PROGRAMMING IN PYTHON I

Basics of Programming

Andreas Schörgenhumer

Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Copyright Statement

GENERAL INFORMATION ON PROGRAMMING

General Information on Programming

- Question: What is programming about?

General Information on Programming

- Question: What is programming about?
- Answer: Letting a machine solve a problem for you!

General Information on Programming

- Question: What is programming about?
- Answer: Letting a machine solve a problem for you!
  1. Identify the problem and its specification

General Information on Programming

- Question: What is programming about?
- Answer: Letting a machine solve a problem for you!
  1. Identify the problem and its specification
  2. Structure your thoughts on how to solve a problem, i.e., come up with an **algorithm**[1]

[1] An algorithm is a step-wise, rigorous procedure to solve a problem.

General Information on Programming

- Question: What is programming about?
- Answer: Letting a machine solve a problem for you!
  1. Identify the problem and its specification
  2. Structure your thoughts on how to solve a problem, i.e., come up with an **algorithm**[1]
  3. Formulate the algorithm as **program** code[2]

[1] An algorithm is a step-wise, rigorous procedure to solve a problem.

[2] A program is the formulation of an algorithm in a programming language.

General Information on Programming

- Question: What is programming about?
- Answer: Letting a machine solve a problem for you!
  1. Identify the problem and its specification
  2. Structure your thoughts on how to solve a problem, i.e., come up with an **algorithm**[1]
  3. Formulate the algorithm as **program** code[2]
  4. Let the machine execute your code

[1] An algorithm is a step-wise, rigorous procedure to solve a problem.

[2] A program is the formulation of an algorithm in a programming language.

General Information on Programming

- Question: What is programming about?
- Answer: Letting a machine solve a problem for you!
  1. Identify the problem and its specification
  2. Structure your thoughts on how to solve a problem, i.e., come up with an **algorithm**[1]
  3. Formulate the algorithm as **program** code[2]
  4. Let the machine execute your code
  5. Look what went wrong and go back to previous steps

[1] An algorithm is a step-wise, rigorous procedure to solve a problem.

[2] A program is the formulation of an algorithm in a programming language.

General Information on Programming

- Question: What is programming about?
- Answer: Letting a machine solve a problem for you!
  1. Identify the problem and its specification
  2. Structure your thoughts on how to solve a problem, i.e., come up with an **algorithm**[1]
  3. Formulate the algorithm as **program** code[2]
  4. Let the machine execute your code
  5. Look what went wrong and go back to previous steps

  → **If the machine does it, you don't have to do it! :)**

[1] An algorithm is a step-wise, rigorous procedure to solve a problem.

[2] A program is the formulation of an algorithm in a programming language.

Example

- Problem: Find the maximum element in a non-empty sequence of numbers.
- Possible algorithm:[3]

[3] There are many ways to specify an algorithm: prose, pseudo-code, flowcharts, ...

Example

- Problem: Find the maximum element in a non-empty sequence of numbers.
- Possible algorithm:[3]
  1. Create an empty variable `max`

[3] There are many ways to specify an algorithm: prose, pseudo-code, flowcharts, ...

Example

- Problem: Find the maximum element in a non-empty sequence of numbers.
- Possible algorithm:[3]
  1. Create an empty variable `max`
  2. For each `number` in the given sequence:

[3] There are many ways to specify an algorithm: prose, pseudo-code, flowcharts, ...

Example

- Problem: Find the maximum element in a non-empty sequence of numbers.
- Possible algorithm:[3]
  1. Create an empty variable `max`
  2. For each `number` in the given sequence:
     - 2.1 Check if `max` is empty or `number` > `max`

[3] There are many ways to specify an algorithm: prose, pseudo-code, flowcharts, ...

Example

- Problem: Find the maximum element in a non-empty sequence of numbers.
- Possible algorithm:[3]
  1. Create an empty variable `max`
  2. For each `number` in the given sequence:
     - 2.1 Check if `max` is empty or `number` > `max`
     - 2.2 If yes, set `max` to `number`

[3] There are many ways to specify an algorithm: prose, pseudo-code, flowcharts, ...

Example

- Problem: Find the maximum element in a non-empty sequence of numbers.
- Possible algorithm:[3]
  1. Create an empty variable `max`
  2. For each `number` in the given sequence:
     - 2.1 Check if `max` is empty or `number` > `max`
     - 2.2 If yes, set `max` to `number`
  3. Return `max`
- Write a program for this algorithm in the programming language of choice.

[3] There are many ways to specify an algorithm: prose, pseudo-code, flowcharts, ...

Typical Problems in Programming

- 2 main sources of problems:

Typical Problems in Programming

- 2 main sources of problems:

1. **Syntax errors**, wrong usage of code tools, insufficient knowledge about programming language
   - Syntax can be learned passively during lectures
   - Learned on-the-fly with exercises and practice

Typical Problems in Programming

- 2 main sources of problems:

1. **Syntax errors**, wrong usage of code tools, insufficient knowledge about programming language
   - Syntax can be learned passively during lectures
   - Learned on-the-fly with exercises and practice
2. **Semantic errors**, mistakes in thinking through/formulating the solution
   - Generally wrong solutions (mistakes in thinking or understanding of task)
   - Missing consideration of some use-cases

Typical Problems in Programming

- 2 main sources of problems:

1. **Syntax errors**, wrong usage of code tools, insufficient knowledge about programming language
   - Syntax can be learned passively during lectures
   - Learned on-the-fly with exercises and practice
2. **Semantic errors**, mistakes in thinking through/formulating the solution
   - Generally wrong solutions (mistakes in thinking or understanding of task)
   - Missing consideration of some use-cases
   - The machine will do what you tell it to do

Typical Problems in Programming

- 2 main sources of problems:

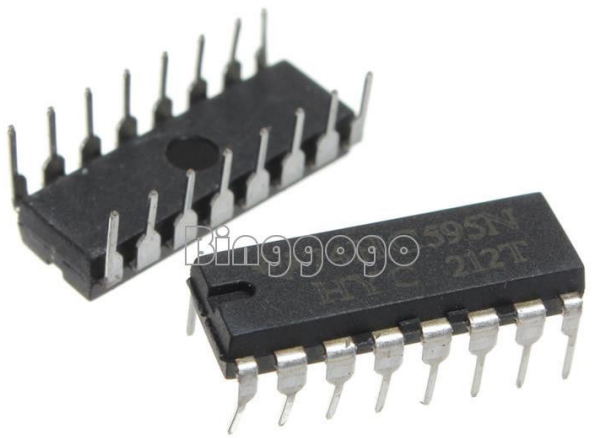
1. **Syntax errors**, wrong usage of code tools, insufficient knowledge about programming language
   - Syntax can be learned passively during lectures
   - Learned on-the-fly with exercises and practice
2. **Semantic errors**, mistakes in thinking through/formulating the solution
   - Generally wrong solutions (mistakes in thinking or understanding of task)
   - Missing consideration of some use-cases
   - The machine will do what you tell it to do

- You will be directly confronted with your own errors
- Errors in code are also referred to as **bugs**

BITS AND BYTES

Bits and Bytes (1)

- Data on computer (usually) stored in **bits**
- **Bit:** element with 2 states (True/False, 0/1, . . .)
  - E.g., transistors, pneumatic elements, magnetic stripes, . . .
- Registers (small storage units) usually able to hold 8 bits (or multitudes of 8 bits)
  - 8 bits are referred to as **byte**



[Image: 10/20/50/100PCS 2.0-6.0 V SN74HC595N 74HC595 8-Bit Shift Register DIP-16 IC, Binggogo]

Bits and Bytes (2)

- We use bits to store all kinds of data
  - Values, text, programs, images, audio, ...
- We **encode** our data in bit patterns and later **decode** it to retrieve the meaning of the data $\rightarrow$ **data types**
- Example:
  1. Right-click on a (small) image, audio, PDF, or any other non-text file
  2. Open it with a text editor (`notepad`, `texteditor`, `gedit`)
  3. Enjoy the bit pattern of the file interpreted as pure text ;)