Aalto University
School of Engineering

YYT-C3002
Application Programming in Engineering

Introduction

Anas Altartouri
Otaniemi 8.1.2019

Lecture content

Course content and structure

Motivation – why do we need to program?

What kind of programming – overview and basic concepts

Useful tools

Demos

Course admin

Lecture-based exercises

- Individual work (mainly!)
- A mix of reading materials and computer-based
- Exercise sessions follow each lecture at 12-14 – some sessions are supported (i.e., the lecturer is there)

→ **Follow lecturers' instructions about the type of their exercises**

→ **Exercises are compulsory – you need to pass all exercises to get the course credits!**

→ **Your performance in the exercises will determine 40% of the final grade. The other 60% is from the course exam**

Course content and structure

Date	Topic	Lecturer
8.1.2019	Introduction to the course	Anas Altartouri
10.1.2019	Software engineering – Introduction	Teemu Peltonen
15.1.2019	Software engineering – Process and methods	Teemu Peltonen
17.1.2019	Software engineering – Delivering quality	Teemu Peltonen
22.1.2019	GIS I – Geoprocessing with FOSS and extending desktop GIS	Anas Altartouri
24.1.2019	GIS II – Spatial databases and web map applications	Anas Altartouri
29.1.2019	Matlab programming for FEM	Jarkko Niiranen
31.1.2019	R programming and R extensions – Hydrostreamer	Marko Kallio
5.2.2019	Software architectures and clouds	Jussi Nikander
7.2.2019	Optimization with Matlab and external solvers	Jani Romanoff
12.2.2019	System integration and interoperability	Jussi Nikander
	EXAM	

Software …

“Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.”[1]

Application software

- System-specific
- Task-specific
- User interaction

[1] Sommerville, Ian (2015). *Software Engineering* (10th ed.). Pearson.
Chapter One available from: https://iansommerville.com/software-engineering-book/web/sample-chapters/

Software engineering

Software engineering

"Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use."[1]

Activities in software production (Sommerville, 2015, p. 7)

- Defining specifications
- Development: designing and **programming**
- Testing and validation
- Evolution

[1] Sommerville, Ian (2015). *Software Engineering* (10th ed.). Pearson.
Chapter One available from: https://iansommerville.com/software-engineering-book/web/sample-chapters/

Should an engineer be able to code? Why?

"***GIS users should be equipped to do the equivalent of swimming a few hundred yards, but need not look like or beat Michael Phelps.***"

Adena Schutzberg (Direction Magazine 27.02.2012)

→ Not specific to GIS and the field of Geoinformatics! The same can be said for engineers in other fields

Should an engineer be able to code? Why?

- **Automate your workflow/routine**
- **Write models/functions that applications do not provide**
- **Have control over the process (adjust computations, memory allocation, disk usage, output type, etc.)**
- **Understand how data are structured and how methods work**
- **Communicate and share your computations/models**
- **Knowing the process of software production would facilitate communication with programmers and software engineers (especially in cross-disciplinary application development)**

Should an engineer be able to code? Why?

Your skills in application programming will benefit you in almost any direction you take

In the industry

"*Now every company is a software company*" David Kirkpatrick (Forbes Magazine 30.11.2011)

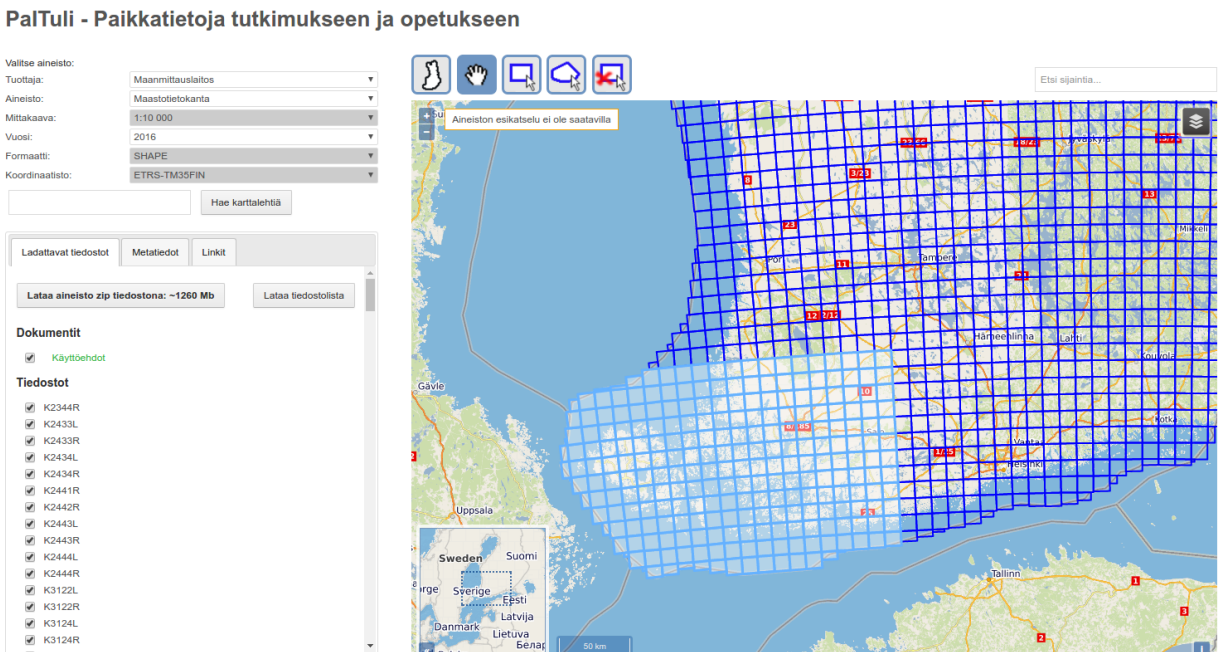
In academia

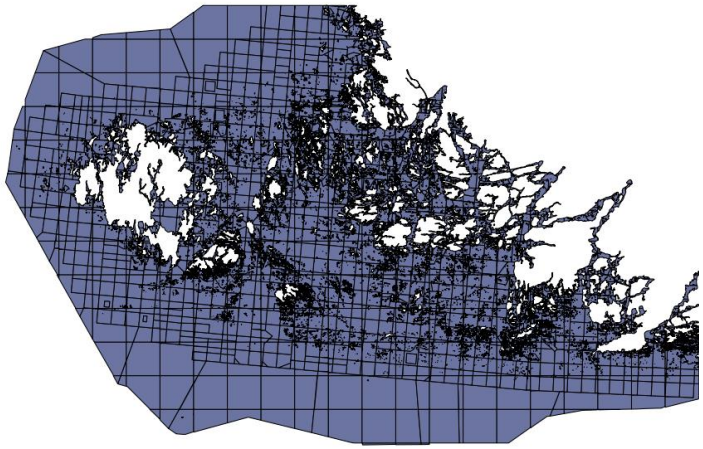
- Tools for your own research
- System development as a research methodology [1]

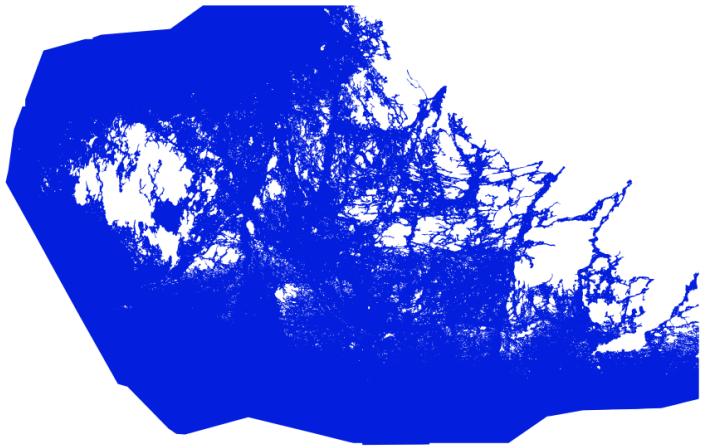
[1] Nunamaker, J. F., Chen, M., Purdin, T. D. M. (1990–91). Systems development in information systems research. Journal of Management Information Systems, 7(3), 89–106. (http://gkmc.utah.edu/7910F/papers/JMIS%20systems%20development%20in%20IS%20research.pdf)

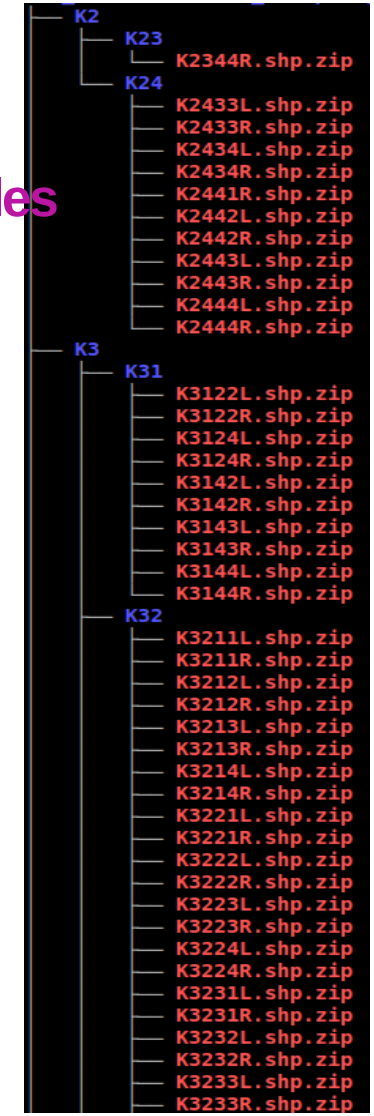
Motivating example 1

Programming to automate processing of a large number of files







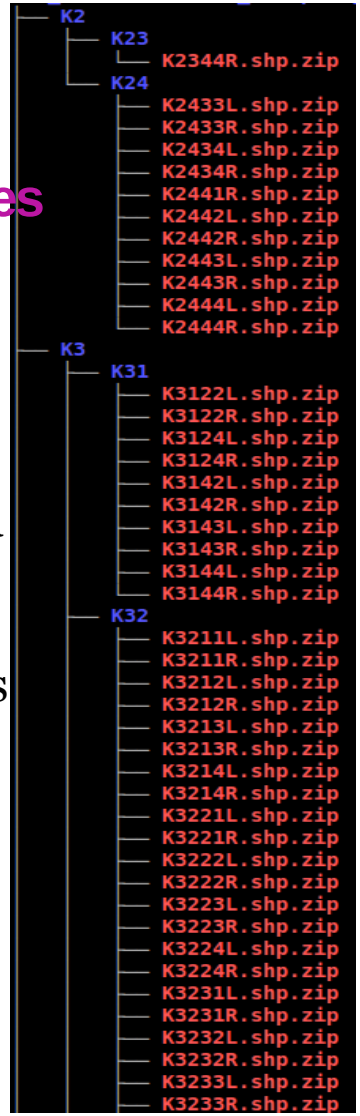


…

16 directories, 254 files

Motivating example 1

Programming to automate processing of a large number of files



…

16 directories, 254 files

Extracting the Archipelago Sea area from the NLS topographic data

→ Create a Shapefile with a single polygon representing the sea area

The procedure:

- Download datasets: 254 ZIP files organized in 16 dirs and sub-dirs
- Unzip the files (254 ZIP files → 4938 x 5 Shapefiles)
- Extract and merge Shapefiles containing information about sea water areas (254 x 5 files)
- Extract polygons that represent sea water from the merger file
- Dissolve the extracted sea water polygons into a single polygon

Solved using Linux shell command-line programs

- Bash `for` loop, `wget`, `unzip`, `cp`, GDAL `ogr2ogr`

Motivating example 2

Programming for the lack of task-specific tools & for big data handling

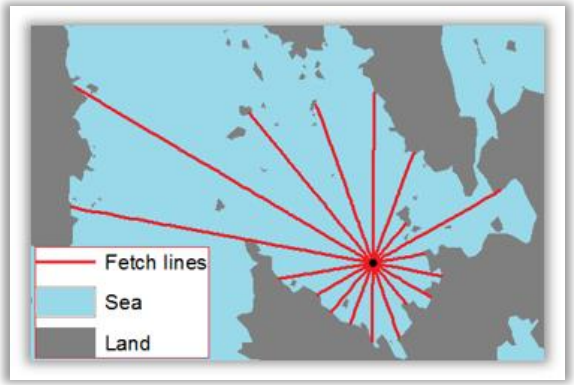
Estimation of relative sea openness based on fetch line abstraction

Given:

Water Framework Directive areas of the Golf of Finland

Grid of points with 100 m x- and y-spacing

18 radial lines from each point



The task:

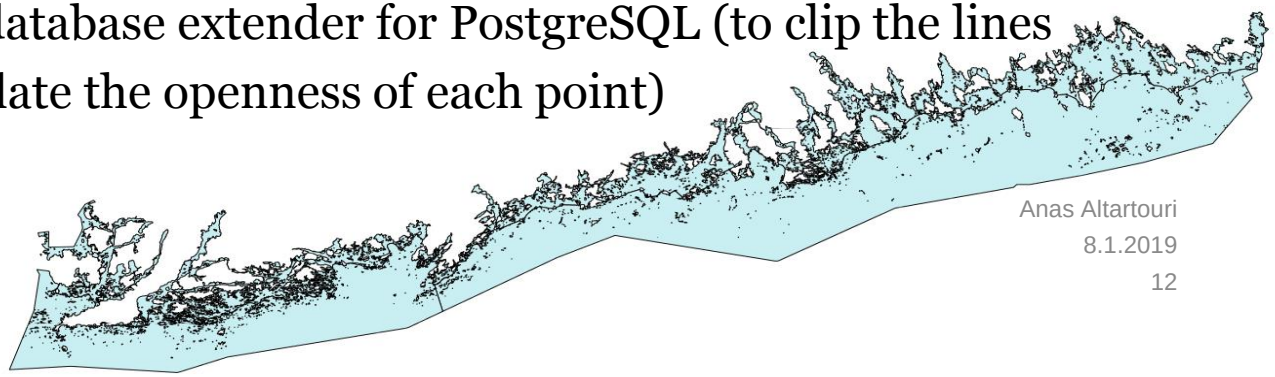
Clip more than 10,000,000 line,

with a 150,000 vertex polygon (simplified)

Solved using:

PySAL: a library of spatial analysis functions (to create the radial lines)

PostGIS: a spatial database extender for PostgreSQL (to clip the lines and calculate the openness of each point)



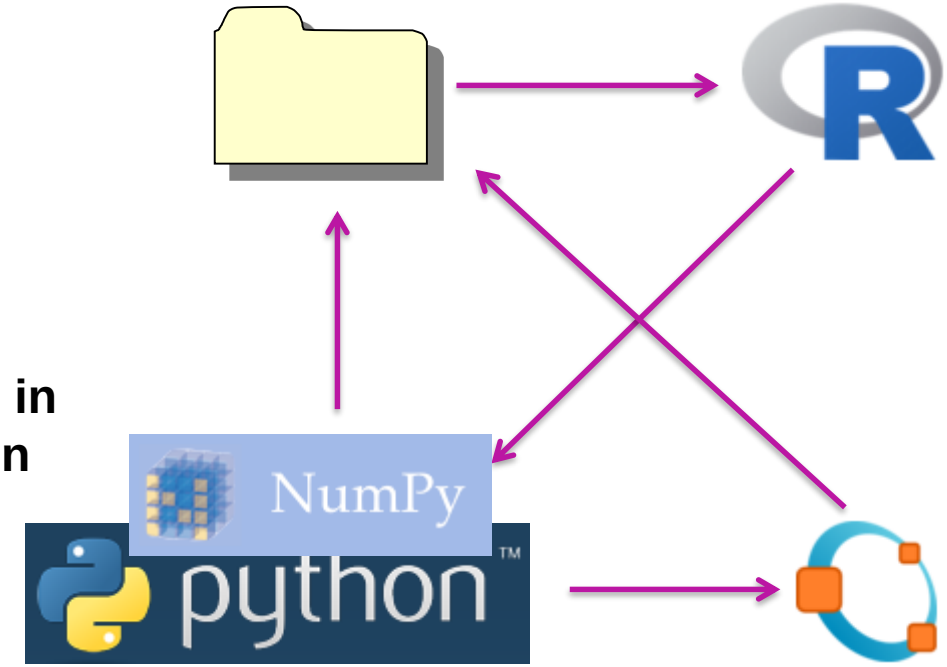
Motivating example 3

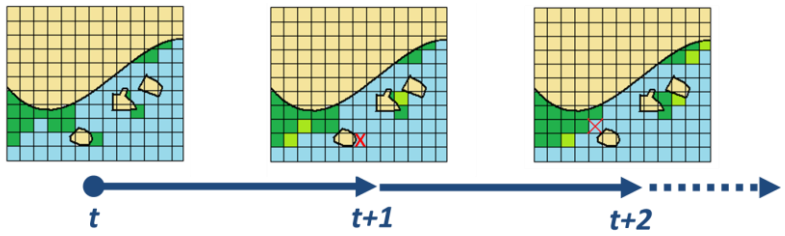
Programming for workflow automation & tailoring

A machine learning based cellular automata model of species distribution

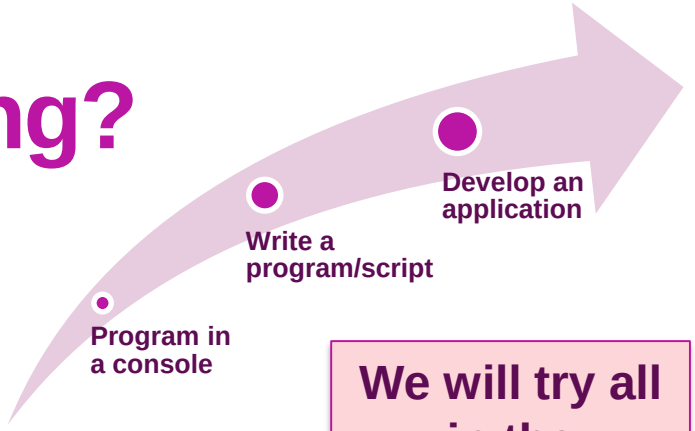
Utilizing multiple functions written in different languages and available in different software packages

- GNU Octave: language for numerical computations
- NumPy: Python extension that provides support for large, multi-dimensional arrays and matrices
- R: software environment for statistical computing and graphics





What kind of programming?



We will try all in the exercises!

Programming directly in a console

Perform a sequence of processing/computing tasks

Writing a program/script

Run a sequence of processing/computing tasks (a piece of code) frequently, e.g.:

- *Run some models with various input parameters and/or datasets*
- *Run a routine frequently with different datasets*

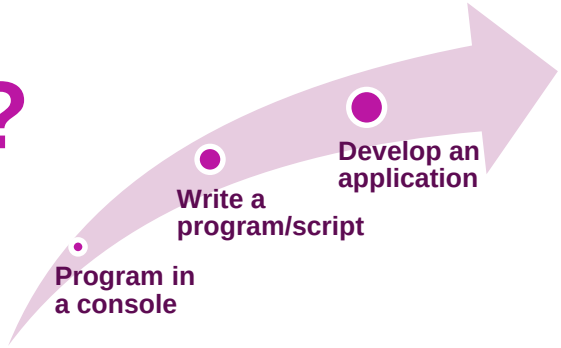
Developing an application software

Allow interaction with your program/script, e.g.:

- *To avoid editing variables in the code every time you run it*
- *To publish your code and provide an application software*

What kind of programming?

Programming directly in a console



Perform a sequence of processing/computing tasks, e.g., with:

- R, Matlab, Python, Bash (Unix shell, a command-line interpreter)

→ **Know the language**

- Syntax
- Data types (Boolean, numeric, string, arrays, spatial data types, etc.)
- Control flow (conditions, iterations, function calls)

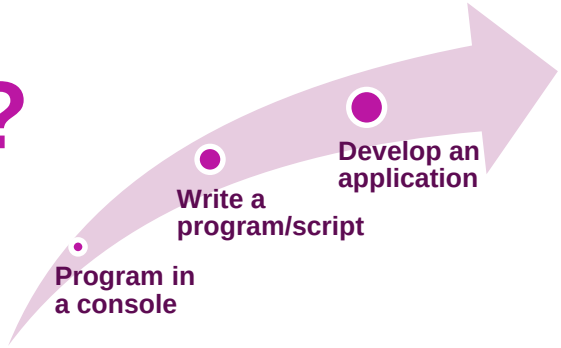
→ **Use software libraries**

- What libraries are available
- Understand the libraries' APIs (application programming interface)
  - *Example: matplotlib histogram function*
    https://matplotlib.org/api/_as_gen/matplotlib.pyplot.hist.html#matplotlib.pyplot.hist


user@osgeolive: ~
user@osgeolive:~$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for mor
e information.
>>> import numpy as np
>>> a = np.arange(12)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
>>> a.size
12
>>> a.ndim
1
>>> a.reshape(2,6)
array([[0, 1, 2, 3, 4, 5],
       [6, 7, 8, 9, 10, 11]])
>>> a.reshape(2,6).ndim
2
>>> a.reshape(3,4).size
12
>>> a.reshape(3,4).shape
(3, 4)
>>>

What kind of programming?

Writing a program/script



Run a program consisting of a sequence of processing/computing tasks

→ Know the language

- Compiled (compile & link)
- Interpreted

Read more: http://www2.hawaii.edu/~takebaya/ics111/process_of_programming/process_of_programming.html

→ Structure your code

- Procedural
- Object-oriented

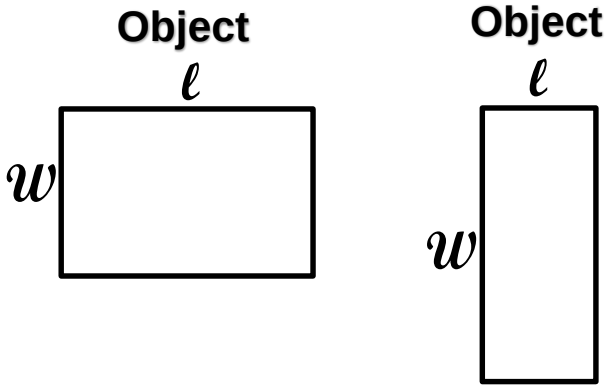
Read more: https://www.cs.utah.edu/~germain/PPS/Topics/oop.html

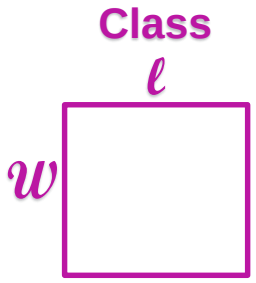
https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zappldev/zappldev_85.htm

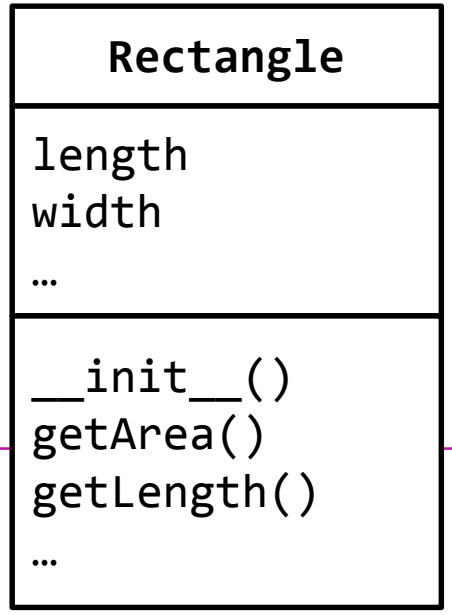
```
a_l = 4 #length in meters
a_w = 3 #width in meters
a = (create_a_rectangle_with_l&w
     …
     …
     …)
a_area = a_l * a_w
a_perimeter = 2 * (a_l + a_w)

b_l = 5
b_w = 2
b = (create_a_rectangle_with_l&w
     …
     …
     …)
b_area = b_l * b_w
b_perimeter = 2 * (b_l + b_w)
```

Cleaner code
Easier to maintain

```
def create_a_rectangle(l, w)
    rectangle = (create_a_rectangle_with_l&w
                 …
                 …
                 …)
    return rectangle

def area(l, w)
    a = l * w
    return a

def perimeter(l, w)
    p = 2 * (l + w) * 100
    return p

a_l = 4
a_w = 3
a = create_a_rectangle(a_l, a_w)
a_area = area(a_l, a_w)
a_perimeter = perimeter(a_l, a_w)

b_l = 5
b_w = 2
b = create_a_rectangle(b_l, b_w)
b_area = area(b_l, b_w)
b_perimeter = perimeter(b_l, b_w)
```

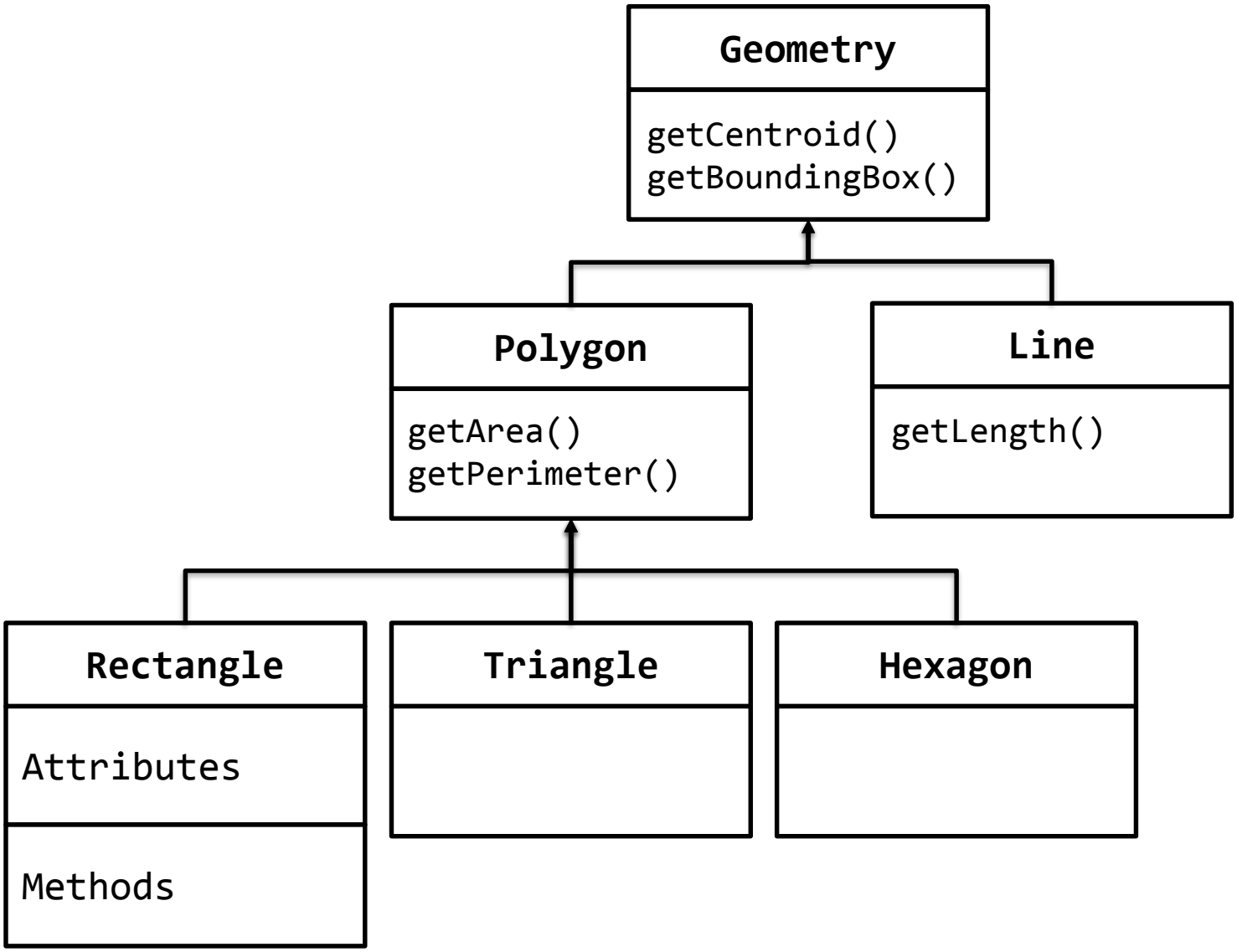






A”

```
class Rectangle:

    def __init__(self, l, w)
        self.length = l
        self.width = w
        (create_a_rectangle_with_l&w … …)

    def getArea(self)
        a = self.length * self.width
        return a

    def getPerimeter(self)
        p = 2 * (self.length + self.width)
        return p

a_l = 4
a_w = 3
a = Rectangle(4, 3)
a_area = a.getArea()
a_perimeter = a.getPerimeter()

b_l = 5
b_w = 2
b = Rectangle(5, 2)
b_area = b.getArea()
b_perimeter = b.getPerimeter()
```


Geometry
getCentroid()
getBoundingBox()
Polygon
getArea()
getPerimeter()
Line
getLength()
Rectangle
Attributes
Methods
Triangle
Hexagon

Object-oriented programming & Unified Modeling Language (UML)

Unified Modeling Language (UML)

"A general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to **visualize the design of a system**. "[1]

13 diagrams of two types:

Structural UML diagrams, e.g.:

- **Class diagram**
- Package diagram
- Object diagram

Behavioral UML diagrams, e.g.:

- Activity diagram
- Sequence diagram
- Use case diagram

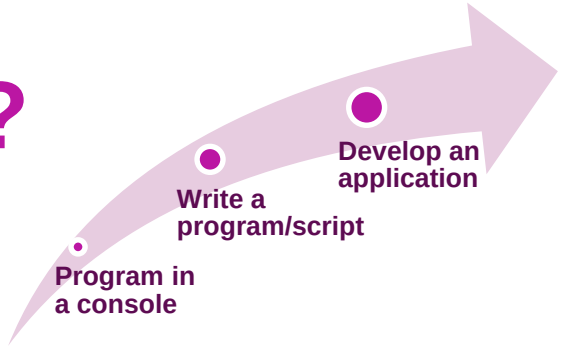
More about object-oriented programming and UML:

https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/index.html
https://docs.oracle.com/javase/tutorial/java/concepts/

[1] https://en.wikipedia.org/wiki/Unified_Modeling_Language

What kind of programming?

Developing an application software



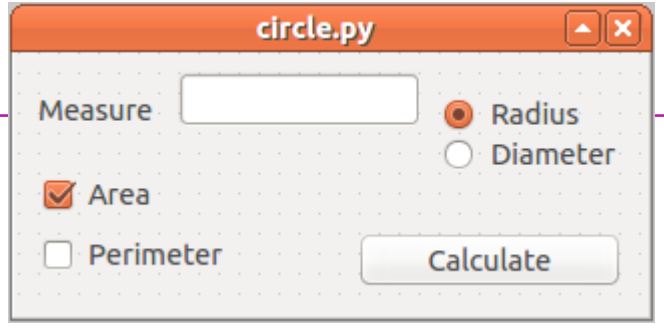
Allow user interaction with the application!

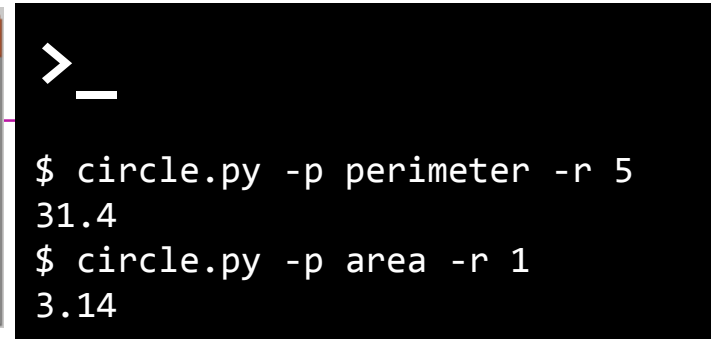
- Command-line or GUI
- Stand-alone or extension
- Desktop or web

Software engineering and system development

- Analysis and design, architecture, prototyping
- Evaluation, usability engineering
- Human–computer interaction

→ More in the following lectures!

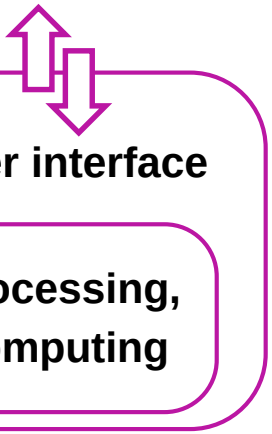




Terms

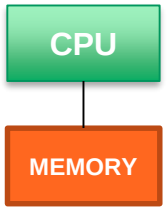
- Variable
- Data type
- Control flow
- Subroutine, function
- Software library, module
- Class and objects
  - Attributes
  - Methods, parameters
- Events and listeners

What kind of programming?

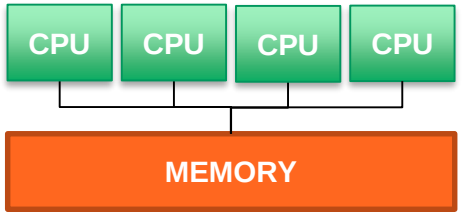


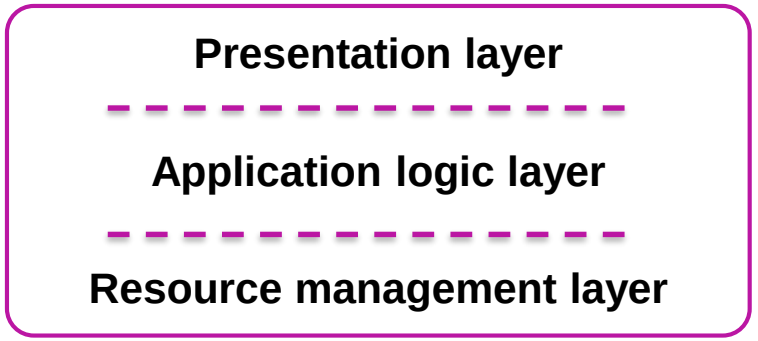
Application software

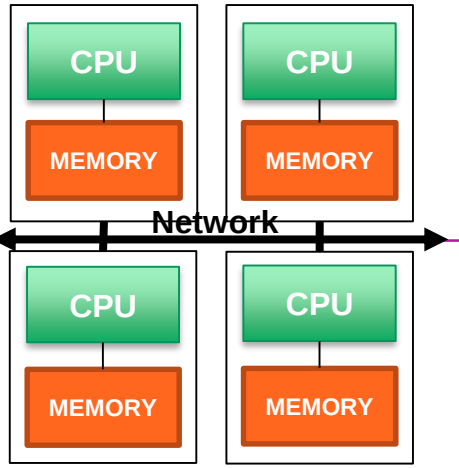
Information system

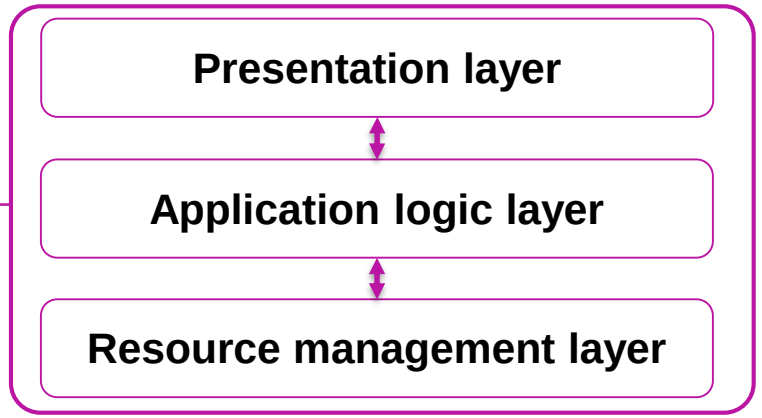


Distributed systems



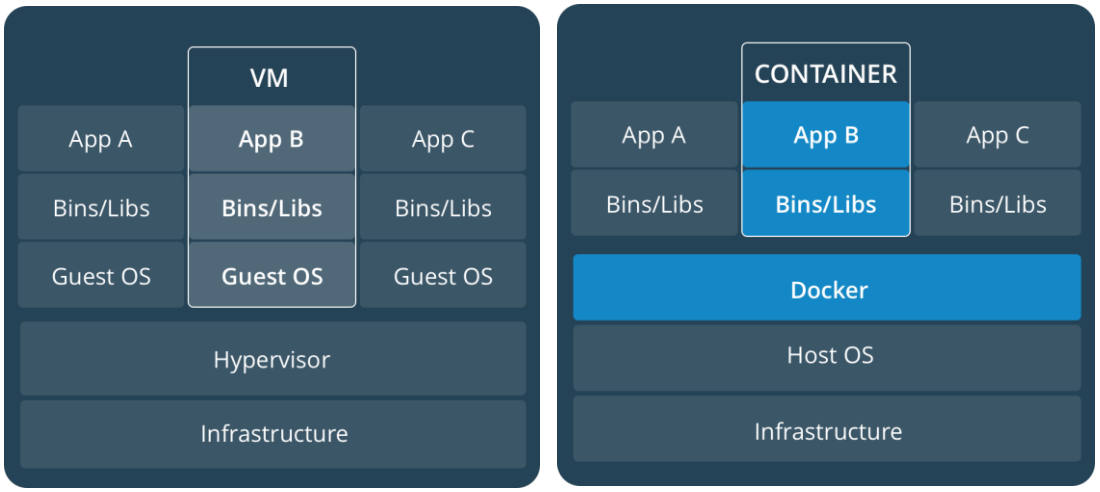






Tools

Image source: https://www.docker.com/what-container (2018)



Virtual machines[1]

- Host OS – the operating system of the physical computer[1]
- Guest OS – the operating system running inside the virtual machine[1]
- VM – the environment that the hypervisor creates for the guest OS[1]

Containers[2]

"A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings"[2]

Notebooks – CSC: https://notebooks.csc.fi/#/

[1] https://www.virtualbox.org/manual/ch02.html
[2] https://www.docker.com/what-container

Tools

“Why is virtualization important?

- Running multiple operating systems simultaneously
- Easier software installations
- Testing and disaster recovery
- Infrastructure consolidation”[1]

Run the VM

- Locally
- On the cloud

[1] https://www.virtualbox.org/manual/ch02.html

Tools

A VM example:

- OSGeo-Live – “A self-contained bootable DVD, USB thumb drive or Virtual Machine based on Lubuntu, that allows trying a wide variety of open source geospatial software without installing anything.”[1]

A Notebook example:

- Jupyter – “The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.”[2]

[1] Source: https://live.osgeo.org
[2] http://jupyter.org/

Demos …

Scripting in a console …

A command-line program …

A desktop extension …

A web application …

CSC notebooks …

Virtual machines …

Locally …

On the cloud (CSC's cPouta cloud service) …

Exercise

Reading material

1. Sommerville, I. (2015). Software Engineering (10th edition). Pearson. Read Chapter 1: **Introduction**, available from: https://www.dropbox.com/s/b8u3j74fkigb5vd/Ch01%20Introduction.pdf?dl=0
2. Alonso, G., Casati, F., Kuno, H., Machiraju, V. (2003). *Web services: Concepts, Architectures and Applications*. Springer, Berlin, 354 p. Read Chapter 1: **Distributed Information Systems**, available from the course web page.

Exercise

Questions

1. Indicate the three layers of an information system and describe the role of each layer in the system.
2. Find a software library from your field. Explore:
   - *What purpose the library serves..*
   - *What language in which it is written..*
   - *Whether bindings are available for the library in other languages.*

   Find a function that the library provides. Explore:
   - *What the function does..*
   - *What arguments (parameters) it requires..*
   - *What result it returns.*

→ You are not required to submit any report for this exercise!