

# **Python.Processing.Arduino**

**Eine Einführung in die Hard- und Softwareentwicklung**

Stand: 13. Oktober 2015

Dieses Script ist eine offene Bildungsressource (OER). Die Autorinnen und Autoren, die daran mitgearbeitet haben, können unter <https://github.com/xldrkp/InformatikII/graphs/contributors> eingesehen werden. Über die Möglichkeiten von Github kann jeder Interessierte auf unterschiedliche Weise an dem Dokument mitarbeiten und an den eigenen Lehrkontext anpassen.

Fork and contribute!



Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung 4.0 International Lizenz. Um eine Kopie dieser Lizenz einzusehen, besuchen Sie bitte <http://creativecommons.org/licenses/by/4.0/>.

# Inhaltsverzeichnis

<b>1 Vorwort</b>	<b>9</b>
1.1 Die Motivation für Python in der gewerblich-technischen Berufsbildung . . . . .	10
1.2 Python und Processing vereint . . . . .	12
1.3 Dieses Buch, ein <i>Living Document</i> . . . . .	12
<b>2 Einführung</b>	<b>15</b>
2.1 Was ist Processing? . . . . .	16
2.2 Was ist Python? . . . . .	17
2.3 Was ist Processing.py? . . . . .	19
2.4 Was ist Arduino? . . . . .	22
2.5 Installation und Konfiguration . . . . .	22
2.6 Los geht's! . . . . .	24
2.7 Fehlermeldungen lesen und lieben . . . . .	25
<b>3 Programmiergrundlagen</b>	<b>27</b>
3.1 Variablen . . . . .	28
3.2 Verzweigungen . . . . .	28
3.3 Schleifen . . . . .	28
3.4 Funktionen . . . . .	28
3.5 Funktionen mit Rückgabewert . . . . .	28
3.6 Interaktionen mit Maus und Tastatur . . . . .	28
3.7 Listen, Sets und Tupel . . . . .	28
3.8 Operatoren . . . . .	28

<b>4</b>	<b>Größere Zusammenhänge</b>	<b>29</b>
4.1	Bilder und Videos einbinden und manipulieren . . . . .	29
4.2	Bibliotheken einbinden und verwenden . . . . .	29
4.3	Objektorientierte Programmierung (OOP) . . . . .	29
<b>5</b>	<b>Arduino</b>	<b>31</b>
5.1	Serielle Kommunikation . . . . .	31
5.2	Processing und Arduino . . . . .	31
<b>6</b>	<b>Literaturverzeichnis</b>	<b>33</b>

# Abbildungsverzeichnis

1.1 Studentisches Projekt (Arduino/Processing/Java): Lernumgebung für auszubildende ElektronikerInnen in der Fachrichtung Energie- und Gebäudetechnik. Quelle: Axel Dürkop (CC-BY) . . . . .	11
2.1 Addition zweier Zahlen mit Python. Ein- und Ausgabe erfolgen auf der Kommandozeile. . . . .	18
2.2 Durch die unterschiedliche Abfolge der Farben in den Zeilen entsteht eine Wahrnehmungsverschiebung, obwohl die beteiligten Farben oben und unten gleich sind. . . . .	21
2.3 Arduino UNO unpacked by Nick Hubbard - Flickr: Arduino UNO. Licensed under CC BY 2.0 via Commons . . . . .	22
2.4 Hinzufügen eines Modes in der IDE . . . . .	23
2.5 Installieren des Modes . . . . .	23
2.6 Ein Rechteck zur Begrüßung . . . . .	24
2.7 Voilà! . . . . .	25



# Listings

2.1 Eine Ellipse (Java-Mode) . . . . .	16
2.2 Der Kreis folgt dem Mauszeiger (Java-Mode) . . . . .	16
5.1 Das Blink-Beispiel . . . . .	31



# 1

## Vorwort

Das vorliegende Buch entsteht aus der Erfahrung, die ich mit einer zweisemestrigen Einführungsveranstaltung zur Informatik gemacht habe. An der TU Hamburg-Harburg führe ich seit 2012 zukünftige Gewerbelehrer und -lehrerinnen in die Grundlagen der Programmierung ein und entwickle ihre Kenntnisse und Kompetenzen im Umgang mit Microcontrollern und Webtechnologien. Meine Veranstaltungen waren bisher erfolgreich und haben in jedem Jahrgang gute bis erstaunliche Ergebnisse hervorgebracht. Dazu beigetragen haben auch meine Tutorinnen und Tutoren, die die Veranstaltung begleiten und die Studierenden jede Woche bei ihren Aufgaben und Projekten unterstützen.

## **1.1 Die Motivation für Python in der gewerblich-technischen Berufsbildung**

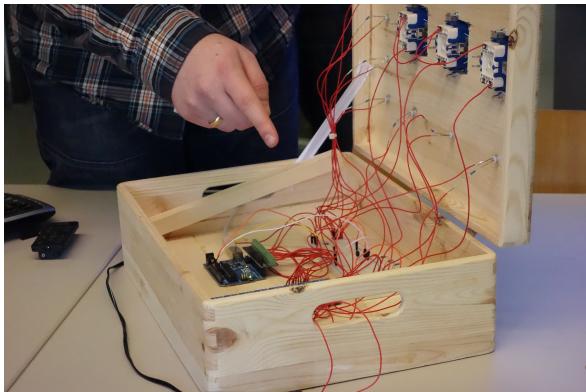
Um die Motivation für den Ansatz *Python mit Processing lernen* zu erklären, will ich kurz die Rahmenbedingungen beschreiben. Die Teilnehmenden meiner Veranstaltung sind Studierende des 3. und 4. Fachsemesters in den Studiengängen Medientechnik und Elektrotechnik-Informationstechnik. Sie werden Berufschullehrende im gewerblich-technischen Bereich und wollen später Auszubildende medientechnischer und elektrotechnischer Berufe unterrichten. Dementsprechend groß sind die Anforderungen im weiten Feld der Informationstechnik.

In meiner Einführungsveranstaltung sitzen Studierende beider Fachrichtungen, die fast alle eine abgeschlossene Berufsausbildung oder ein Äquivalent vorweisen können. Sie kommen mit sehr unterschiedlichen Berufserfahrungen und auch ganz verschiedenen Blicken auf Technik und Informatik. Dieser heterogenen Gruppenkonstellation gerecht zu werden, ist und bleibt eine Herausforderung.

Um den beschriebenen Anforderungen entsprechen zu können, hatte ich als Ziel für die erste Hälfte der zweisemestrigen Veranstaltung folgende Ziele definiert:

- Die Studierenden kennen die Grundlagen der Programmierung einschließlich Objektorientierung und sind in der Lage, komplexere Probleme zu lösen.
- Sie können Daten von Sensoren eines Arduino Uno verarbeiten und angeschlossene Aktoren steuern.
- Sie können Daten mithilfe der Programmiersprache Processing visualisieren.

Dieser Ansatz hat sich als sehr erfolgreich erwiesen, da Processing und Arduino einen sehr einfachen Einstieg ermöglichen und schnell Erfolgsergebnisse zeitigen.



**Abbildung 1.1:** Studentisches Projekt (Arduino/Processing/Java): Lernumgebung für auszubildende ElektronikerInnen in der Fachrichtung Energie- und Gebäudetechnik. Quelle: Axel Dürkop (CC-BY)

Den Einstieg in die Programmierung machten die Studierenden mit Processing<sup>1</sup> sowie der Arduino-IDE. Dabei erzielten sie schnelle Erfolge bei gleichzeitig geäußertem Spaß an der Sache.

Im zweiten Semester sollen die Studierenden die Grundlagen moderner Webtechnologien erarbeiten und sich an einem ganzheitlichen Arbeitsprozess aus dem Bereich des Webdesigns abarbeiten. Folgende Ziele sind dafür formuliert worden:

- Die Studierenden kennen die Grundlagen von HTML5 und CSS/CSS3 und können sie für ein selbstgewähltes Projekt einsetzen.
- Die Studierenden können dynamische Webseiten mithilfe der Skriptsprache PHP erstellen.
- Sie kennen die Grundlagen der Serveradministration unter Linux und können ihr Webseitenprojekt unter Realbedingungen ins Netz bringen.
- Sie können Aktoren und Sensoren aus einer Webseite heraus ansteuern (Raspberry Pi).

---

<sup>1</sup> <http://processing.org>

An diesen Zielsetzungen war auch vieles gut, jedoch wurde der Wechsel von Processing/Arduino zu PHP im zweiten Semester als Bruch empfunden. HTML5 und CSS3 sind mittlerweile zu komplexen Konstrukten ausgewachsen, die den Studierenden viel abverlangen, sodass PHP in der verbleibenden Zeit nie in Gänze ausgeleuchtet werden konnte. Die Studierenden beschrieben oft, dass sie erst in den Semesterferien Zeit und Muße gefunden hatten, sich auf PHP einzulassen. Was fehlte, war eine semesterübergreifende Konstante, eine Sprache, auf die die Studierenden im zweiten Semester aufbauen konnten. PHP im ersten der beiden Semester machte keinen Sinn und Processing im zweiten auch nicht.

## 1.2 Python und Processing vereint

Als PHP-Kenner und -Nutzer habe ich Python erst spät kennen und lieben gelernt. Mich hat immer das Web interessiert, und Python hatte ich lange nicht als Websprache wahrgenommen. Durch die Arbeit mit dem Raspberry Pi und den Auftritt des schlanken Webframeworks Flask<sup>2</sup> hat sich das geändert. Mir scheint, das Python gerade für die Zwecke, in denen meine Zielgruppe später agieren wird, eine bessere Wahl ist als PHP (vgl. Kapitel zu Einsatzbereichen von Python). Daher ist das Ziel meiner künftigen Lehrveranstaltung wie auch dieses Buchs, meine Studierenden früh an Python heranzuführen und dabei die guten und positiven Erfahrungen aus der Vergangenheit zu bewahren.

## 1.3 Dieses Buch, ein *Living Document*

Als ich die Portierung von Processing auf JPython<sup>3</sup> entdeckte, erschien es mir sofort möglich, diesen Plan durchzuführen. Das vorliegende Buch ist dafür die Grundlage. Es soll zum einen als Skript zur Veranstaltung dienen, zum anderen aber auch andere einladen, an dem Konzept mitzudenken und mitzuge-

---

<sup>2</sup> <http://flask.pocoo.org/>

<sup>3</sup> <https://github.com/jdf/processing.py>

stalten. Die freie Lizenzierung und das offene Dateiformat sollen dazu beitragen. Ich gehe davon aus, dass die Kapitel eine Überarbeitung im Rahmen der durchgeführten Veranstaltung erfahren, aber auch durch *feature requests* und *pull requests* der Community, die sich mit ähnlichen Themen beschäftigt.



# 2

## Einführung

Für Python braucht es ebenso wenig ein neues Lehrbuch wie für Processing. Der Buchmarkt und das Netz bieten genügend Auswahl für jeden Erfahrungs-horizont (s. Anhang). Die Kombination von Processing *und* Python ist allerdings in der Literatur noch nicht so abgebildet, dass Lernende einen guten Start in die Welt des Programmierens hinlegen können, wenn sie sich für diesen Weg entscheiden.

Daher sind die folgenden Kapitel kein Ersatz für die vielen Grundlagenwerke, die es für Python und Processing gibt. Vielmehr soll versucht werden, den Start ins Coden mit Processing.py möglichst plausibel und anschaulich zu gestalten. Der Erwerb von ersten Python-Kenntnissen steht dabei im Vordergrund, Processing ist nur ein Vehikel. Diese Priorisierung soll die Processing-Gemeinde keineswegs verärgern. Processing bietet ebenfalls faszinierende Möglichkeiten, das Programmieren zu lernen und ernsthafte Projekte umzusetzen, kann aber im Bereich Webtechnologien nicht punkten, wenn es um eine berufsbezogene Beschäftigung mit diesem Themenfeld geht.

Im folgenden sollen die Sprachen Processing und Python einzeln vorgestellt werden, um anschließend ihr Zusammenspiel in Processing.py aufzuzeigen. Das Kapitel endet mit einem kurzen Einblick in die Hardwareplattform Arduino<sup>1</sup> und einem kleinen Programmierbeispiel.

## 2.1 Was ist Processing?

Processing ist ein Dialekt der Programmiersprache Java<sup>2</sup>. Das bedeutet, dass sie Java sehr ähnlich ist, deren Umfang aber um Funktionen erweitert, die die Erstellung graphischer und interaktiver Programme erleichtern<sup>3</sup>. Da Lernende schon mit der ersten Zeile Code ein visuelles Feedback erhalten, fällt ihnen der Einstieg ins Programmieren beobachtbar leichter, als wenn sie mit mathematiklastiger Informatiktheorie beginnen würden.

Gibt man die folgende Zeile Code in der Programmierumgebung von Processing ein (im Java-Mode, vgl. Abschnitt 2.5), wird ein Kreis auf die Leinwand gezeichnet:

---

```
1 ellipse(10,10,20,20);
```

---

Listing 2.1: Eine Ellipse (Java-Mode)

Mit geringen Änderungen folgt der Kreis dem Mauszeiger:

---

```
1 void draw() {  
2     ellipse(mouseX, mouseY, 20, 20);  
3 }
```

---

Listing 2.2: Der Kreis folgt dem Mauszeiger (Java-Mode)

---

<sup>1</sup> <http://arduino.cc>

<sup>2</sup> [https://de.wikipedia.org/wiki/Java\\_%28Programmiersprache%29](https://de.wikipedia.org/wiki/Java_%28Programmiersprache%29)

<sup>3</sup> Wanner (2010) erklärt den Zusammenhang kurz auf S. 59

Um diese Lösung in anderen Programmiersprachen zu erreichen, sind in der Regel viel mehr Vorwissen und Programmiercode notwendig. Der niedrigschwellige Zugang zu Processing macht die Sprache so stark, wenn die Zielgruppe Grundkenntnisse der Programmierung erwerben soll, aber keine tiefergehenden informationstechnischen Ambitionen hat. Vielmehr richtet sich die Sprache an Künstlerinnen und Künstler, an Datenjournalisten und *Data Scientists*, an Musikerinnen und Musiker wie auch an Mathematiker und Mathematikerinnen.

Die Einfachheit im Zugang soll daher nicht täuschen. Mit Processing lassen sich anspruchsvolle Projekte realisieren, die auf der Homepage des Projekts<sup>4</sup> in einer Ausstellung gezeigt werden.

Processing kann durch eine Vielzahl so genannter Bibliotheken erweitert werden. Auch hier empfiehlt sich ein Ausflug auf die entsprechende Seite des Projekts<sup>5</sup>. Folgt man den Links der Bibliotheken, eröffnen Anwendungsbeispiele viele neue Möglichkeiten und beflügeln die Phantasie.

## 2.2 Was ist Python?

Python<sup>6</sup> ist eine universelle Programmiersprache, mit der sich Probleme und Aufgaben aus unterschiedlichsten Bereichen lösen lassen. Python setzt den Schwerpunkt auf Lesbarkeit und Einfachheit und verzichtet weitestgehen auf Klammern und das obligatorische Semikolon am Ende einer Zeile. Stattdessen ist penibel auf die korrekte Einrückung von Codezeilen und -blöcken zu achten. Ein Vergleich der Java-Syntax aus dem vorgegangenen Kapitel mit der Schreibweise in Python soll den Unterschied zeigen:

Das Beispiel in Java-Syntax:

---

```
1 void draw() {
```

<sup>4</sup> <https://processing.org/exhibition/>

<sup>5</sup> <https://processing.org/reference/libraries/>

<sup>6</sup> [https://de.wikipedia.org/wiki/Python\\_%28Programmiersprache%29](https://de.wikipedia.org/wiki/Python_%28Programmiersprache%29)

```
2   ellipse(mouseX, mouseY, 20, 20);  
3 }
```

---

In Processing bzw. Java ist die Einrückung in Zeile 2 optional. Das Semikolon darf nicht fehlen, damit der Compiler<sup>7</sup>, der das Processing-Programm übersetzt, weiß, wo eine Zeile zuende ist.

In Python müssen Zeilen, die zu einem Block gehören, zwingend eingerückt werden, damit der Interpreter<sup>8</sup> die Struktur des Programms korrekt erfassen kann.

---

```
1 def draw():  
2     ellipse(mouseX, mouseY, 20, 20)
```

---

Die Erfahrung aus der Lehre zeigt, dass dieser Zwang zur übersichtlichen Formatierung das Verständnis fördert. Einrückungen sollten konsequent mit der Tabulatortaste vorgenommen werden, eine Mischung mit Leerzeichen kann zu Problemen führen. Das Semikolon am Ende der Zeile ist in Python nicht vorgesehen und führt zu Fehlern bei der Ausführung des Programms. Ein weiteres Beispiel sowie weitergehende Ausführung zur Syntax von Python finden sich in der Wikipedia<sup>9</sup>.

Der Zugang zu Python ist ebenfalls sehr einfach, da sich auch hier schon mit wenigen Zeilen Code erste Ergebnisse erzielen lassen. Allerdings fällt das visuelle Feedback hier eher spärlich aus, denn die ersten Schritte mit Python machen die meisten Lehrbücher auf der Kommandozeile.

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)  
[GCC 4.8.2] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print( 3 + 4 )  
7
```

**Abbildung 2.1:** Addition zweier Zahlen mit Python. Ein- und Ausgabe erfolgen auf der Kommandozeile.

---

<sup>7</sup> <https://de.wikipedia.org/wiki/Compiler>

<sup>8</sup> <https://de.wikipedia.org/wiki/Interpreter>

<sup>9</sup> [https://de.wikipedia.org/wiki/Python\\_%28Programmiersprache%29#Syntax](https://de.wikipedia.org/wiki/Python_%28Programmiersprache%29#Syntax)

Dieser Ansatz, ins Programmieren von Computern einzusteigen, ruft in der Regel nicht bei allen Beteiligten Begeisterung hervor.

Nun bietet auch Python einen eigenen Zugang zur Erstellung von Programmen, die eher graphisch orientiert sind. Zu nennen wären hier Pygame<sup>10</sup> oder auch Turtle<sup>11</sup>. Aber auch hierbei sind schon einige Zeilen Code zu schreiben, bis sich erste Erfolgsergebnisse einstellen, und die begrenzte Erweiterungsmöglichkeit setzt der Phantasie schnell Grenzen.

Das Potenzial von Processing gepaart mit Python ermöglichen einen Einstieg in die Programmierung, der viele Türen öffnet und eine flache Lernkurve verspricht.

## 2.3 Was ist Processing.py?

Etwas unscheinbar wird bisher auf der Homepage von Processing auf Processing.py hingewiesen. Auf der Projektseite<sup>12</sup> heißt es:

„Processing was initially released with a Java-based syntax, and with a lexicon of graphical primitives that took inspiration from OpenGL, Postscript, Design by Numbers, and other sources. With the gradual addition of alternative programming interfaces — including JavaScript<sup>13</sup>, Python<sup>14</sup>, and Ruby<sup>15</sup> — it has become increasingly clear that Processing is not a single language, but rather, an arts-oriented approach to learning, teaching, and making things with code.“

Processing.py ermöglicht folglich den Zugang zu den Möglichkeiten von Processing über die Programmiersprache Python.<sup>16</sup>

In Abschnitt 2.2 wurde gezeigt, wie mit Python ein Kreis in Processing gezeichnet werden kann, der dem Mauszeiger folgt:

---

<sup>10</sup> <http://www.pygame.org/hifi.html>

<sup>11</sup> <http://pythonturtle.org/>

<sup>12</sup> <http://py.processing.org/>

<sup>13</sup> <http://p5js.org/>

<sup>14</sup> <http://py.processing.org/>

<sup>15</sup> <https://github.com/jashkenas/ruby-processing>

<sup>16</sup> Der Name wird in Anlehnung an die typische Dateinamenextension für Pythondateien so geschrieben.

---

```
1 def draw():
2     ellipse(mouseX, mouseY, 20, 20)
```

---

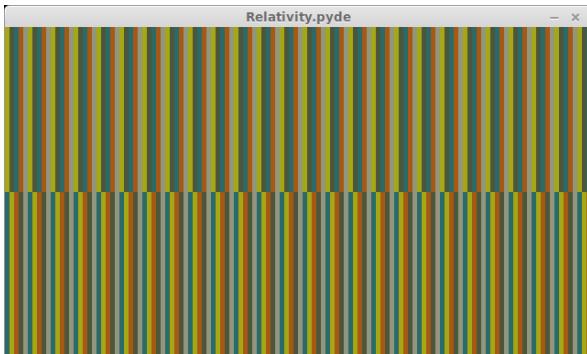
Ein weiteres Beispiel, das der Entwicklungsumgebung entnommen ist, vermittelt, was wenige Zeilen Python-Code ausrichten:

---

```
1 a = color(165, 167, 20)
2 b = color(77, 86, 59)
3 c = color(42, 106, 105)
4 d = color(165, 89, 20)
5 e = color(146, 150, 127)
6
7
8 def setup():
9     size(640, 360)
10    noStroke()
11    noLoop()      # Draw only one time
12
13
14 def draw():
15     drawBand(a, b, c, d, e, 0, width / 128)
16     drawBand(c, a, d, b, e, height / 2, width / 128)
17
18
19 def drawBand(v, w, x, y, z, ypos, barWidth):
20     num = 5
21     colorOrder = (v, w, x, y, z)
22     for i in range(0, width, barWidth * num):
23         for j in range(num):
24             fill(colorOrder[j])
25             rect(i + j * barWidth, ypos, barWidth, height / 2)
```

---

Das Ergebnis sieht auf der Leinwand wie folgt aus:



**Abbildung 2.2:** Durch die unterschiedliche Abfolge der Farben in den Zeilen entsteht eine Wahrnehmungsverschiebung, obwohl die beteiligten Farben oben und unten gleich sind.

Dass der abgebildete Code es in sich hat, soll niemandem vom Weiterlesen abhalten. Im Laufe des Buchs wird klar werden, was die Ausdrücke und Abschnitte des Programms bedeuten.

### 2.3.1 Vorteile von Processing.py

Abschließend lässt sich sagen, dass der Zugang zu Processing über Python große Vorteile hat:

- plattformübergreifende freie Entwicklungsumgebung
- visuelles Feedback schon mit wenigen Zeilen Code
- Erlernen einer Sprache (Python), die viele individuelle Lernwege in unterschiedlichsten Bereichen erlaubt

## 2.4 Was ist Arduino?

Arduino ist eine Open-Source-Hardwareplattform, deren Programmierung sehr einfach ist.



Abbildung 2.3: Arduino UNO unpacked by Nick Hubbard - Flickr: Arduino UNO. Licensed under CC BY 2.0 via Commons

Ein Arduino-Board besitzt einen Mikrocontroller, der ähnlich einfach zu programmieren ist wie Grafiken in Processing oder Processing.py. Ein Arduino-Board kann Messdaten über Sensoren verarbeiten und Aktoren wie LEDs oder Motoren steuern. Durch die Erweiterung mit so genannten Shields kann ein Arduino-Board um bestimmte Funktionen erweitert werden. Weitere Informationen zu Entstehung und Aufbau liefert die Homepage des Projekts<sup>17</sup>.

## 2.5 Installation und Konfiguration

Um Processing mittels Python zu programmieren, muss die Entwicklungsumgebung (IDE - Integrated Development Environment) im Python-Modus gestartet werden.

<sup>17</sup> <https://www.arduino.cc/en/Guide/Introduction>

tet werden. Dieser muss zunächst installiert werden.

Der folgende Vorgang ist analog auf Linux, Mac und Windows durchzuführen.



Abbildung 2.4: Hinzufügen eines Modes in der IDE

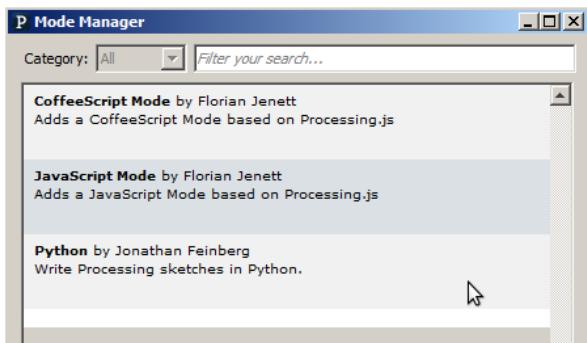


Abbildung 2.5: Installieren des Modes

Der Python-Mode wird ausgewählt und dann mit Klick auf "Install" installiert (vgl. Abb. 2.5). Anschließend muss die IDE neu gestartet und der Python-Mode oben rechts ausgewählt werden. Die IDE startet erneut und befindet sich nun im Python-Mode.

## 2.6 Los geht's!

Jetzt ist es an der Zeit, ein erstes Programm zu schreiben! Wie üblich, wenn man eine neue Programmiersprache lernt, sollte man die Welt wissen lassen, dass man in Zukunft mit von der Partie ist. Begrüßen wir also die Welt mit einer einfachen geometrischen Figur:

---

```
1 rect(10,10,10,10)
```

---

Das entspricht dem *Hello World!*<sup>18</sup> in anderen Sprachen, bei denen man mit Textausgaben beginnt.

In der IDE sieht das folgendermaßen aus:

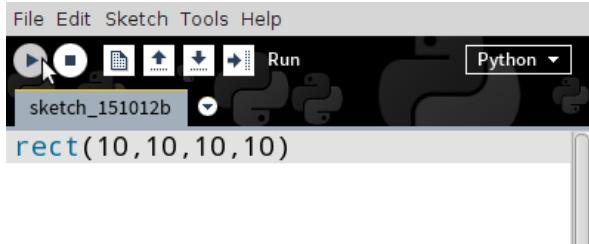


Abbildung 2.6: Ein Rechteck zur Begrüßung

Ein Klick auf den Pfeil oben links startet das Programm und zeigt folgendes Ergebnis:

---

<sup>18</sup> <https://de.wikipedia.org/wiki/Hallo-Welt-Programm>

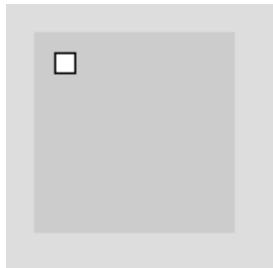


Abbildung 2.7: Voilà!

## 2.7 Fehlermeldungen lesen und lieben

Die Processing-IDE zeigt im unteren Bereich Fehlermeldungen an. Das ist prinzipiell eine gute Sache, auch wenn es natürlich nervt, Fehler zu machen. Am Ende sind es aber die Fehler, die deutlich machen, was richtig ist.

Fehler kann man beim Programmierung auf unterschiedlichste Art und Weise machen. Syntaxfehler, Rechenfehler, Logikfehler u.a. können einem den Einstieg ordentlich verriesen. Dennoch sollte man nicht ins wilde Probieren verfallen, bis die Fehler verschwinden. Stattdessen lohnt sich genaues Lesen und Herausfinden, was die Bedeutung der Fehlermeldung ist.



# 3

# Programmiergrundlagen

3.1 Variablen

3.2 Verzweigungen

3.3 Schleifen

3.4 Funktionen

3.5 Funktionen mit Rückgabewert

3.6 Interaktionen mit Maus und Tastatur

3.7 Listen, Sets und Tupel

3.8 Operatoren

# 4

## Größere Zusammenhänge

**4.1 Bilder und Videos einbinden und manipulieren**

**4.2 Bibliotheken einbinden und verwenden**

**4.3 Objektorientierte Programmierung (OOP)**



# 5

## Arduino

### 5.1 Serielle Kommunikation

### 5.2 Processing und Arduino

---

```
1 add_library('serial')
2 add_library('arduino')
3
4 import time
5
6 led_pin = 13
7 a = Arduino(this, '/dev/ttyACM1', 57600)
8
9
10 class Blink:
11
12     def on(self):
```

```
13         a.pinMode(led_pin, Arduino.HIGH)
14         time.sleep(.2)
15
16     def off(self):
17         a.pinMode(led_pin, Arduino.LOW)
18         time.sleep(.2)
19
20 def setup():
21     a.pinMode(led_pin, Arduino.OUTPUT)
22     b = Blink()
23
24 def draw():
25     Blink.on()
26     Blink.off()
```

---

**Listing 5.1:** Das Blink-Beispiel

# 6

## Literaturverzeichnis

Wanner, A. (2010). *Processing: eine Einführung in die Programmierung - Version 1.1 (Arduino kompl. überarb.)*. Surrey, BC: Selbstverl.