ETH zürich

Departement Informatik

Programming and Problem-Solving

Introduction to the Course

Dennis Komm

Spring 2021 – February 25, 2021

Welcome to the Course

Material

Lecture website

`https://lec.inf.ethz.ch/ppl`

Moodle Course

`https://moodle-app2.let.ethz.ch/course/view.php?id=14883`

The Team

Lecturer	Dennis Komm
Assistants	Manuela Fischer Jonas Hein David Sommer Cathirn Elich Lea Fritschi Sarah Kamp Safira Piasko Sara Steiner

Appointments

Lecture	Thursday, 16:15 – 18:00
Exercises	Monday, 13:15 – 15:00 Thursday, 10:15 – 12:00
Exam	End of the semester

Goal of Today's Lecture

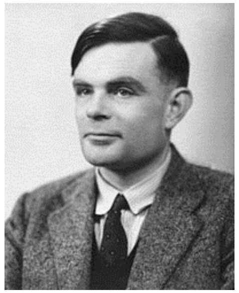
- General information about the lecture
- The projects, using **[code]expert**
- Introduction to Algorithms
- The first Python program

Introduction to the Course

Computers and Algorithms

Computer – Concept

- What does a computer have to be able to do to compute?
- Does it have to be able to multiply?
- Isn't it sufficient to be able to add?



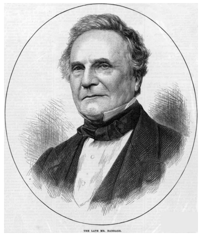
Alan Turing [Wikimedia]

Turing Machine [Alan Turing, 1936]

- Finite number of states
- Memory consisting of arbitrarily many cells
- Pointer to current cell
- Pointer can change cell's content and move left or right

Computer – Implementation

- **Analytical Engine** – Charles Babbage (1837)
- **Z1** – Konrad Zuse (1938)
- **ENIAC** – John von Neumann (1945)



Charles Babbage [Wikimedia]

Konrad Zuse [Wikimedia]

John von Neumann [Wikimedia]

Algorithm: Central Notion of Computer Science

Algorithm

- Method for step-by-step solution of a problem
- Execution does not require intellect, only accuracy
- after **Muhammad al-Chwarizmi**; author of a arabic math book (around 825)

"'Dixit algorizmi. . . "' Latin translation [Wikimedia]

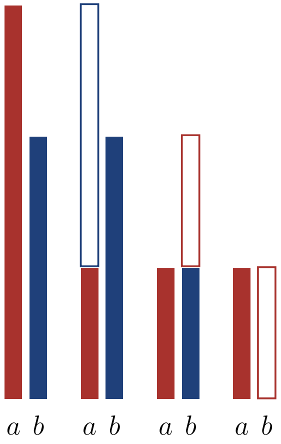
"The Oldest (Known) Non-Trivial Algorithm"

Euclid's Algorithm

from Euclid's *Elements*, 300 BC

- Input: integers $a > 0, b > 0$
- Output: gcd of a and b

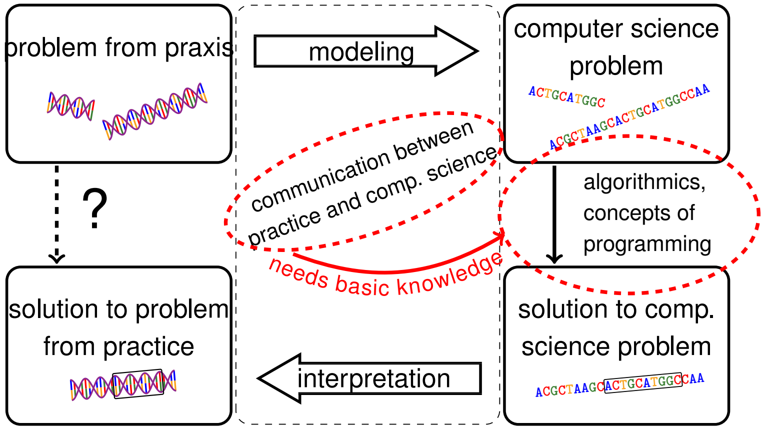
```
Input: $a$ and $b$
while $b$ != 0:
    if $a$ > $b$:
        $a = a - b$
    else:
        $b = b - a$
Output: $a$
```



Introduction to the Course

Goals

1. Computer Science in the Natural Sciences



problem from praxis

modeling

computer science problem

ACTGCATGGC

ACGCTAAGCACTGCATGGCCAA

?

communication between practice and comp. science

needs basic knowledge

algorithmics, concepts of programming

solution to problem from practice

interpretation

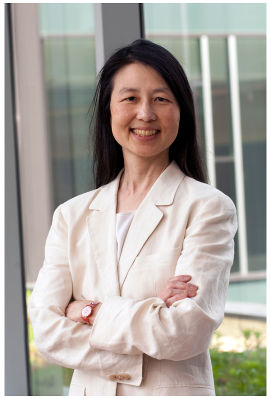
solution to comp. science problem

ACGCTAAGCACTGCATGGCCAA

2. Computational Thinking

- Systematic solving of given problems
- This implies creativity, abstraction skills etc.
- Formulation of solution as algorithm
- Solution can be “understood” by a computer

2. Computational Thinking



Jeannette Wing

"“Computational thinking is a way humans solve problems; it is not trying to get humans to think like computers. Computers are dull and boring; humans are clever and imaginative. We humans make computers exciting.”"

3. Algorithms Design Techniques

Most practically relevant problems have easy solutions

- Easy to implement
- Are based on trying out possibly many possibilities (“solution candidates”)
- This means impractically large time to spend

Many problem allow for “faster” solutions

- Needs a little more skill
- Different techniques: **greedy algorithms**, **divide and conquer**, **dynamic programming** etc.

Introduction to the Course

Projects

Projects

During the semester, you work on a few small projects

- The project tasks will be published via [code]expert

  `https://expert.ethz.ch`

- You work on the tasks on your own
- The exercise hours are meant for answering your questions
- Presentation of the solutions via Zoom

Projects

The projects will be presented in the exercise hours

- Presentation and discussion with assistants
- Teams of 2 students each
- Grading by assistants, feedback by students
- **Presentation is mandatory**
- **but without effect on the grade**
- [code]expert allows you to test your solution before handing it in

Introduction to Python

Programming Tools

- **Editor:** Program to modify, edit and store Python program texts
- **Compiler:** Program to translate a program text into machine language (intermediate code, respectively)
- **Computer:** Machine to execute machine language programs
- **Operating System:** Program to organize all procedures such as file handling, editing, compiling, and program execution

English vs. Programming Language

English

"Science is what we understand well enough
to explain to a computer.
Art is everything else we do."

DONALD KNUTH

Python

```
# computation
b = a * a    # b = a**2
b = b * b    # b = a**4
```

Syntax and Semantics

- Like our language, programs have to be formed according to certain rules
  - **Syntax:** Connection rules for elementary symbols (characters)
  - **Semantics:** Interpretation rules for connected symbols
- Corresponding rules for a computer program are simpler, but also more strict because computers are relatively stupid

Kinds of Errors Illustrated with English Language

The car drove too fast.	Syntactically and semantically correct
Thecar drove too fsat.	Syntax error: word building
Red the car is.	Syntax error: word order
I find inspiration in cooking my dog and my cat	Syntax error: missing punctuation marks
She is not tall and red-haired.	Syntactically correct, but ambiguous [no analogon]
I own an red car.	Syntactically correct, but gramatically and semantically wrong: wrong article [type error]
The bike gallops fast.	Syntactically and gramatically correct, but semantically wrong [run-time error]
We saw her duck.	Syntactically and sematically correct, but ambiguous [no analogon]

Introduction to Python

Used Software

Used Software

- There are numerous Python development environments (IDEs)
- These contain an editor and several tools
- We use **[code]expert**

```
https://expert.ethz.ch/enroll/SS21/ppl
```

- Also recommended (offline): **PyCharm Education**

```
https://www.jetbrains.com/pycharm-educational/download/
```

- Download the Community Edition

Introduction to Python

A First Python Program

A First Python Program

```
print("This is a Python program")

x = 20
print("The value of x is", x)
y = x * x    # y is the square of x
print("The value of y is", y)
z = y * y    # z is the square of y
print("The value of z is", x * x * x * x)
```

Behavior of a Program

At compile time

- Program accepted by the compiler (syntactically correct)
- Compiler error

During runtime

- correct result
- incorrect result
- program crashes
- program does not **terminate** (endless loop)

Comments

```
print("This is a Python program")

x = 20
print("The value of x is", x)
y = x * x    # y is the square of x
print("The value of y is", y)
z = y * y    # z is the square of y
print("The value of z is", x * x * x * x)
```

Comments

Comments and Layout

Comments

- are contained in every good program
- document, **what** and **how** a program does something and how it should be used
- are ignored by the compiler
- Syntax: # until the line end

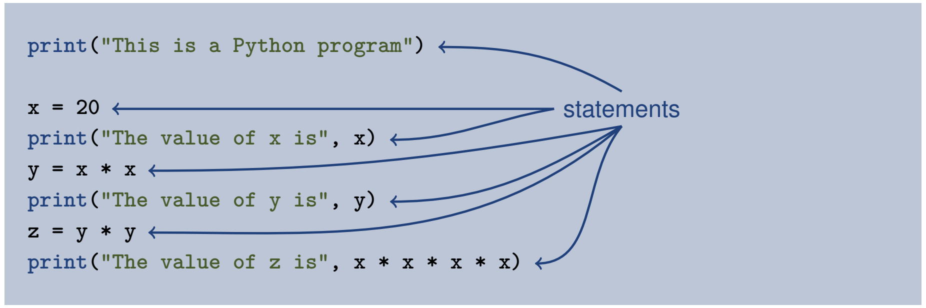
Please note

- empty lines are ignored
- Python dictates indentations that reflect the program logic

Introduction to Python

Statements

Statements



```
print("This is a Python program")

x = 20
print("The value of x is", x)
y = x * x
print("The value of y is", y)
z = y * y
print("The value of z is", x * x * x * x)
```

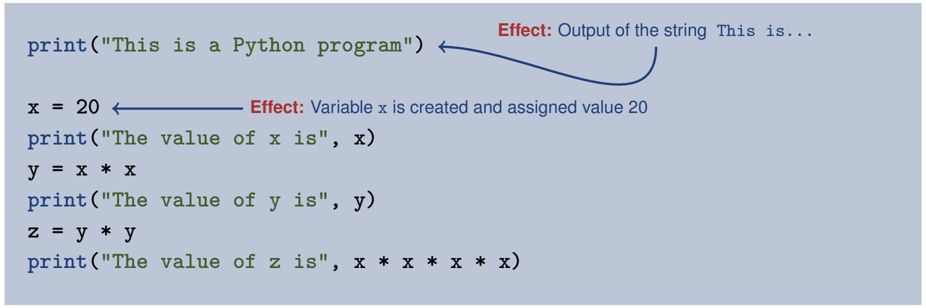
Statements

Statements

- are building blocks of a Python program
- are **executed** (sequentially)
- are given in one line

Any statement (potentially) provides an **effect**

Statements – Values and Effects



```
print("This is a Python program")

x = 20
print("The value of x is", x)
y = x * x
print("The value of y is", y)
z = y * y
print("The value of z is", x * x * x * x)
```

Effect: Output of the string `This is...`

Effect: Variable `x` is created and assigned value 20

Introduction to Python

Variables

Fundamental Types

Variables represent (varying) values

- integers
- real numbers (float)
- strings
- ...

In contrast to, for example, **Java** or **C**, the type is not explicitly stated when a variable is declared (used for the first time)

Introduction to Python

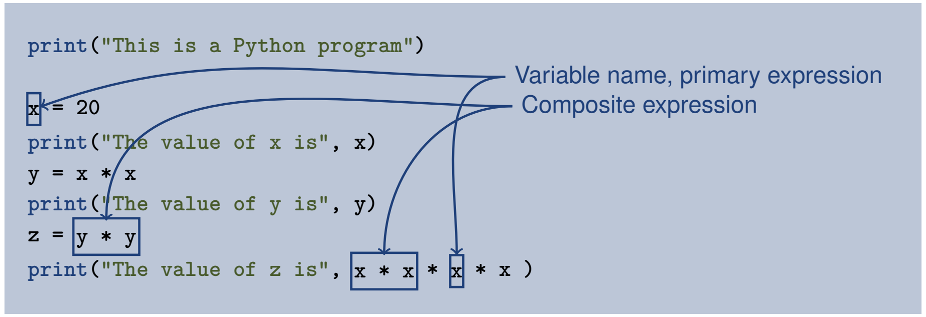
Expressions

Expressions

Expressions

- represent **computations**
- are either **primary** (`x`)
- or **composed** (`x * x`)
- ...from different expressions by **operators**
- ...and parentheses

Expressions



```
print("This is a Python program")

x = 20
print("The value of x is", x)
y = x * x
print("The value of y is", y)
z = y * y
print("The value of z is", x * x * x * x )
```

Variable name, primary expression

Composite expression

Expressions

- represent **computations**
- are **primary** or **composite**
  (by other expressions and operations)

Example

`a * a` is composed of
variable name, operator symbol, variable name
variable name: primary expression

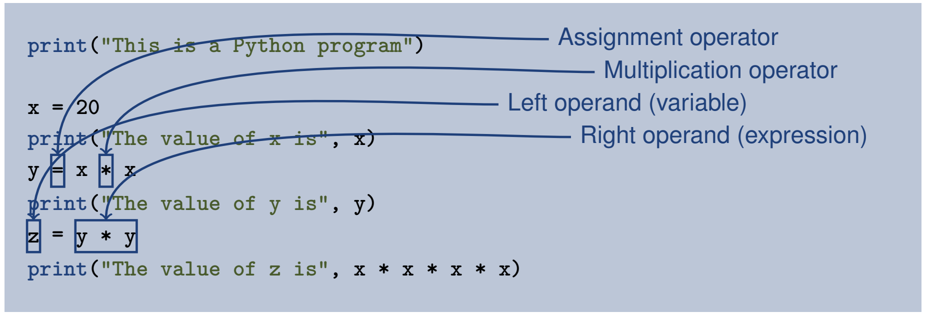
- can be put into parentheses

`a * a` can be written as `(a * a)`

Introduction to Python

Operators and Operands

Operators and Operands



```
print("This is a Python program")

x = 20
print("The value of x is", x)
y = x * x
print("The value of y is", y)
z = y * y
print("The value of z is", x * x * x * x)
```

Operators

Operators

- make expressions (**operands**) into new composed expressions
- have an arity

Example (Multiplication) `a * a`

Operand `a`, Operator `*` , Operand `a`

Multiplication Operator ∗

Multiplication operator

- expects two R-values of the same type as operands (arity 2)
- “returns the product as value of the same type,” that means formally:

> The composite expression is value of the product of the value of the two operands

Examples

- `a * a`
- `b * b`

Assignment Operator =

- Assigns to the left operand the value of the right operand and returns the left operand

Examples

- `b = b * b`
- `a = b`

Attention

The operator "‘=”’ corresponds to the assignment operator of mathematics ($:=$), not to the comparison operator ($=$)

Exercise – Celsius to Fahrenheit Calculator

Write a program that

- interprets a number (like, e. g., 31) as a temperature in degree Celsius
- outputs the same temperature in degree Fahrenheit
- uses the formula

$$\text{fahrenheit} = \frac{9 \cdot \text{celsius}}{5} + 32$$

Exercise – Celsius to Fahrenheit Calculator

```
celsius = 31
fahrenheit = 9 * celsius / 5 + 32
print(fahrenheit)
```