First steps to python

Scientific Computing Winter 2016/2017

Lecture 15

Jürgen Fuhrmann

juergen.fuhrmann@wias-berlin.de

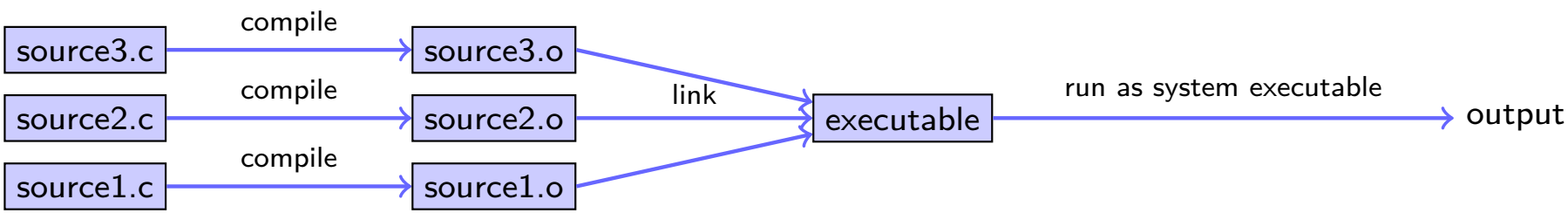
made wit pandoc

Compiled high level languages

- Algorithm description using mix of mathematical formulas and statements inspired by human language
- Translated to machine code (resp. assembler) by *compiler*

```c
#include <stdio.h>
int main (int argc, char *argv[])
{
  printf("Hello world");
}
```

- "Far away" from CPU $\Rightarrow$ the compiler is responsible for creation of optimized machine code
- Fortran, COBOL, C, Pascal, Ada, Modula2, C++, Go, Rust, Swift
- Strongly typed
- Tedious workflow: compile - link - run

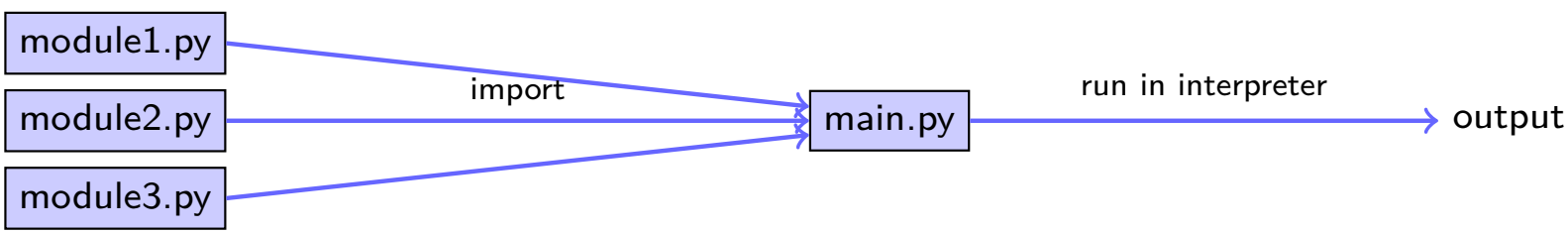


High level scripting languages

- Algorithm description using mix of mathematical formulas and statements inspired by human language
- Need intepreter in order to be executed

```
print("Hello world")
```

- Very far away from CPU $\Rightarrow$ usually significantly slower compared to compiled languages
- Matlab, Python, Lua, perl, R, Java, javascript
- Less strict type checking, often simple syntax, powerful introspection capabilities
- Immediate workflow: "just run"
  - in fact: first compiled to *bytecode* which can be interpreted more efficiently



Python

- Developed since 1989, led by Guido van Rossum
- Can be seen as "open source" matlab
- Main advantage: huge ecosystem of packages for scientific computing
- Some use cases:
  - matlab replacement
  - glue language for different tools
  - system independent implementation of tools (e.g. mercurial)
  - driver language for software written in C/C++
    - quickly change parameters without recompiling etc.
    - make use of plotting capabilities
- Documentation: https://docs.python.org
  - current versions around: 2.7, 3.x
  - most python3 code works with 2.7
- Tutorial: `https://docs.python.org/3/tutorial/`

Numpy / Scipy /matplotlib

- numpy: add-on of an efficient array class for numerical computations, written in C
- Python lists would be too slow
- Interfacing to lapack etc. need dense arrays
- scipy: Scientific computation package with LAPACK etc.
- matplotlib: data plotting + visualization

Python and C++

- Architecture of python
  - Interpreter
  - Application programming interface (API) for interaction of C/C++ with python interpreter
    - register wrapper function with name
    - wrapper function knows how to fetch parameters/return values from/to python interpreter
    - wrapper function calls C++ function
  - Wrapper code with code to be wrapped linked to a "shared object" (UNIX), "dylib" (Mac), "DLL" (Windows)
  - Import of wrapper code makes it available in python
- Automatic tools for accessing API

SWIG

- Several possibilities
  - Cython (python dialect with possible inclusion of C)
  - pybind11 (C++11 classes for wrapping python)
  - **SWIG** ("classical tool" for wrapping interpreter)
- Simplified Wrapper and Interface Generator:
  - Tool to automatically create wrapper code from C++ style description
  - Create wrapper code in C++ which is linked together with library to be wrapped