DM550 / DM857
Introduction to Programming

Peter Schneider-Kamp

petersk@imada.sdu.dk

http://imada.sdu.dk/~petersk/DM550/

http://imada.sdu.dk/~petersk/DM857/

COURSE ORGANIZATION

Course Elements

- Lectures Mondays 12-14 in all weeks
- Lectures Thursdays 14-16 in all weeks except 38, 41, 46, 48, 49

- Exercises (marked "TE" in your schedule)

- Labs (marked "TL" in your schedule)

- Exam = project consisting of 2 practical project assignments and 3 project qualification tests

Sections

- 5 sections
  - H7 (ABM), H8 (TF), & H9 (MGK):
    - Computer Science (1st year)
  - H1 (KA) & M1 (TF):
    - Mathematics-Economy (2nd year)
    - Applied Mathematics (2nd year)
    - Minor in Computer Science (2nd year)
- 4 teaching assistants
  - ABM = Anders Bjørn Moeslund
  - KA = Kristoffer Abell
  - MGK = Mads Grau Kristensen
  - TF = Tobias Frisch

Single Point of Contact

- First point of contact is the teaching assistant of your section
- Anders Bjørn Moeslund is the Head Teaching Assistant
- Special mail adress for (almost) all further questions/issues:
  - prog@imada.sdu.dk
- Contact Peter Schneider-Kamp directly in case of:
  - Complaints about/issues with teaching assistants

Practical Issues / Course Material

- You need an IMADA account (≠ SDU account), IF you want to use the terminal room.
- Regularly check one of the (identical) course home pages:
  - http://imada.sdu.dk/~petersk/DM550/
  - http://imada.sdu.dk/~petersk/DM857/
  - Slides, weekly notes, projects, additional notes
- Reading material for the Python part:
  - Allen B. Downey: *Think Python*, 2nd edition, version 2.2.21, Green Tea Press, 2017.
  - Available as PDF and HTML from: http://greenteapress.com/wp/think-python-2e/

Code Café

- manned Code Cafe for students
- first time Wednesday, September 6
- last time Wednesday, December 20
- closed in Week 42 (efterårsferie)

- Mondays, 15.00 – 17.00, Nicky Cordua Mattsson
- Wednesdays, 15.00 – 17.00, Troels Risum Vigsøe Frimer
-

- Nicky and Troels can help with any coding related issues
- issues have to be related to some IMADA course (fx this one)

Expected Workload

- Prepare/attend/process lecture: 1+2+1 hour
- Prepare/attend exercise/lab: 2+2 hours
- Project: 2+50+2+50+2
- Total: 21*4 + 21*4 + = 274 hours (10 ECTS)

Course Goals

- **Solve problems by writing computer programs**

- To this end, you will learn
  - to view programming as a problem solving approach
  - principles of imperative & object-oriented programming
  - how to model, implement, test, debug, and document programs

- Focus on general principles, NOT on the languages Python and Java

Course Contract

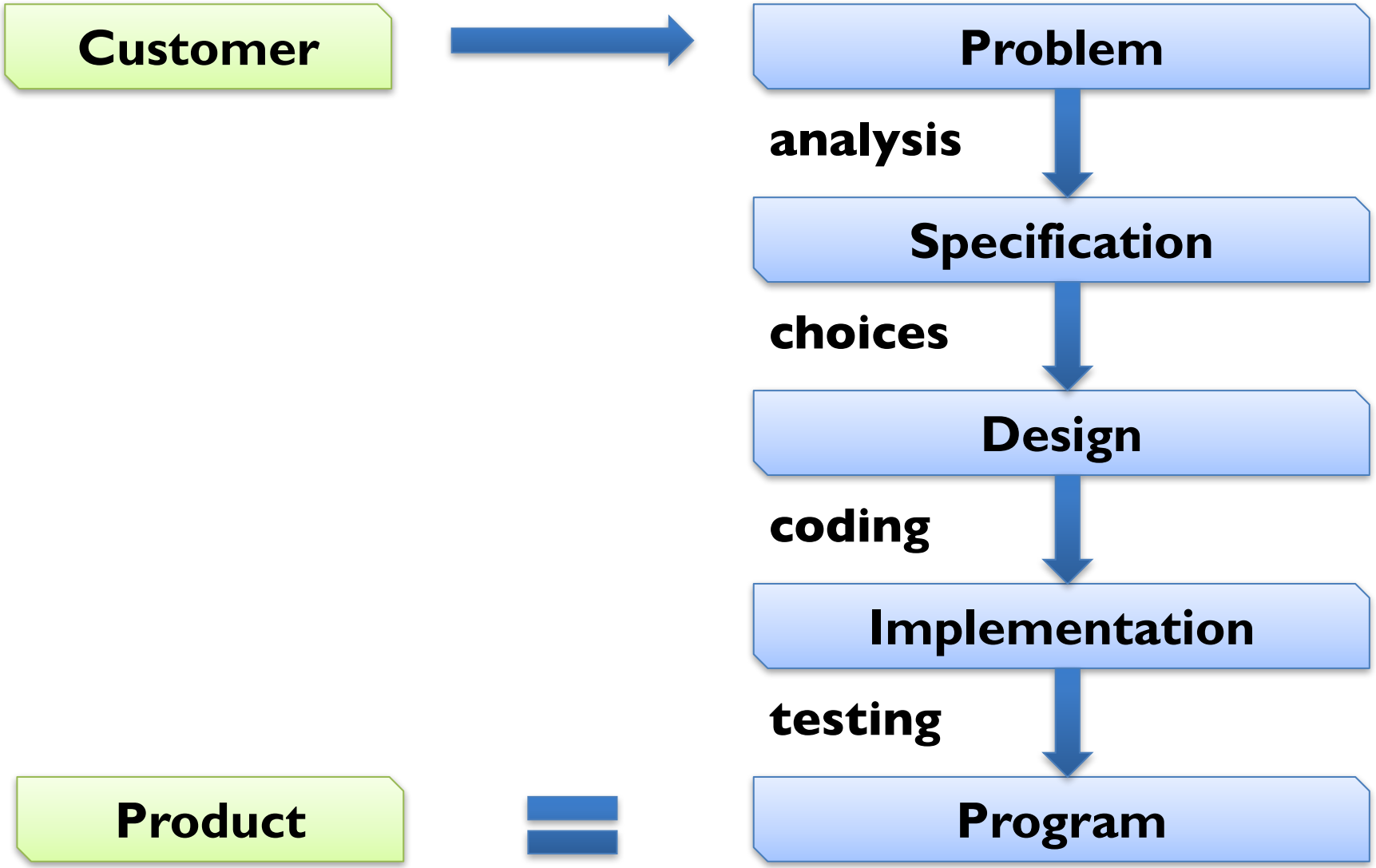
- I am offering you the following:
  1. I explain all needed concepts (as often as needed)
  2. I answer your questions (in class and during breaks)
  3. I guide your learning by assigning exercises

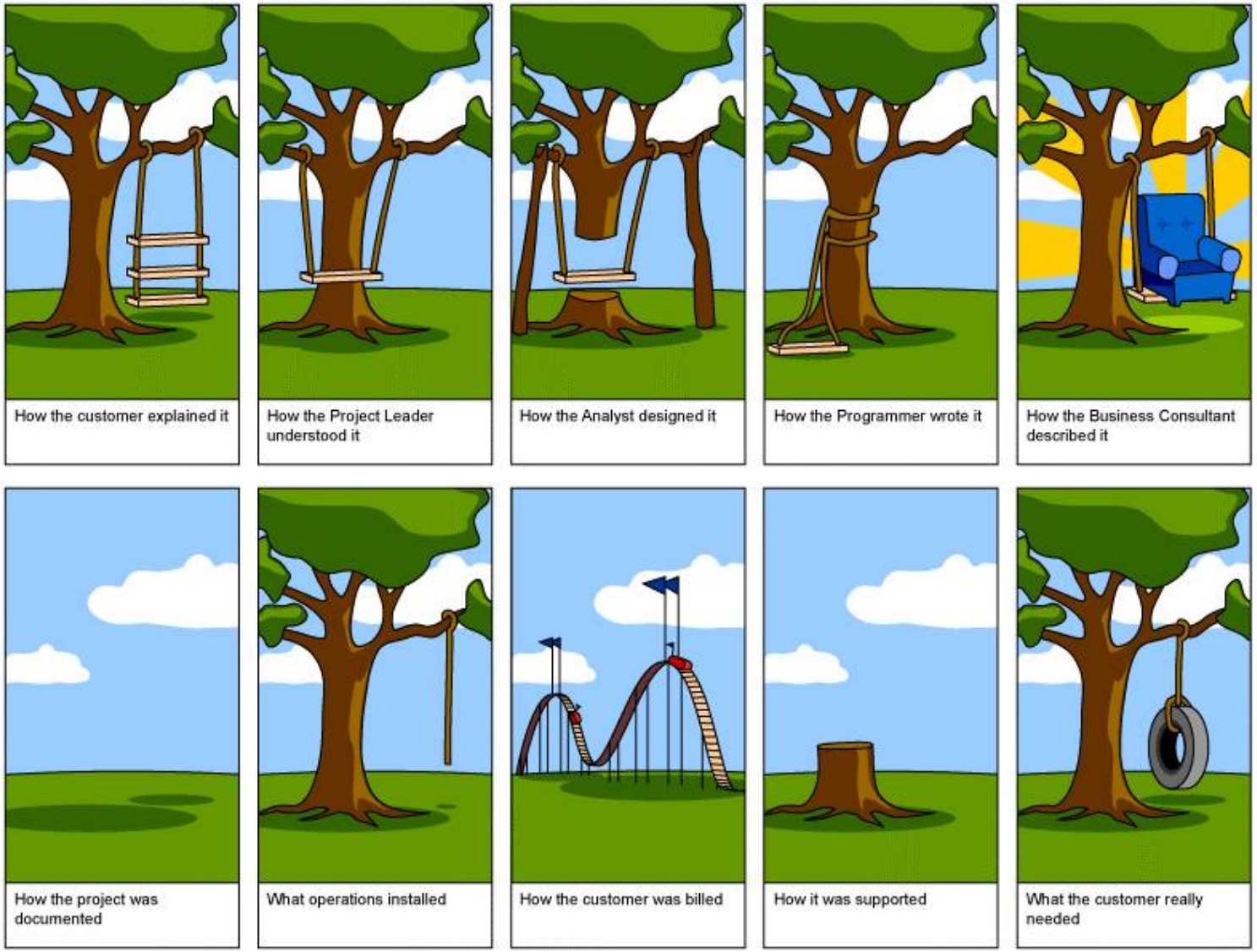
- From you I expect the following:
  1. You ask questions, when something is unclear
  2. You contact your TA (or head TA) early, when you need help
  3. You prepare for lectures and discussion sections

- You and I have the right and duty to call upon the contract!

PROGRAMMING

Programming as Problem Solving





Real Life "Programming"

Programming in a Nutshell

- Computers only have very limited abilities
- Computers are used to solve complex problems

- Programmers needed to break down complex problems into a sequence of simpler (sub-)problems

- program = sequence of simple instructions
- instructions = vocabulary of a programming language

- Programmers needed to express problems as sequence of instructions understandable to the computer

Simple Instructions

- Administrative:
```
from math import sqrt
```
- Input:
```
a = float(input())
b = float(input())
```
- Arithmetic operations:
```
c = sqrt(a**2+b**2)
```
- Output:
```
print("Result:", c)
```
- That is basically ALL a computer can do.

Combining Instructions

- Sequence:

  <instr$_1$>; <instr$_2$>; <instr$_3$>

- Conditional Execution:

  if <cond>:
      <instr$_1$>; <instr$_2$>
  else:
      <instr$_3$>; <instr$_4$>; <instr$_5$>

- Subprograms / Functions:

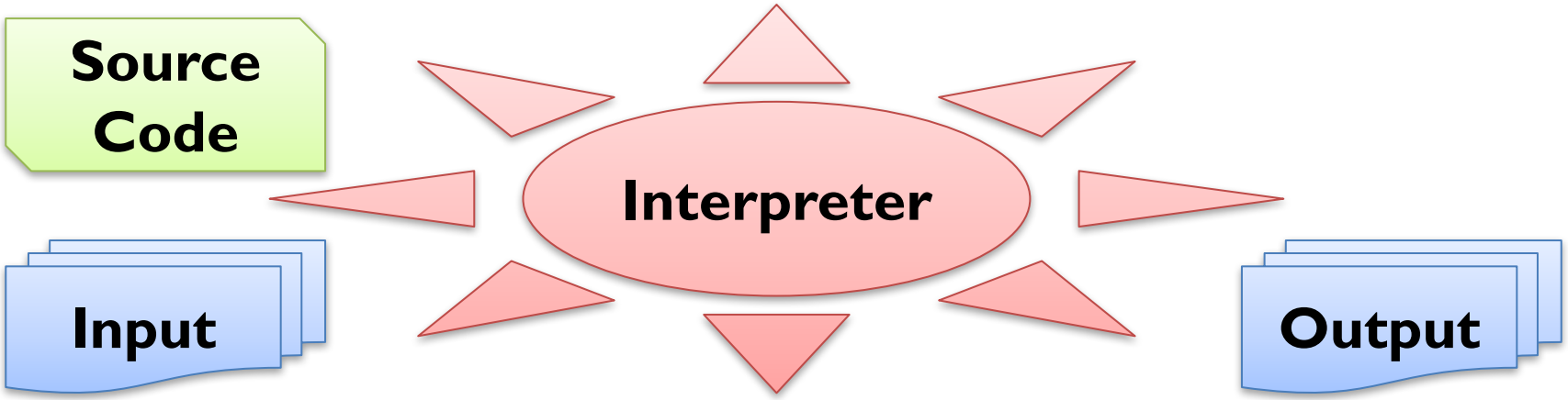
  def <function>(<argument>):
      <instr$_1$>; <instr$_2$>
  <var> = <function>(<input>)

- Repetition:

  while <cond>:
      <instr$_1$>; <instr$_2$>; <instr$_3$>

Executing Programs

- Program stored in a file (*source code* file)
- Instructions in this file executed top-to-bottom
- Interpreter executes each instruction



Debugging

- Any reasonably complex program contains errors
- Three types of errors (in Python)
  - Syntax Errors `a = input)(`
  - Runtime Errors `c = 42 / 0`
  - Semantic Errors `c = a**2+b**2`
- Debugging is finding out why an error occurred

VARIABLES, EXPRESSIONS & STATEMENTS

Values and Types

- Values = basic data objects — 42, 23.0, "Hello!"
- Types = classes of values — integer, float, string

- Values can be printed:
  - print(<value>) — print("Hello!")

- Types can be determined:
  - type(<value>) — type(23.0)

- Values and types can be compared:
  - <value> == <value> — type(3) == type(3.0)

Variables

- variable = name that refers to a value
- program state = mapping from variables to values

- values are *assigned* to variables using "=":
  - <var> = <value> b = 4

- the value referred to by a variable can be printed:
  - print(<var>) print(b)

- the type of a variable is the type of the value it refers to:
  - type(b) == type(4)

Variable Names

- start with a letter (convention: a-z)
- contain letters a-z and A-Z, digits 0-9, and underscore "_"

- can be any such name except for 33 reserved names:

and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	
def	for	lambda	return	

Multiple Assignment

- variables can be assigned to different values at different times:
  - Example:
    ```
    x = 3
    x = 4
    ```
  - Instructions are executed top-to bottom => x refers to 4

- be careful, e.g., when exchanging values serially:
  - Example:
    ```
    x = y
    y = x
    ```
  - later x and y refer to the same value
  - Solution 1 (new variable): `z = y; y = x; x = z`
  - Solution 2 (parallel assign.): `x, y = y, x`

Operators & Operands

- Operators represent computations: + * - / // % **
  - Example: 23+19 day+month*30 2**6-22

- Addition "+", Multiplication "*", Subtraction "-" as usual
- Exponentiation "**": x**y means x^y

- Division "//" rounds down integers:
  - Example 1: 21//42 has value 0, NOT 0.5
  - Example 2: -21//42 has value -1
- For modulo "%" and "//": x == (x//y)*y + (x%y)

Expressions

- Expressions can be:
  - Values: 42 23.0 "Hej med dig!"
  - Variables: x y name1234
  - built from operators: 19+23.0 x**2+y**2
- grammar rule:
  - <expr> => <value> |
    <var> |
    <expr> <operator> <expr> |
    (<expr>)
- every expression has a value:
  - replace variables by their values
  - perform operations