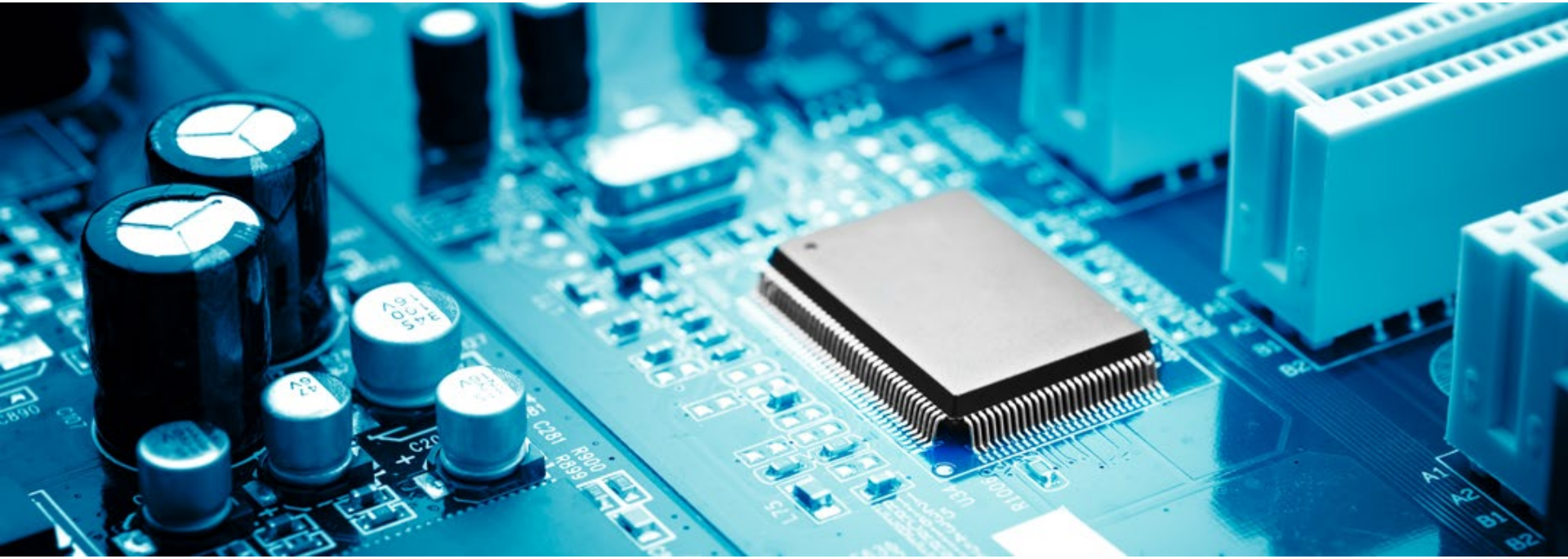


Programming

Why programming?

- You study a **Computer Science** course!
- I.e. for you, there is no “why” anymore!
- Programming is the **most-used basic skill in necessary** in almost any Computer Science field.
- Your lectures from semester 2 to 4:
  - Mathematics 2
  - Programming 2
  - Algorithms and Data Structures
  - Internet Technologies
  - Computational Logic
  - Foreign Language
  - Data bases
  - Statistics
  - Assistance Systems
  - AI Programming
  - Key Competencies 1-3
  - Natural Language Processing
  - Human Factors and Human -Machine Interaction
  - Machine learning
  - Computer vision
  - Software engineering

Why programming?

- You study a **Computer Science** course!
- I.e. for you, there is no "why" anymore!
- Programming is the **most-used basic skill in necessary** in almost any Computer Science field.
- Your lectures from semester 2 to 4:
  - Mathematics 2
  - Programming 2
  - Algorithms and Data Structures
  - Internet Technologies
  - Computational Logic
  - Foreign Language
  - Data bases
  - Statistics
  - Assistance Systems
  - AI Programming
  - Key Competencies 1-3
  - Natural Language Processing
  - Human Factors and Human -Machine Interaction
  - Machine learning
  - Computer vision
  - Software engineering

Green: You'll need programming skills. Orange: I'm pretty confident you'll need programming skills (but don't know for sure).

What is programming?

- Computers only understand 0s and 1s
- They can directly process "**machine code**" (written in **Assembly**), store things in memory, manipulate memory…
- Assembly is 1:1 translated (via code tables) into machine code
- The **machine language codes are defined by the type of chip**/processor inside the device (e.g. a CPU – central processing unit)

```
IMM    R0,    0x80
LOAD   R0,    R0
IMM    R1,    0x84
LOAD   R1,    R1
IMM    R2,    0x0
IMM    R3,    0x4
```

Assembly

```
0x 60 00 00 80
0x A4 00 00 00
0x 60 01 00 84
0x A4 01 01 00
0x 60 02 00 00
0x 60 03 00 04
```

Machine code

CPU

Higher level programming languages

- Coding means "write computer commands in a text file".
- **Assembly** is the "**lowermost**" language that humans can code in.
- **Usually "higher level" programming** languages are used for programming
- "Higher level" (gut feeling description):
  - The language **takes care of often-used tasks**
  - Tasks require **less code (i.e. lines)** to be written
  - **Security measures** are built into the language

```
#include<stdio.h>

int main() {
            printf("Hello World\n");
            return 0;
}
```

High level programming example (**C**)

?

```
0x 60 00 00 80
0x A4 00 00 00
0x 60 01 00 84
0x A4 01 01 00
0x 60 02 00 00
0x 60 03 00 04
```

Machine code (exemplary)

CPU

Compiled vs. Interpreted code

- 2 sorts of high level programming languages:
  - **Compiled**:
    - A program ("Compiler") translates the complete program code to machine code
    - The user directly runs the machine / binary code
  - **Interpreted:**
    - A program ("Interpreter) translates the program code to machine code line by line on runtime
    - The user has to have the "Translator" program
- **Program code**: A text file that follows a specific syntax that is defined by the programming language

```
#include<stdio.h>

int main() {
            printf("Hello World\n");
            return 0;
}
```

High level programming example (**C**)

Compiler or Interpreter

```
0x 60 00 00 80
0x A4 00 00 00
0x 60 01 00 84
0x A4 01 01 00
0x 60 02 00 00
0x 60 03 00 04
```

Machine code (exemplary)

CPU

- Compiled code:
  - (Usually) **Faster**
  - Harder to debug (i.e. find errors)
  - **Platform-specific**: Has to be compiled for every chipset that it should run on. Certain language extensions ("libraries") will only work on a specific operating system.
- Interpreted ("script") code:
  - **Slower** than compiled code.
  - **Easier to debug** – programs can be "stepped" through
  - The interpreter is platform specific, not the source code
  - The user must have the interpreter

What programming languages are out there?

- There are literally **hundreds (active) programming languages**. Wikipedia Timeline of Programming Languages
- Some are "general purpose", some specialized to certain types of programs.
- A few will cost you (substantial) money, most are open source

Questions:

Have you ever programmed yourself?

Which language did you use?

Of which programming languages have you heard of?

Programming languages

- Haskell
- Java
- C
- C++
- C#
- Basic
- Cobol
- Pascal
- PHP
- Visual Basic
- Python
- Ruby
- Java Script
- Perl
- Matlab
- R
- Fortran
- Assembly
- Scheme / Lisp
- Brainfuck
- Scratch
- Processing
- Max / MSP
- Pure Data
- Swift
- Rust
- ASP / Clingo
- Prolog
- …

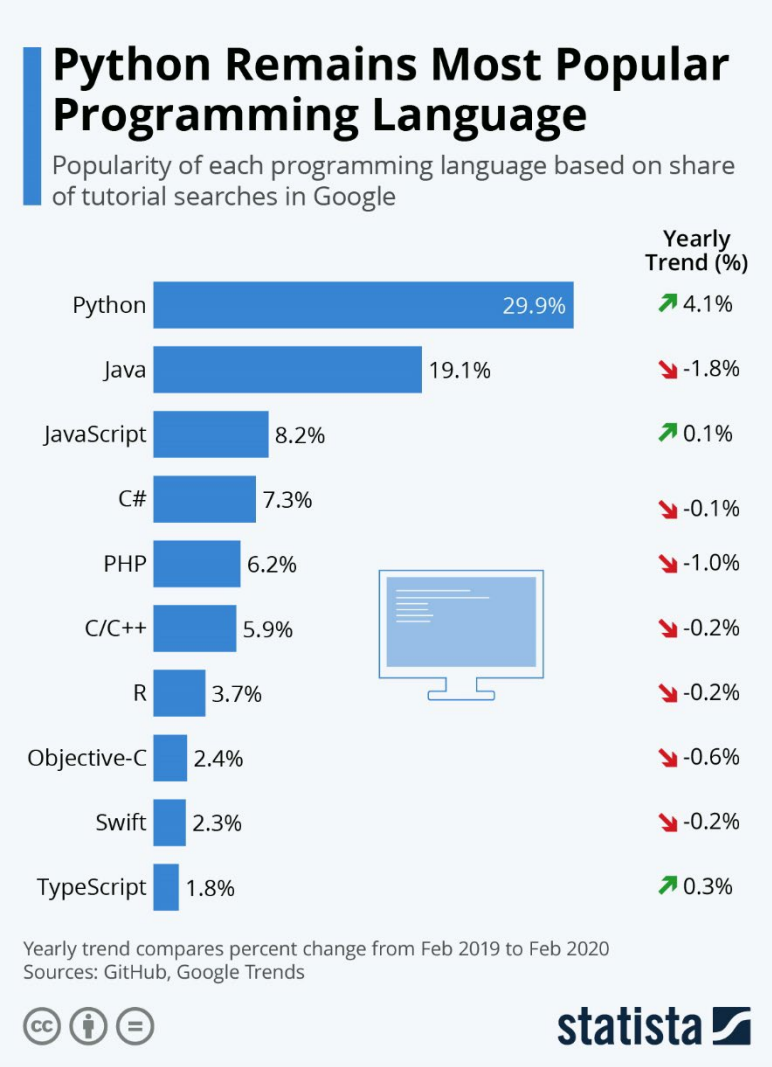
Lifespan of programming languages

- Programming languages are invented – and sometimes are disbanded from "active use" after a certain lifetime.
- A community of programmers is necessary to keep a language "alive".
- What happens often: **Language developers base a new language on a predecessor** and take "good things" over and try to improve on the rest.
- The **choice of a programming language** in commercial or government projects is often influenced if a living community is still supporting the language.
- For teaching long-livety is no factor:

  The basics are always the same!

- See O'Reilly programming languages timeline (2010-03-31, by Jeana Frost, all rights reserved).

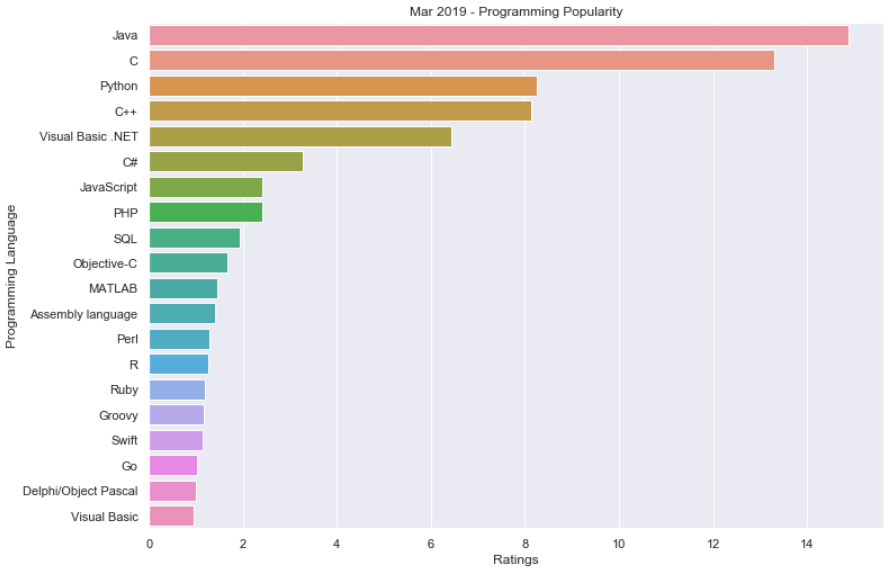
Popular languages



- Based on the measurement (online searches, code in GitLab/GitHub, TIOBE index) the most popular language is always different.
- **There is a set of languages that is** popular (better to say: **widely used**) for sure:
  Java, C, C++, C#, Python, PHP, R…
- These languages are either:
  - (Extremely) general purpose (Java, C, C++, C#)
  - Easy to learn and extendable (Python)
  - So good suited to a specific task that almost no one uses something else (PHP, R).

CC license, source: https://www.statista.com/chart/21017/most-popular-programming-languages/

Popular languages (2)

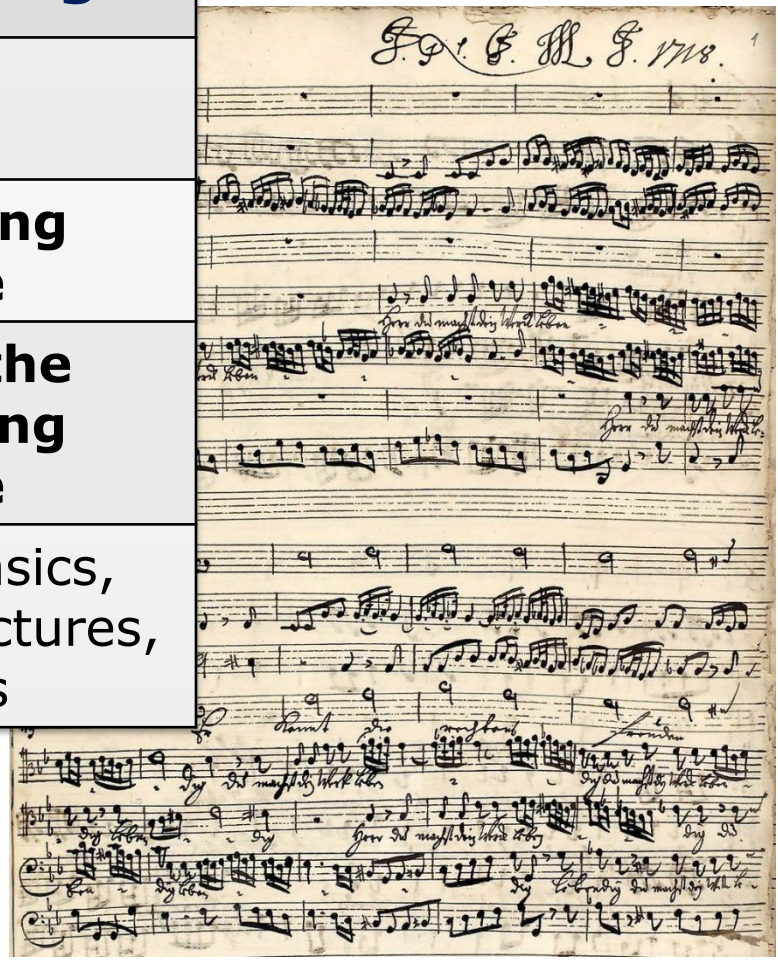


Programming popularity (based on the TIOBE Programming Community Index) by Abdul Majed Raja. MIT license. https://towardsdatascience.com/visualize-programming-language-popularity-using-tiobeindexpy-f82c5a96400d

- Based on the measurement (online searches, code in GitLab/GitHub, TIOBE index) the most popular language is always different.
- **There is a set of languages that is** popular (better to say: **widely used**) for sure:
  Java, C, C++, C#, Python, PHP, R…
- These languages are either:
  - (Extremely) general purpose (Java, C, C++, C#)
  - Easy to learn and extendable (Python)
  - So good suited to a specific task that almost no one uses something else (PHP, R).

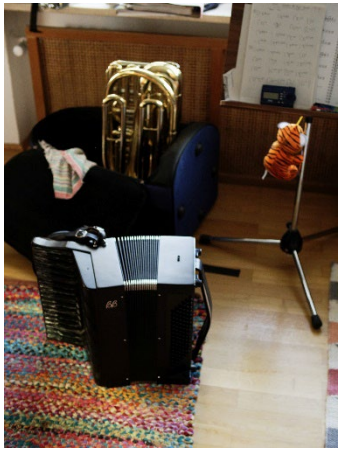
An analogy: Programming and Music

Music	Programming
Concert, recording, sheet	Program
Instrument	**Programming language**
Practicing the instrument	**Practicing the programming language**
Music theory (harmony, rhythm, instrumentation)	Programing basics, math, data structures, algorithms



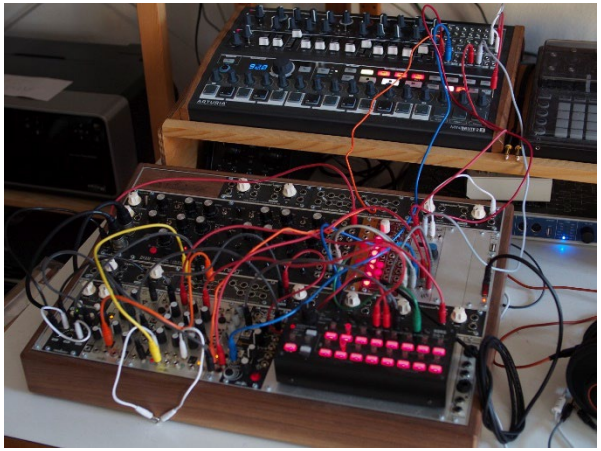
Graupner's cantata "Herr du machst dein Werk lebendig" (GWV 1113/18), UBL Darmstadt, from wikimedia, no attribution required

There are many different music instruments…



Accordion and Tuba

Ukulele



Modular and semimodular synthesizers

- You can make music with any of these instruments!
- Some are better suited to a certain "musical style or convention" than others.
- But: If pressed hard, you could write a symphony for ukuleles and drum computers!
- If you want to start out (without external force):
  Start with the one that you have the feeling you'll like.

Practicing

- Programing is like **playing an instrument:**
  - Nobody expects that you are a *Ernst Reijseger* or *Yo-Yo Ma* from day 1 on!
  - **You don’t get to be a master without practice and failure.**
  - It takes years to master an instrument.
  - **It SHOULD BE fun to play an instrument from day 1 on!**

Ernst Reijseger by Harald Bischoff (https://commons.wikimedia.org/wiki/File:Ernst_Reijseger_6606.jpg), „Ernst Reijseger 6606“, https://creativecommons.org/licenses/by-sa/3.0/legalcode

Choice of language

In general:

1. Company rules
2. **Your knowledge and preferences**
3. Suitability to the task
   (or take a general purpose language with task-specific extensions)
4. Available community support

For our specific goal**: "Learn the basics of programming in the first semester of the AI studies":**

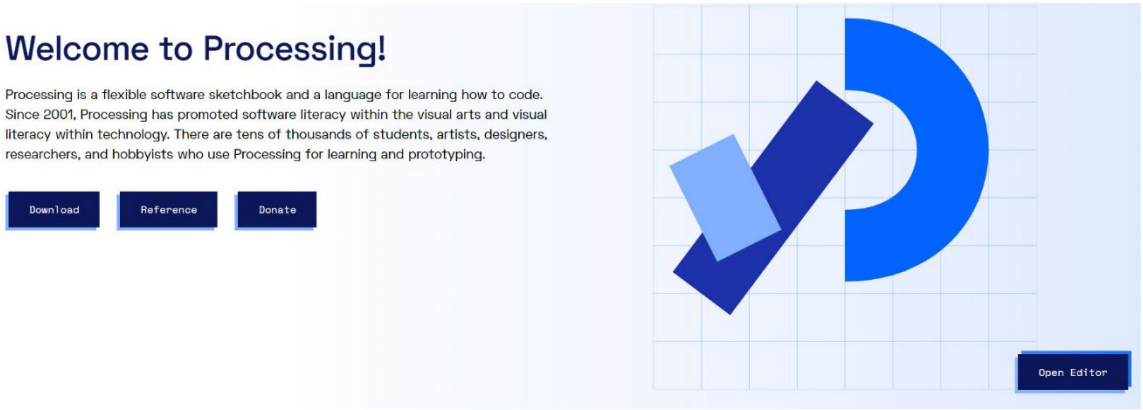
I have already chosen for you ☺

(I can not spare you the necessary failures and boring theory.)

Hopefully, you will have some fun!

How to learn programming / Our plan

- We start out with Processing: https://processing.org



Screenshot from the Processing webpage on 2022-09-23

- For the last 2 lectures of the semester, we'll switch to Java:

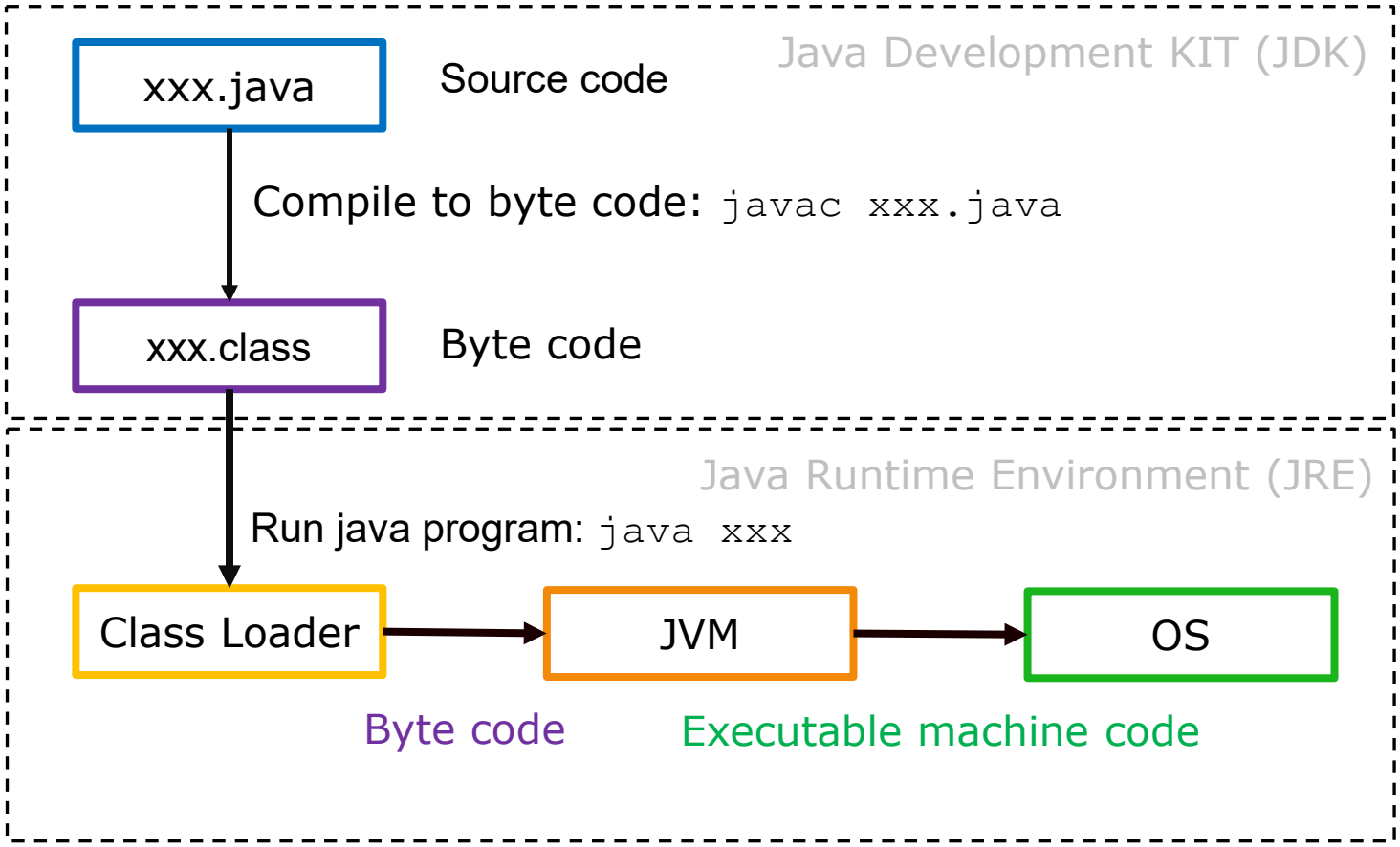
Processing

- From the processing homepage:

  "Processing is a **flexible software sketchbook** and a **language** for learning how to code. Since 2001, Processing has promoted software literacy within the **visual arts** and **visual literacy** within technology. "

- Processing is:
  - **A simple IDE** (intelligent development environment).
  - A software that translates processing code to pure Java code.
  - The respective Java libraries that allow Processing to perform some tasks (especially visuals) way, way, way more easy than in pure Java.
  - **The syntax of Processing is** (with exceptions) **Java syntax.**

Is Java/Processing code compiled or interpreted?



- Java is a little bit of both:
  The source code gets **compiled** to "byte code", which is then **interpreted** by a Java Virtual Machine (JVM).

Java: Byte code concept

- The **byte code** concept is the **most prominent feature of Java**.
- Byte code is platform independent:
  "Compile once, run everywhere!"
- Java virtual machine (JVM): Lightweight piece of software to interpret the bytecode.
- The JVM is platform-dependent.
- Amount of interpretation reduced to the minimum for platform independent byte code.
- **Slower than completely compiled code, faster than completely interpreted languages.**

Upcoming…

In lecture 1b:

- We'll write our first program in Processing.
- Simple 2D computer graphics.