihnen in Form neuer **Programme** formuliert werden.
- Die Kinder entwickeln **ihre eigene Programmiersprache**. Der **modulare Entwurf** ist unser Weg.

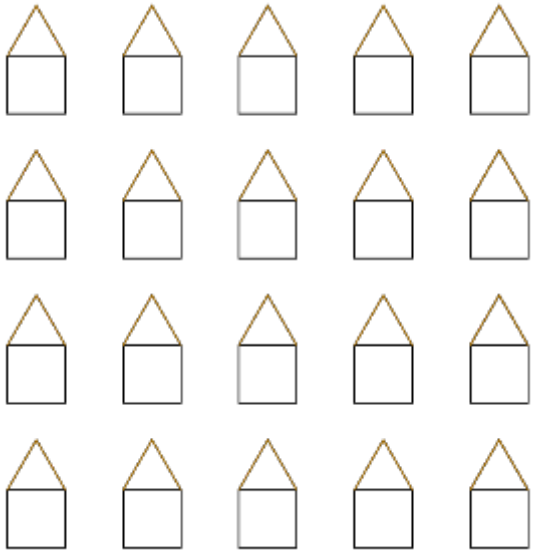
Beispiel 3.4: Ein selbstgebauter Kreis

- Was passiert, wenn wir ein regulärer Vieleck mit seeeehr vielen Ecken zeichnen?

```
to VIELECK
  repeat 1000 [fd 1 rt 360/1000]
end
```

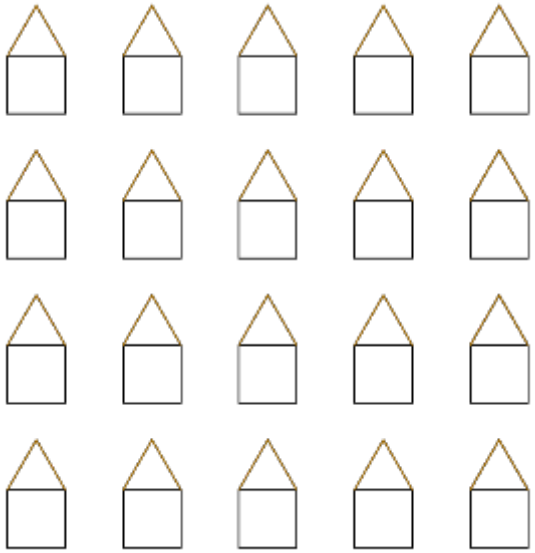
- Die Kinder entwickeln eines eigenes Programm, um einen Kreis zu zeichnen. Sie brauchen den eingebauten Kreisbefehl gar nicht!

Beispiel 3.5: Vom Haus zur Siedlung



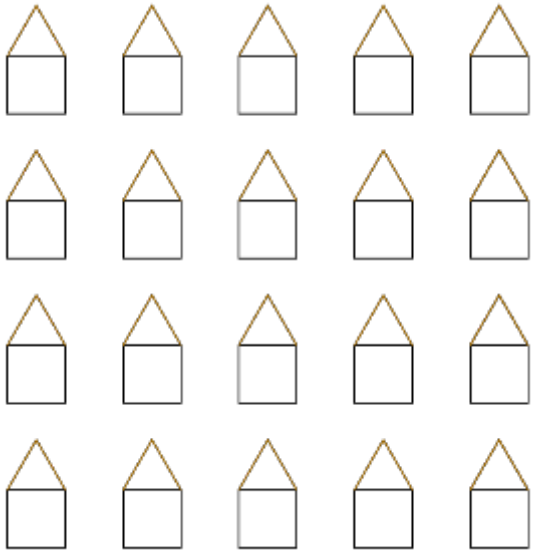
```
to HAUS
  rt 90
  repeat 4 [fd 50 rt 90]
  lt 60 fd 50 rt 120 fd 50
  lt 150
end
```

Beispiel 3.5: Vom Haus zur Siedlung



```
to HAUS
  rt 90
to HAUSREIHE
  repeat 5 [
  HAUS
  rt 90 pu fd 50 lt 90 pd ]
end
```

Beispiel 3.5: Vom Haus zur Siedlung



```
to HAUS
  rt 90
```

```
to HAUSREIHE
  repeat 5 [
  HAUS
```

```
to SIEDLUNG
  repeat 5 [
    HAUSREIHE
    lt 90 pu fd 500 rt 90
    fd 100 pd ]
end
```

Beispiel 3.6: Schleife im Spiralcurriculum

1. Schritt: Ohne Variablen

Logo

```
repeat 4 [ forward 100 left 90 ]
```

Python

```
repeat 4:
  forward(100)
  left(90)
```

Spezialität von TigerJython

Beispiel 3.6: Schleife im Spiralcurriculum

2. Schritt: Mit Parametern, aber immer noch ohne Variablen

```
to polygon :n
  repeat :n [ forward 100 left 360/:n ]
end
```

```
def polygon(n):
  repeat n:
    forward(100)
    left(360/n)
```

Beispiel 3.6: Schleife im Spiralcurriculum

3. Schritt: Nun mit Variablen, aber nicht im Schleifenkopf

```
x = 30
repeat 20:
  forward(x)
  left(90)
  x += 10
```

Beispiel 3.6: Schleife im Spiralcurriculum

4. Schritt: Mit einer Zählvariable im Schleifenkopf

```
def polygon(n):
  repeat n:
    forward(100)
    left(360/n)
```

4. Die Denkweise der Informatik

Die Informatik ist jung & dennoch alt

- Die Informatik ist die Wissenschaft, die sich mit der **Automatisierung der intellektuellen Arbeit** befasst.
- Sie ist **eine junge Wissenschaft**, aber ihre **Denkweise** trägt **seit jeher zur Entwicklung der Menschheit** bei.
- Vor 5000 Jahren entwickelten die Sumerer die erste Schrift, um Daten ihres Reiches mit 1 Mio. Einwohnern zu speichern und zu bearbeiten (Steuern, Eigentum).

Informatik als Wissenschaft

Informatik entstand als eigenständige Wissenschaft als:

- die Zielsetzungen und Aufgabenstellungen so genau verstanden und Vorgehensweisen zu ihrer Erreichung/ Lösung so exakt beschrieben wurden, dass es keine Improvisationsfähigkeit (also keine intellektuelle Leistung) mehr brauchte, um sie durchzuführen,
- die Technologie es ermöglichte, die Ausführung der Algorithmen den Maschinen zu überlassen.

Denken wie ein Informatiker

- Wir möchten den Kindern beibringen, wie ein Informatiker zu denken.
- Diese Denkweise (und nicht die Produkte der Wissenschaft) stellt den Beitrag der Informatik zur allgemeinen Bildung dar, und ist unter den Bezeichnungen
  - „Algorithmisches Denken“
  - „Computational Thinking“
  - „Algorithmic Thinking“

  bekannt.

Algorithmisches Denken

- Knuth (1980)

  Knuth informally described algorithmic thinking as his "perception of the most typical thought processes used by a computer scientist".

- Aho (2012)

  We consider computational thinking to be the thought process involved in formulating problems so their solutions can be represented as computational steps and algorithms.

- Cuny, Snyder, Wing (2010)

  Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.

Was ist Informatik? Was ist sie nicht?

- Eine Definition von Informatik:

  „Die Informatik ist die Wissenschaft der systematischen, automatisierten Verarbeitung von Information, der Informationsspeicherung, -verwaltung und -übertragung. [...]“

- Sie ist ein MINT-Fach!
- Informatik ≠ Medienbildung
- Informatik ≠ Anwendungen (Textverarbeitung, usw.)

Informatik in der Schule

- Gymnasium:
  - Die EDK entscheidet **Ende Oktober**, ob Informatik ein Grundlagenfach wird.
- Volksschule (Lehrplan 21):
  - Die Schülerinnen und Schüler können Daten aus ihrer Umwelt darstellen, strukturieren und auswerten.
  - Die Schülerinnen und Schüler können einfache Problemstellungen analysieren, mögliche Lösungsverfahren beschreiben und in Programme umsetzen.
  - Die Schülerinnen und Schüler verstehen Aufbau und Funktionsweise von informationsverarbeitenden Systemen und können Konzepte der sicheren Datenverarbeitung anwenden.

… tut auch der Mathematik gut!

Auf Grund der Erfahrung im Logo-Unterricht, wage ich es, drei Hypothesen aufzustellen:

1. Im Logo-Unterricht beobachten die Klassenlehrpersonen oft, dass Kinder, die Mühe mit dem **Winkelbegriff** in der Geometrie haben, einen neuen Zugang zu diesem Konzept entwickeln. Liegt es an der Tatsache, dass sie die Winkel „konstruieren“?

2. Die frühe Einführung des **eher konkreten Variablenbegriffs** in der Informatik bzw. im Programmierunterricht vereinfacht die spätere Thematisierung des **eher abstrakten Variablenbegriffs** im Mathematikunterricht.

… tut auch der Mathematik gut!

3. Der Programmierunterricht erzieht zur Genauigkeit:
   - Semantisch: Man beschreibt einen Lösungsweg so genau, dass es eine Maschine (ohne Intellekt) dies durchführen kann.
   - Syntaktisch: Wenn die „Sprachregeln“ nicht eingehalten werden, läuft gar nichts. Diese Genauigkeit ist im Mathematikunterricht oft leider nur „nice to have“.

5. Zusammenfassung und Abschluss

Zusammenfassung und Abschluss

- Informatik ist eine junge Disziplin. Die Denkweise der Informatik ist jedoch „so alt wie die Menschheit“.
- Algorithmisches Denken „lernt man fürs Leben“.
- Spezifische Programmiersprachen sind selber ein **Produkt** der Wissenschaft.
- Darum: Wir bringen den Kindern bei, **wie** sie eine eigene Sprache entwickeln, um mit dem Computer zu kommunizieren, dies ausgehend von einem kleinen Grundwortschatz.
- Der **modulare Entwurf** ist unser fachdidaktisches Werkzeug.

Literaturempfehlungen

[00] Warum Informatik in der Schule?

[01] Homo Informaticus

[02] Informatik und allgemeine Bildung

[03] Das Zeitalter der Informatik

[04] Lehrplan21: Medien und Informatik

[05] Programmieren, so wichtig wie schreiben und lesen

[06] Combining the Power of Python with the Simplicity of Logo for a Sustainable Computer Science Education

[07] Programmieren macht Schule

[08] Programmierunterricht für Kinder und deren Lehrpersonen

[09] Examples of Algorithmic Thinking in Programming Education