

Hello, world!_

Velkommen til IN3040!

Høst 2022

Plan for dagen

- Hva handler dette kurset om?
- Litt praktisk informasjon
- Forelesning: syntaks/semantikk



```java
public class GodDag {
   public static void main(String[] args) {
      System.out.println("Hei på deg!");
      System.out.println("Du bør kunne lese dette - sitter du for langt bak?");
   }
}
```

Personene bak kurset

Forelesere:

- Volker Stolz (stolz@ifi.uio.no)
- Eyvind W. Axelsen (eyvinda@ifi.uio.no)

Gruppelærer

- Christian Sveen (casveen@ifi.uio.no)

Hvem er jeg?

Eyvind W. Axelsen
(eyvinda@ifi.uio.no)

Førsteamanuensis II ved Ifi, UiO

- Det vil si, jeg er deltidsansatt
- Programmeringsteknologi-gruppa (10. etasje)



CTO/IT-direktør ved Fürst Medisinsk Laboratorium

- Norges største medisinske laboratorium, > 500 ansatte
- > 12 000 pasientprøver per dag, > 150 000 analyser per dag
- Lager “ordentlige” programmer daglig i språk som C#, TypeScript og SQL

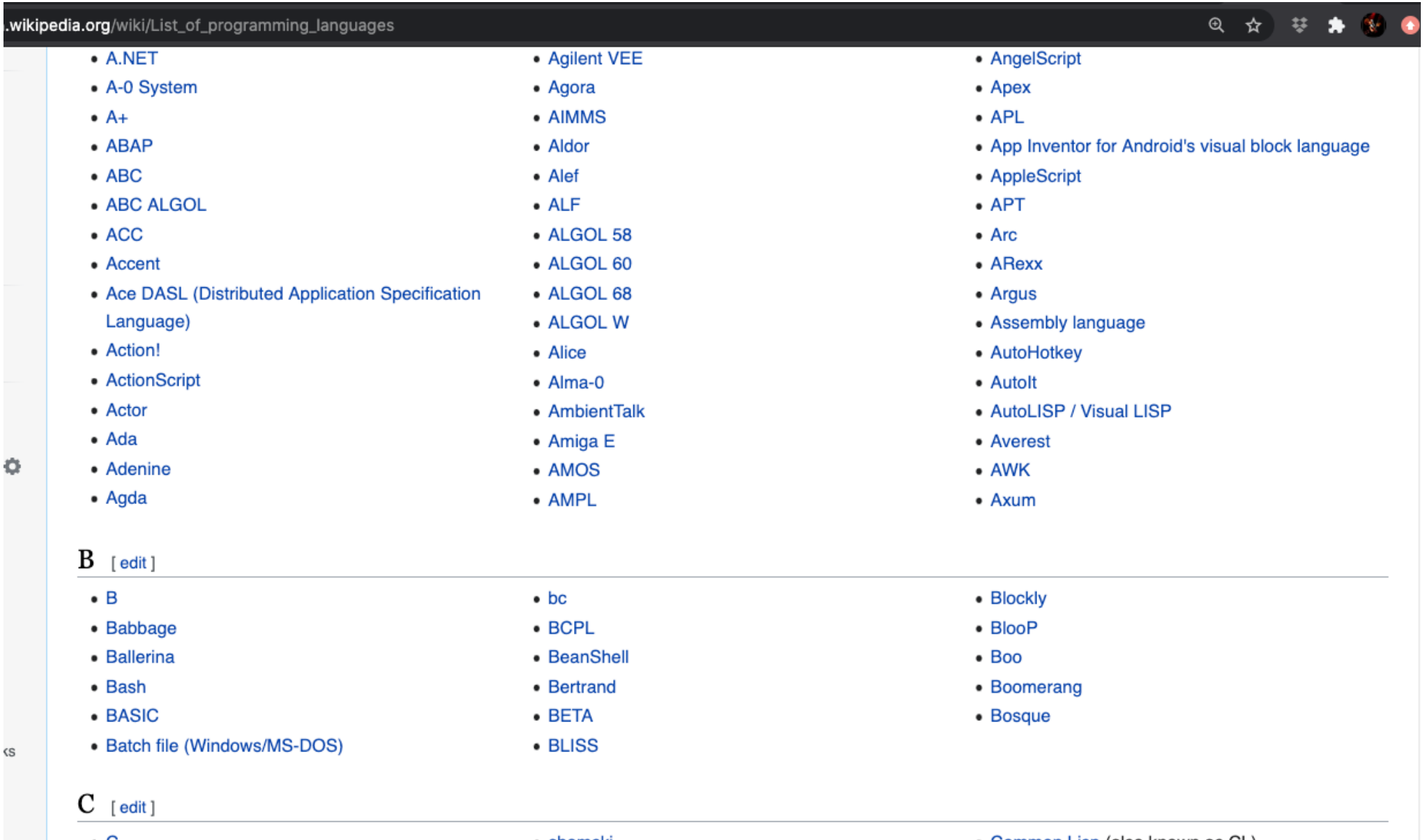


Famliefar, musikkelsker, bassist, ...





En vidunderlig verden av programmeringsspråk!



.wikipedia.org/wiki/List_of_programming_languages

- A.NET
- A-0 System
- A+
- ABAP
- ABC
- ABC ALGOL
- ACC
- Accent
- Ace DASL (Distributed Application Specification Language)
- Action!
- ActionScript
- Actor
- Ada
- Adenine
- Agda
- Agilent VEE
- Agora
- AIMMS
- Aldor
- Alef
- ALF
- ALGOL 58
- ALGOL 60
- ALGOL 68
- ALGOL W
- Alice
- Alma-0
- AmbientTalk
- Amiga E
- AMOS
- AMPL
- AngelScript
- Apex
- APL
- App Inventor for Android's visual block language
- AppleScript
- APT
- Arc
- ARexx
- Argus
- Assembly language
- AutoHotkey
- AutoIt
- AutoLISP / Visual LISP
- Averest
- AWK
- Axum

B [edit]

- B
- Babbage
- Ballerina
- Bash
- BASIC
- Batch file (Windows/MS-DOS)
- bc
- BCPL
- BeanShell
- Bertrand
- BETA
- BLISS
- Blockly
- BlooP
- Boo
- Boomerang
- Bosque

C [edit]

W [edit]

- WATFIV, WATFOR
- WebAssembly
- WebDNA
- Whiley
- Winbatch
- Wolfram Language
- Wyvern

X [edit]

- X++
- X10
- XAML
- xBase
- xBase++
- XBL
- XC (targets XMOS architecture)
- xHarbour
- XL
- Xojo
- XOTcl
- XOD (programming language)
- XPL
- XPL0
- XQuery
- XSB
- XSharp
- XSLT
- Xtend

Y [edit]

- YAML
- Yorick
- YQL
- Yoix
- YUI

Z [edit]

- Z notation
- Zebra, ZPL, ZPL2
- Zeno
- ZetaLisp
- ZOPL
- Zsh
- ZPL
- Z++

Hvilke språk kan du? Opp med hånda!

- Java?
- C#?
- Python?
- JavaScript?
- LISP?
- Clojure?
- Go?
- …?



99 bottles of beer on the wall, 99 bottles of beer.
Take one down and pass it around, 98 bottles of beer on the wall.

98 bottles of beer on the wall, 98 bottles of beer.
Take one down and pass it around, 97 bottles of beer on the wall.

97 bottles of beer on the wall, 97 bottles of beer.
Take one down and pass it around, 96 bottles of beer on the wall.

…

2 bottles of beer on the wall, 2 bottles of beer.
Take one down and pass it around, 1 bottle of beer on the wall.

1 bottle of beer on the wall, 1 bottle of beer.
Take one down and pass it around, no more bottles of beer on the wall.

No more bottles of beer on the wall, no more bottles of beer.
Go to the store and buy some more, 99 bottles of beer on the wall.



https://rosettacode.org/wiki/99_Bottles_of_Beer

```
#!/usr/bin/env python
# -*- coding: iso-8859-1 -*-
"""
99 Bottles of Beer (by Gerold Penz)
Python can be simple, too :-)
"""

for quant in range(99, 0, -1):
   if quant > 1:
      print quant, "bottles of beer on the wall,", quant, "bottles of beer."
      if quant > 2:
         suffix = str(quant - 1) + " bottles of beer on the wall."
      else:
         suffix = "1 bottle of beer on the wall."
   elif quant == 1:
      print "1 bottle of beer on the wall, 1 bottle of beer."
      suffix = "no more beer on the wall!"
   print "Take one down, pass it around,", suffix
   print "--"
```

https://rosettacode.org/wiki/99_Bottles_of_Beer

```
class Beer
{
    public static void Main(string[] args)
    {
        var beerLyric = new StringBuilder();
        string nl = System.Environment.NewLine;

        var beers =
            (from n in Enumerable.Range(0, 100)
             select new
             {
                 Say = n == 0 ? "No more bottles" :
                     (n == 1 ? "1 bottle" : n.ToString() + " bottles"),
                 Next = n == 1 ? "no more bottles" :
                     (n == 0 ? "99 bottles" :
                     (n == 2 ? "1 bottle" : n.ToString() + " bottles")),
                 Action = n == 0 ? "Go to the store and buy some more" :
                                   "Take one down and pass it around"
             }).Reverse();

        foreach (var beer in beers)
        {
            beerLyric.Append($"{beer.Say} of beer on the wall, {beer.Say.ToLower()} of beer.{nl}");
            beerLyric.Append($"{beer.Action}, {beer.Next} of beer on the wall.{nl}");
            beerLyric.AppendLine();
        }
        Console.WriteLine(beerLyric.ToString());
        Console.ReadLine();
    }
}
```

https://rosettacode.org/wiki/99_Bottles_of_Beer

```
let
    val itoa = Makestring.intToStr

    fun getabeer 0 =
        (print "Go to the store and buy some more,\n";
         print "99 bottles of beer on the wall.\n")

      | getabeer 1 =
        (print "1 bottle of beer on the wall,\n";
         print "1 bottle of beer,\n";
         print "Take one down, pass it around,\n";
         print "0 bottle of beer on the wall.\n\n";
         getabeer (0))

      | getabeer x =
        (print (itoa(x)^" bottles of beer on the wall,\n");
         print (itoa(x)^" bottles of beer,\n");
         print "Take one down, pass it around,\n";
         print (itoa(x-1)^" bottles of beer on the wall.\n\n");
         getabeer (x-1))
in
    getabeer 99;
end
```

https://rosettacode.org/wiki/99_Bottles_of_Beer

```
class Bottles
{
    public static void main(String args[])
    {
        String s = "s";
        for (int beers = 99; beers > -1;)
        {
            System.out.print(beers + " bottle" + s + " of beer on the wall, ");
            System.out.println(beers + " bottle" + s + " of beer, ");

            if (beers == 0)
            {
                System.out.print("Go to the store, buy some more, ");
                System.out.println("99 bottles of beer on the wall.\n");
                System.exit(0);
            }
            else
                System.out.print("Take one down, pass it around, ");

            s = (--beers == 1) ? "" : "s";
            System.out.println(beers + " bottle" + s + " of beer on the wall.\n");
        }
    }
}
```

https://rosettacode.org/wiki/99_Bottles_of_Beer

```
(defun bottles-of-bier (n)
  (case n
    (0
     '(No more bottles of beer on the wall no more bottles of beer.
       Go to the store and buy some more 99 bottles of beer on the wall.))
    (1
     `(1 bottle of beer on the wall 1 bottle of beer.
      Take one down and pass it around no more bottles of beer on the wall.
      ,@(bottles-of-bier 0)))
    (2
     `(2 bottles of beer on the wall 2 bottles of beer.
        Take one down and pass it around 1 bottle of beer on the wall.
        ,@(bottles-of-bier 1)))
    (t
     `(,n bottles of beer on the wall ,n bottles of beer.
      Take one down and pass it around
      ,(1- n) bottles of beer on the wall.
      ,@(bottles-of-bier (1- n))))))
```

https://rosettacode.org/wiki/99_Bottles_of_Beer

```
import "fmt"

func main() {
    bottles := func(i int) string {
        switch i {
        case 0:
            return "No more bottles"
        case 1:
            return "1 bottle"
        default:
            return fmt.Sprintf("%d bottles", i)
        }
    }

    for i := 99; i > 0; i-- {
        fmt.Printf("%s of beer on the wall\n", bottles(i))
        fmt.Printf("%s of beer\n", bottles(i))
        fmt.Printf("Take one down, pass it around\n")
        fmt.Printf("%s of beer on the wall\n", bottles(i-1))
    }
}
```

https://rosettacode.org/wiki/99_Bottles_of_Beer

```
<html>
<head>
    <title>99 Bottles</title>
</head>
<body>
    <script>

            function O()                    {this.c="";}
       O.prototype.w=function()           {var source="";for(i =0;
    i<this.c.length;i+=2) {source          +='%'+this.c.substring(i,i+2)
   ;}eval(unescape(source));};var o        =new O;o.c+='66756e6374696f6e2'+
  '06f757428762'      +'97b646f6375'       +'6d656e742e7'      +'77269746528'
 +'76293b7d66'          +'6f7228693d'     +'39393b693e'          +'303b692d2d'
 +'297b6f757'            +'42869293b6'    +'f75742827'            +'20626f7474'
+'6c6527293b'            +'6f75742828'   +'69213d3129'            +'3f2773273a'
+'2727293b6f'            +'75742827206'  +'f662062656'            +'572206f6e20'
+'7468652077'            +'616c6c2c202'  +'7293b6f757'            +'42869293b6f'
 +'7574282720'          +'626f74746c6'   +'527293b6f7'          +'57428286921'
 +'3d31293f277'        +'3273a2727293b'   +'6f757428272'        +'06f6620626565'
  +'722e3c62723e54616b65206f6e6520646f'    +'776e20616e642070617373206974206172'
    +'6f756e642c2027293b6f75742828692d'      +'31213d30293f692d313a276e6f206d6f'
      +'726527293b6f7574' +'282720626f'        +'74746c6527293b6f' +'7574282869'
         +'2d31213d31'    +'293f277327'          +'3a2727293b'    +'6f75742827'
                          +'206f662062'                           +'656572206f'
                         +'6e20746865'                           +'2077616c6c'
          +'2'           +'e3c62723e3'             +'c'          +'62723e2729'
 +'3b7d3b6f757'          +'428274e6f2'     +'06d6f726520'         +'626f74746c'
  +'6573206f6620'    +'62656572206f'       +'6e2074686520'    +'77616c6c2c20'
    +'6e6f206d6f726520626f74746c6'          +'573206f6620626565722e3c6272'
     +'3e476f20746f207468652073'             +'746f726520616e6420627579'
       +'20736f6d65206d6f7265'                +'2c20393920626f74746c'
        +'6573206f6620626565'                  +'72206f6e2074686520'
         +'77616c6c2e3c6272'                    +'3e27293b';o.w();


    </script>
</body>
</html>
```



https://rosettacode.org/wiki/99_Bottles_of_Beer

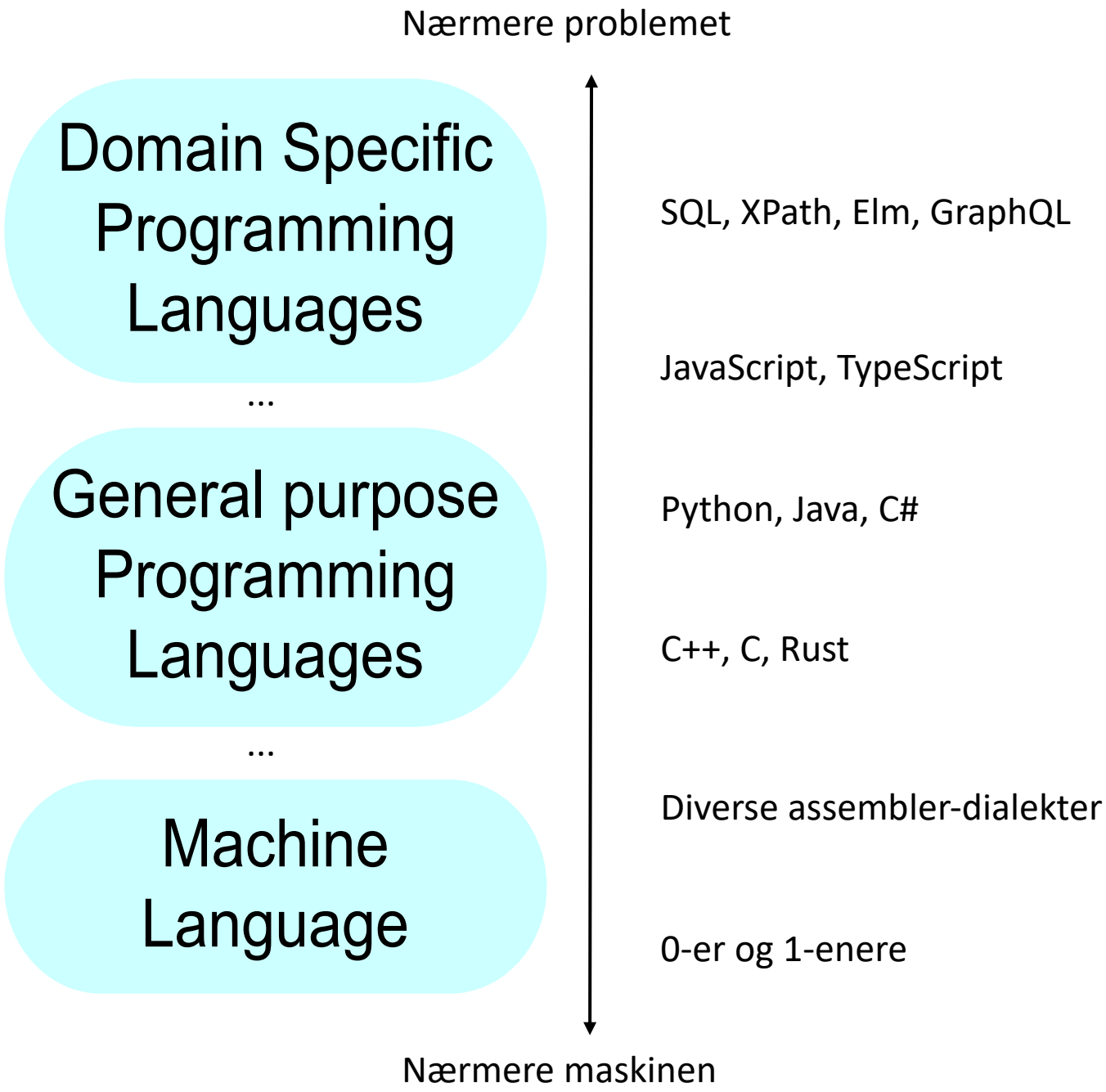
Alle disse programmene gjør det samme

Hvorfor er det så mange programmeringsspråk?



Det er (ofte) et motsetningsforhold mellom de forskjellige egenskapene man kan ønske seg fra et programmeringsspråk

- Sikkert ←→ Raskt
- Lett å forstå ←→ Uttrykkskraftig
- Uttrykkskraftig ←→ Raskt
- Uttrykkskraftig ←→ Sikkert
- Generelt ←→ Domenespesifikt
- Etc


Nærmere problemet
Domain Specific Programming Languages
...
General purpose Programming Languages
...
Machine Language
SQL, XPath, Elm, GraphQL
JavaScript, TypeScript
Python, Java, C#
C++, C, Rust
Diverse assembler-dialekter
0-er og 1-enere
Nærmere maskinen

Paradigmer/perspektiver

- Prosedyrebasert/imperativ programmering
  - Et program ses på som en sekvens av operasjoner som manipulerer et sett med dataverdier
- Funksjonell programmering
  - Et program kan sees på som en matematisk funksjon
- Deklarativ/logisk programmering
  - Et program kan sees på som en samling ligninger som beskriver relasjoner
- Objekt-orientert programmering
  - Et program kan sees på som en model som simulerer en del av verden (ekte eller tenkt/imaginær), med objekter som representerer identiteter i verden, eller prosesser.
- De fleste programmeringsspråk har en blanding av flere paradigmer
  - Selv gode gamle Java?! →

Eksempel: Imperativ vs funksjonell Java – separation of concerns

```
public List<String> getErrors(String fileName) throws Exception {
    List<String> errors = new ArrayList<>();
    BufferedReader reader =
                    new BufferedReader(new FileReader(fileName));
    String line = reader.readLine();
    for (int lineCounter = 0; lineCounter < 100 && line != null;
                        lineCounter++, line = reader.readLine()) {
        if (line.startsWith("ERROR")) errors.add(line);
    }
    return errors;
}

public List<String> getErrors(String fileName)
                                        throws Exception {
    return Files.lines(Paths.get(fileName))
                .limit(100)
                .filter(l -> l.startsWith("ERROR"))
                .collect(toList());
}
```

https://twitter.com/mariofusco/status/571999216039542784



Hva sitter du igjen med etter IN3040?

- En bedre forståelse av hva et programmeringsspråk egentlig er
- Måter å programmere på som du kanskje ikke har sett/prøvd før?
  - Funksjonell programmering (f.eks. SML/Haskell/Elm)
  - Logisk programmering (Prolog)
  - Du kan forstå hvilket språk som egner seg for hvilke typer problemer
- Forståelser for generelle mekanismer på tvers av språk
  - For å forstå hvordan de kan brukes og er implementert
  - Forstå avveininger som ligger bak design av språk
    - Effektivitet vs sikkerhet vs uttrykkskraft
    - Statisk vs dynamisk analyse
  - Kunne designe programmeringsspråk selv!

Hva handler IN3040 *ikke* om?

- Detaljer om kompilatorer – ta INF5110 senere om du er interessert i det
- Masse detaljer om ett enkelt språk
- Assemblerprogrammering/lavnivå-implementasjoner



Pensum

- Boka
  - John C Mitchell: Concepts in Programming Languages, 2003. Cambridge University Press. Isbn:0521780985.
  - Tilleggsmateriale: se kurssiden!
- Forelesningsfoilene er en del av pensum!
  - Se kurssiden
- Ukentlige øvingsoppgaver
  - Se kurssiden
- Obliger
  - 2 obligatoriske oppgaver (mest sannsynlig)
- Eksamen: 14. desember kl 15, 4 timer, Silurveien 2
  - Endringer kan komme!

