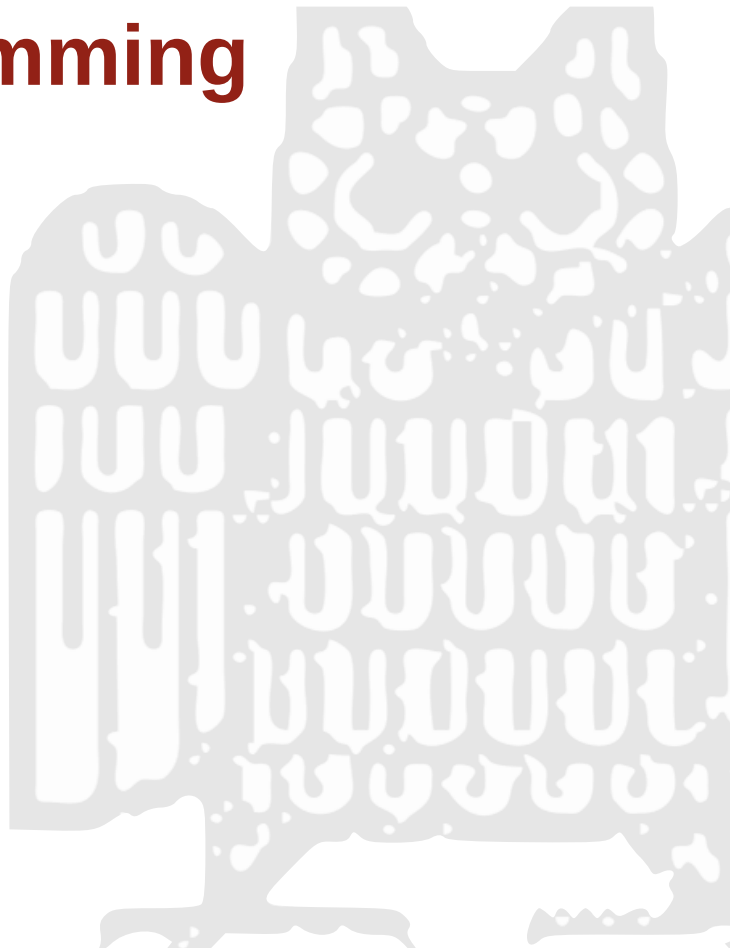
Introduction to Python Programming

3 – Algorithms

Josef van Genabith (Stefan Thater)
Dept. of Language Science & Technology
Universität des Saarlandes

WS 2022/23



What is Programming?

- **Programming** = give (sequence of) step-by-step instructions to the computer to solve a task/problem
- Everyday example tasks/problems we solve step-by-step:
  - Wash clothes using washing machine.
  - …

What is Programming?

- **Programming** = give (sequence of) instructions to computer to solve a problem
- Everyday example tasks/problems we solve step-by-step:
  - Wash clothes using washing machine.
  - Bake a cake.
  - Cook a dish.
  - Find (an approximation of) the square root of a number
  - Find the largest number in a list of numbers.
  - Sort a list of numbers
  - Translate text from one language to another
  - …

What is Programming?

- **Programming** = give (sequence of) instructions to the computer in a language the computer understands to solve a problem
- **Algorithm** = abstract sequence of instructions to solve a problem, not necessarily on a computer (mathematics, a recipe …)

What is Programming?

- **Algorithm** = abstract, detailed computing instructions that solve the task/problem (“recipe”)
- Example: *Wash clothes using the washing machine*
  1. Load the laundry
  2. Add detergent and additive
  3. Switch on the machine
  4. If clothes are wool: select wool program; otherwise select normal program
  5. Start the program
  6. Wait until done
  7. Remove clothes

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the task/problem (“recipe”)
- Example: *print hello* (this is the **task description**)

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

algorithm

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the task/problem (“recipe”)
- Example: *print hello* (this is the **task description**)

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

algorithm

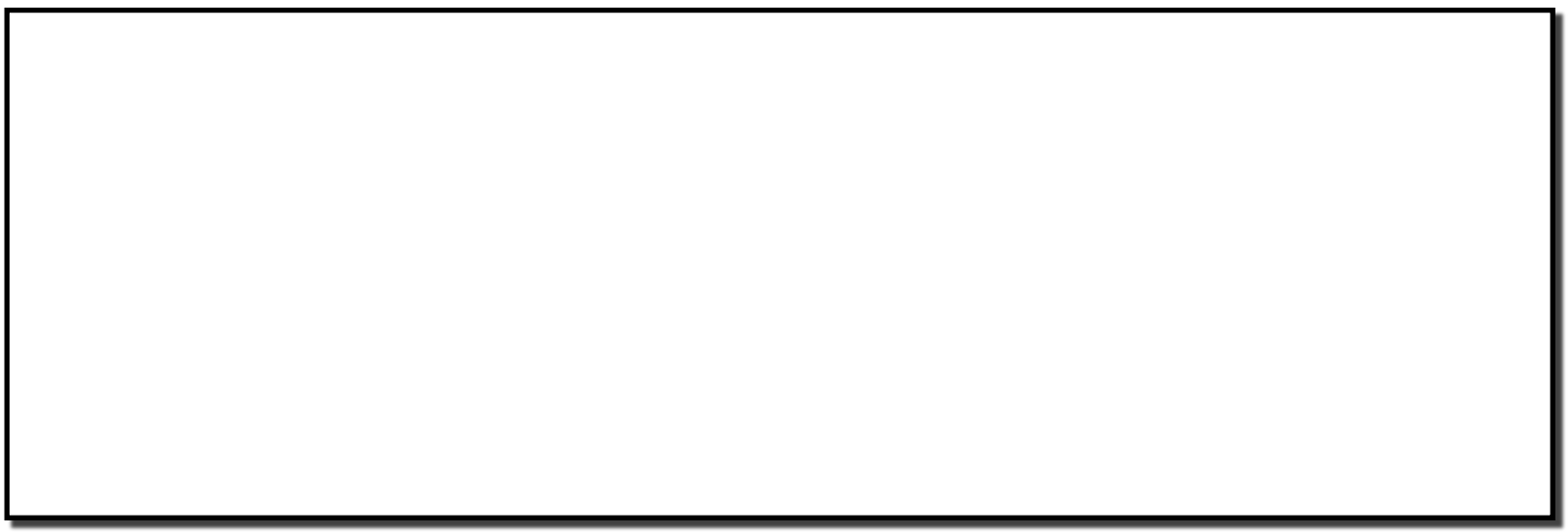
- **Programming** = translating **task description** into **algorithm** that can run on a computer.

What is Programming?

- **Programming** = **translating** [task description] into [algorithm that can run on a computer].
- Two languages involved
  - Language of task description (text, a mathematical description, an informal idea in your head, …)
  - Language of the computer: a programming language! Python, Java, C, C++, R, Prolog, Lisp, Haskell, ….
- Programming: task description ⇒ a programm/an algorithm that can run on computer

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *print hello 3 times* (task description)



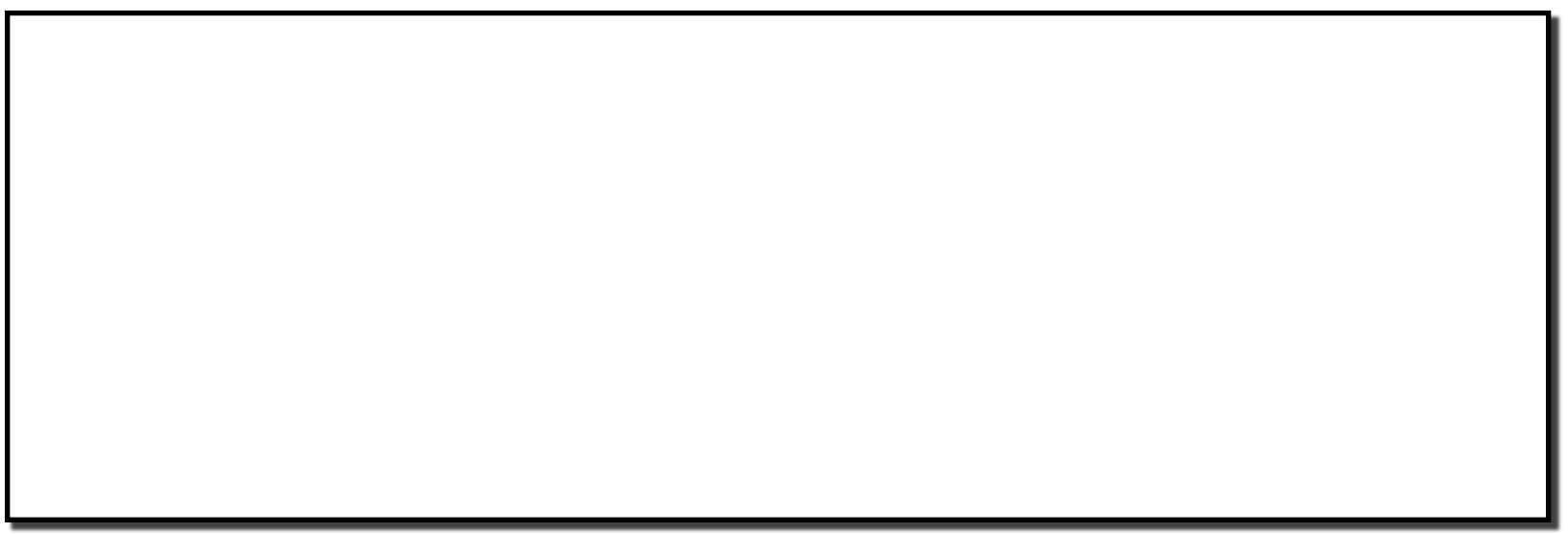
What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *print hello 3 times*

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

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *print hello 5 times*



What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *print hello 5 times*

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

What is Programming?

- **Imperative Programming, Procedural Programming**
- You **tell** the computer what to do **step-by-step**
- You **instruct** the computer what to do **step-by-step**
- The computer **executes** what you tell it to do **step-by-step**

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

What is Programming?

- **Imperative Programming, Procedural Programming**
- You **tell** the computer what to do **step-by-step**
- You **instruct** the computer what to do **step-by-step**
- The computer **executes** what you tell it to do **step-by-step**

```
i=1
while i < 6:
   print("Hello")
   i = i + 1
```

What is Programming?

- **Imperative Programming, Procedural Programming**
- You **tell** the computer what to do **step-by-step**
- You **instruct** the computer what to do **step-by-step**
- The computer **executes** what you tell it to do **step-by-step**

```
i=0
while i < 5:
   print("Hello")
   i = i + 1
```

What is Programming?

- **Imperative Programming, Procedural Programming**
- You **tell** the computer what to do **step-by-step**
- You **instruct** the computer what to do **step-by-step**
- The computer **executes** what you tell it to do **step-by-step**

```
i=0
while i < 5:
    print("Hello")
    # i = i + 1
```

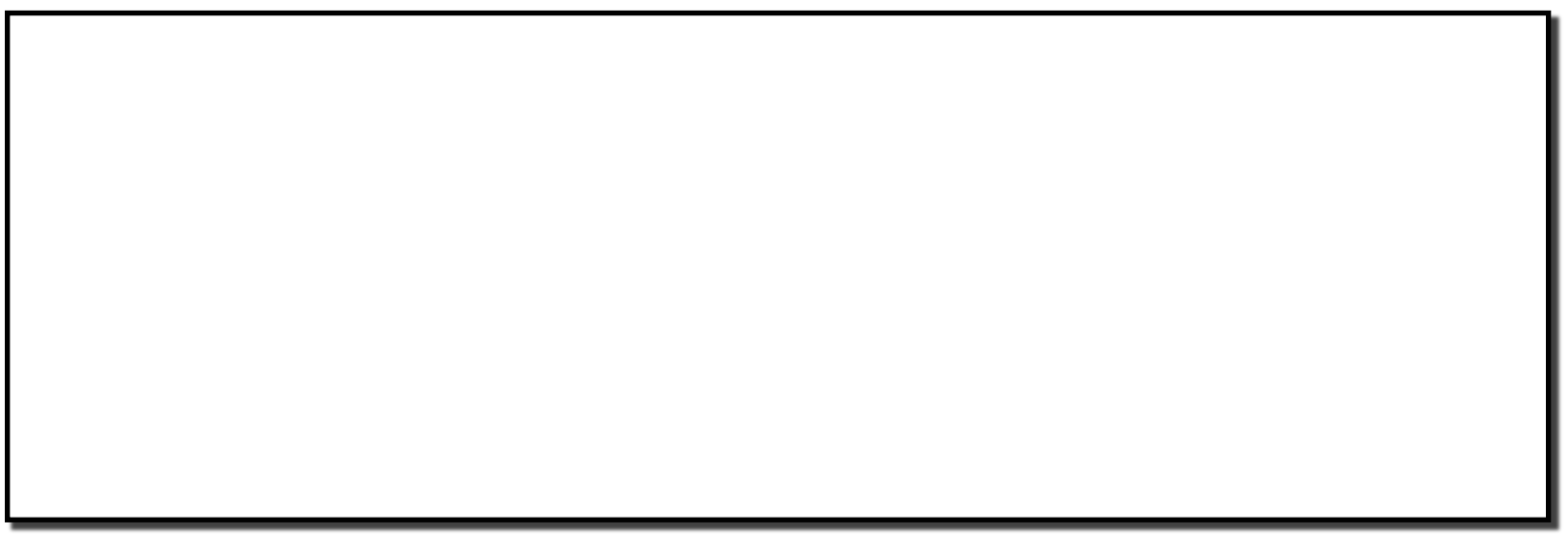
What is Programming?

- **Imperative Programming, Procedural Programming**
- You **tell** the computer what to do **step-by-step**
- You **instruct** the computer what to do **step-by-step**
- The computer **executes** what you tell it to do **step-by-step**

```
for i in range(5):
   print("Hello")
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *print hi 5 times*



What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *print hi 5 times*

```
print("hi")
print("hi")
print("hi")
print("hi")
print("hi")
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *you have two friends Mo and Jo*, *print hi Mo and print hi Jo*

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *you have two friends Mo and Jo*, *print hi Mo and print hi Jo*

```
print("hi Mo")
print("hi Jo")
```

What is Programming? Many Solutions …

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *you have two friends Mo and Jo*, *print hi Mo and print hi Jo*

```
print("hi" + "Mo")
print("hi" + "Jo")
```

- Like in human translation: **not just one** solution …! Many …!
- Some may be better than others …

What is Programming? Many Solutions …

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *you have two friends Mo and Jo*, *print hi Mo and print hi Jo*

```
print("hi " + "Mo")
print("hi " + "Jo")
```

What is Programming? Many Solutions …

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Example: *you have two friends Mo and Jo*, *print hi Mo and print hi Jo*

```
for name in ["Mo", "Jo"]:
   print("hi " + name)
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem/task (“recipe”)
- Exercise: *for each of your five friends Jill, John, Jane, Tarzan, and Peggy*, *print hi Friend*

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *for each of your five friends Jill, John, Jane, Tarzan, and Peggy*, *print hi friend*

```
for name in ["Jill", "John", "Jane", "Tarzan", "Peggy"]:
   print("hi " + name)
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *for each of your five friends Jill, John, Jane, Tarzan, and Peggy*, *print hi friend*

```
for friend in ["Jill", "John", "Jane", "Tarzan", "Peggy"]:
   print("hi " + friend)
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *for each of your five friends Jill, John, Jane, Tarzan, and Peggy*, *print hi Friend*

```
for x in ["Jill", "John", "Jane", "Tarzan", "Peggy"]:
   print("hi " + x)
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *for each of your five friends Jill, John, Jane, Tarzan, and Peggy*, *print hi Friend*

```
for friend in ["Jill", "John", "Jane", "Tarzan", "Peggy"]:
   print("hi " + freind)
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *for each of your five friends Jill, John, Jane, Tarzan, and Peggy*, *print hi Friend*

```
for x in ["Jill", "John", "Jane", "Tarzan", "Peggy"]:
   print("hi " + y)
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *input a name on the command line*, *print hi name*

```
print("Please enter a name: ")
y = input()
name = str(y)
print("Hi " + name)
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *input a name on the command line*, *print hi name*

```
print("Please enter a name: ")
y = input()
print("Hi " + y)
```

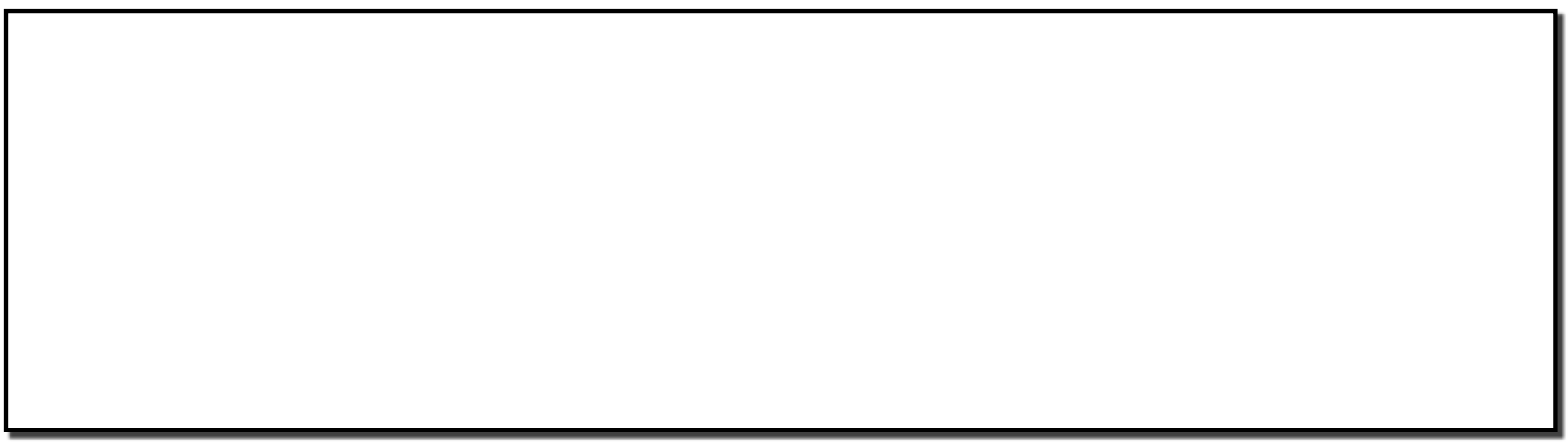
What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *input a name on the command line*, *print hi name*

```
print("Please enter a name: ")
name = input()
print("Hi " + name)
```

What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *input an integer on the command line*, devide the integer by two, *print the result*



What is Programming?

- **Algorithm** = abstract, detailed computing instruction that solves the problem (“recipe”)
- Exercise: *input an integer on the command line*, devide the integer by two, *print the result*

```
print("Please enter an integer: ")
x = input()
y = int(x)
z = y/2
print("The result of ", x, "divided by 2 is ", z)
```

What is Programming?

- **Algorithm** = abstract, detailed computing instructions that solve the problem (“recipe”)
- Example: *Compute (an approximation of)* $\sqrt{x}$

What is Programming?

- **Algorithm** = abstract, detailed computing instructions that solve the problem (“recipe”)
- Example: *Compute (an approximation of)* $\sqrt{x}$
  1. Make a guess g, say, $g = x$
  2. Improve the guess by averaging over g and x / g
  3. Keep improving the guess until it’s good enough

- Note: here we have a mathematical description (already kind of an algorithm) that we need to translate into a computer programme

What is Programming?

- **Program** = realization of the algorithm in a specific programming language

```
def sqrt(x):
  g = x
  while abs((g * g) - x) > 0.0001:
    g = (g + (x / g)) / 2
  return g
```

Properties of Algorithms

- An algorithm can be executed with different inputs
  - Washing: should wash different laundry correctly: wool, normal, towels, curtains, …
  - Maximum of a list: should find the maximum of any given input list: [1, 2, 5, 7, 9]; [5, 2, 29, 0,11,5]; []; [5]; …
- Algorithms should terminate with an output

How do we Solve Complex Tasks/Problems?

Source:
wikipedia

How do we Solve Complex Tasks/Problems?

Divide and Conquer

Source:
wikimedia

Problems & Subproblems

- To solve a problem/task: decompose/break it down into subproblems.
- Then solve each subproblem: that solves the big problem …
- Example
  - Problem: Find greatest number in list.
  - Subproblem: Find maximum of two numbers.
- Algorithms are a way of breaking down a problem into a number of subproblems, and solving the subproblems, and thereby solving the overall problem.
- Can you think of examples for problems & subproblems?

Algorithms: Summary

- “Recipe” for solving a problem.
- Breaking problem down into subproblems.
- Should work for all inputs of the problem.
- Must terminate in a finite number of steps.
- Granularity of the steps:
  - Recipe must be defined clearly. Depends on audience / programming language it is designed for / …
- There may be more than one algorithm per problem.
- Efficiency = measured in time and / or memory usage (often a trade-off between the two).

Programming Languages

- CPU only understands machine language instructions
  - 0110101101… ⇒ inconvenient for humans
- Programming languages
  - hide complexity of machine language
  - provide more abstract constructs (easier to understand for humans!)
  - make programmer/human-machine interaction efficient
- Different programming languages
  - are designed for different use-cases
  - support different programming styles: procedural / functional / logic / object-oriented
  - have different advantages and disadvantages

Python

- Python is a bit like a mix of Maths and English …

```
print("Please enter an integer: ")
y = input()
x = int(y)
z = x/2
print("The result of ", x, "divided by 2 is ", z)
```

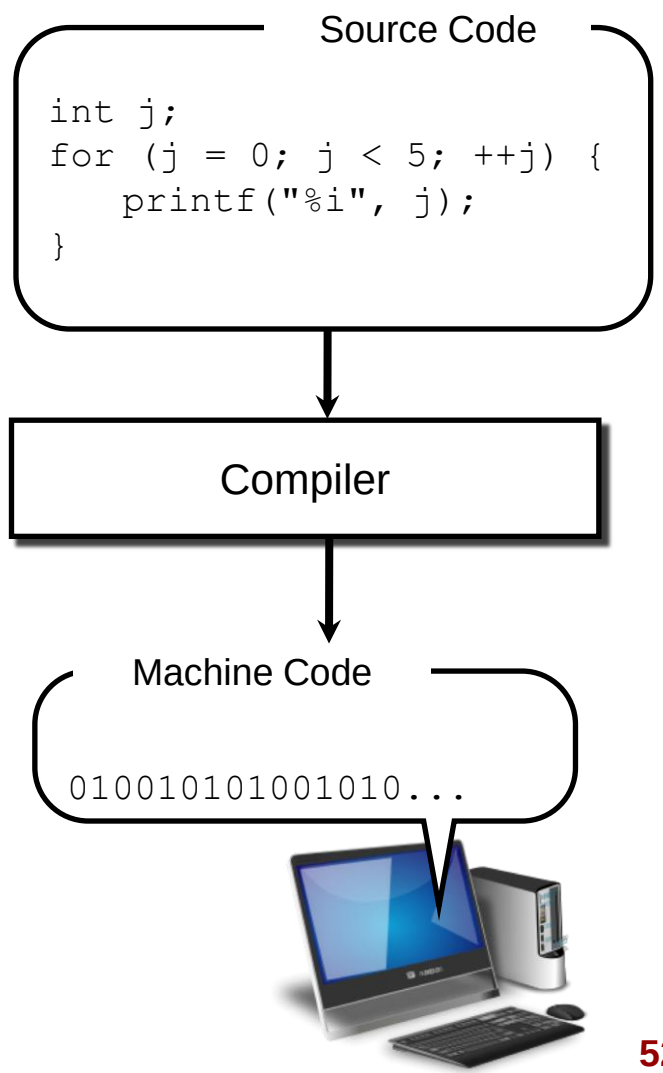
- Good bit easier to write/understand than 0110101101…

Python

- Procedural / Object-oriented programming language.
  - also (partially) supports functional programming
  - a good “scripting language”
- History
  - Successor of “teaching language” ABC
  - Development started in late 80ies
  - For ambitious users without programming skills
  - Focus on easy file handling

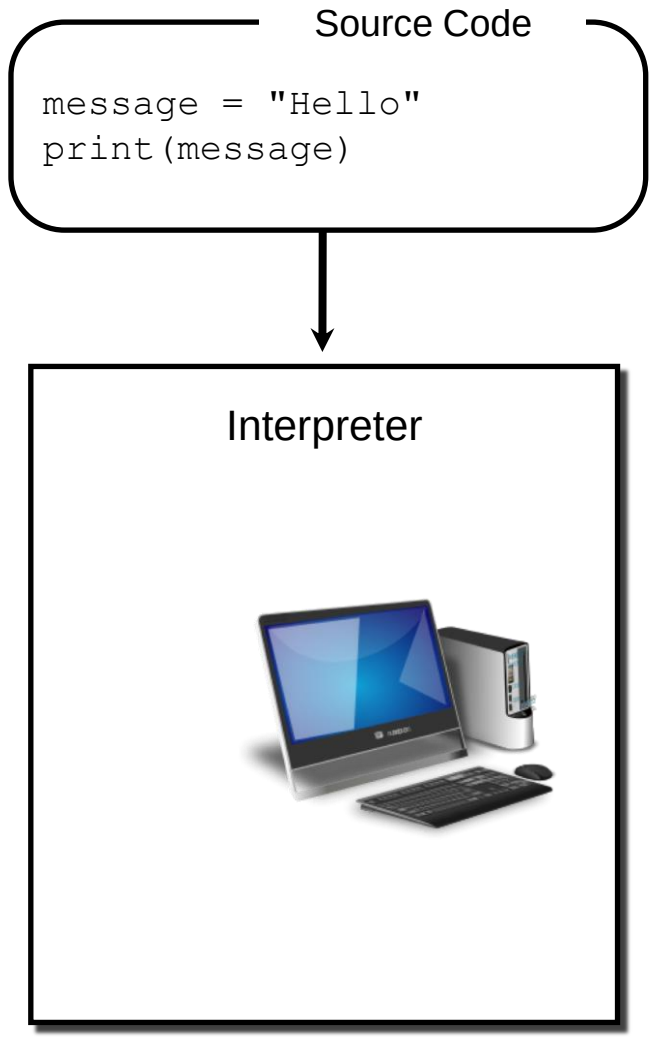
Compilation

- Compiler = a program that translates source code into machine code during compile time
- CPU executes this machine code during runtime
- Example: C, C++ is a compiled language
- Advantage: once compiled, programs run very fast
- Machine code is specific for a platform (Linux / Windows / …)



Interpretation

- **Interpreter** = program that executes the source code per command (line by line).
- Enables easy step-by-step interaction between system and programmer.
- Python is an interpreted language.
- The Python interpreter is specific for the platform.
- Your Python code can run on any platform.



Advantages of Python

- Simple syntax
- Very flexible – little is forbidden, much is convention.
- Rich standard library (“batteries included”)
- Easy file handling; full unicode support
- Handles arbitrarily large integers
- Convenient support for regular expressions
- Natural Language Toolkit + many other frameworks
- Not a “lean“ language …

Some Recommended Reading

- *The Semicolon Wars* by Brian Hayes
  - Every programmer knows there is one true programming language. A new one every week.
  - Jeder Programmierer weiß, dass es nur eine einzig wahre Computersprache gibt. Jede Woche eine neue.
- http://www.americanscientist.org/issues/pub/the-semicolon-wars
- Deutsche Version: Brian Hayes: Der Strichpunkt-Krieg (Spektrum Wissenschaft).

Literature

- Mark Lutz: Learning Python (Animal Guide), 4th edition, 2009, O'Reilly Standard Python Reference Book, *comprehensive*
- Michael Dawson: Python Programming for the absolute beginner, 3rd edition, 2010, Course Technology / Cengage Learning; *Excellent for absolute beginners, good explanations, as fun to read as a programming book can be*
- Steven Bird, Ewan Klein, Edward Loper: Natural Language Processing with Python, O'Reilly Media 2009
  Explains the toolkit NLTK
  available online: http://www.nltk.org/book

Exercise: a Division and Modulo Calculator

Please enter the numerator (needs to be an integer):

8

Please enter the denominator (needs to be an integer):

3

Result of the division operation: 2.6666666666665

Result of the modulo operation: 2

$$fraction = \frac{numerator}{denominator}$$