Kursintroduktion

Data- och informationsvetenskap: Objektorienterad programmering och modellering för IA

DA361A 7,5hp LP1

Johan Holmberg 2019-09-02

MALMÖ UNIVERSITET

Dagens agenda

- Vem håller i kursen?
- Upprop
- Kursplanen
  - Kursens syfte
  - Innehåll
  - Kursplanering
  - Kurslitteratur och -material
  - Examination och bedömningsformer
- Frågor kring kursen?
- Introduktion till kursinnehållet

MALMÖ UNIVERSITET

Kursens lärare

Anton Tibblin

anton.tibblin@mau.se

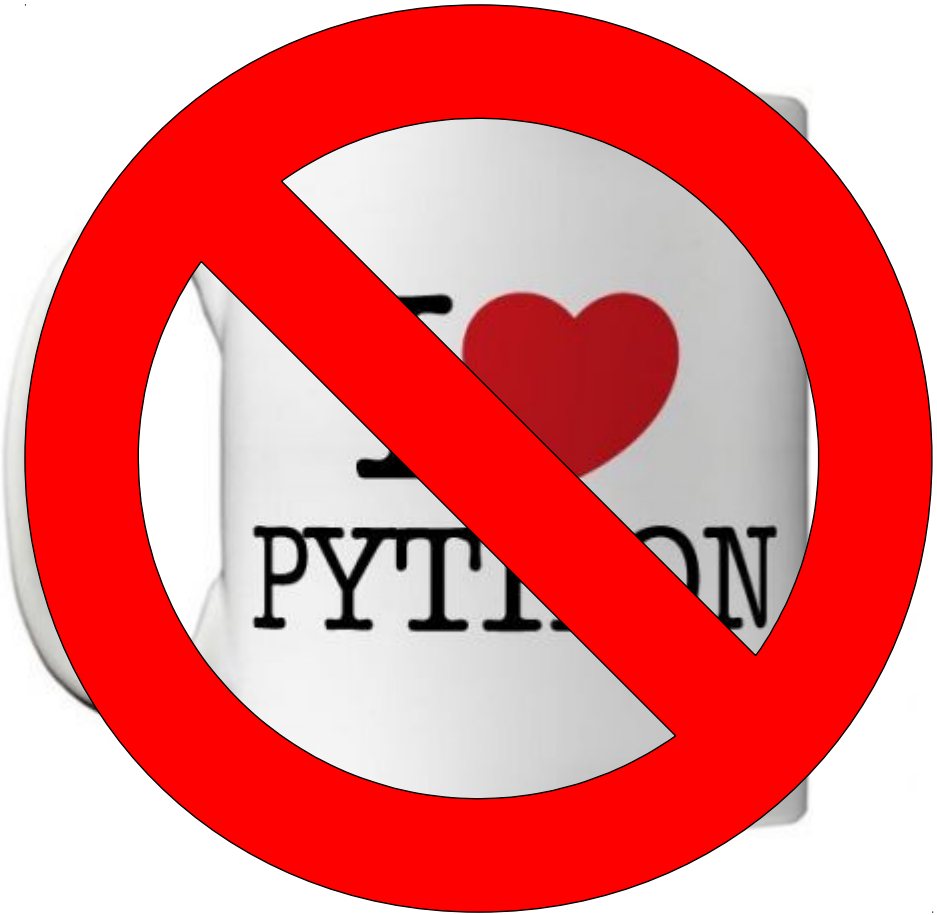
Johan Holmberg

Johan.holmberg@mau.se

MALMÖ UNIVERSITET

Vad är detta?

MALMÖ UNIVERSITET



MALMÖ UNIVERSITET

Detta är inte en kurs om Python

Det är en kurs som bara råkar använda Python...

MALMÖ UNIVERSITET

Hur jobbar du idag?

I projekt, förstås, men hur?

MALMÖ UNIVERSITET

Cowboy-kodaren

Skriver kod. That’s it.

En “fri själ”, men inte så bra:

- Ofta orutinerad
- Ostrukturerad
- Saknar kvalitetstänk
- Inte hållbar

Hur bör du jobba egentligen?

I projekt, förstås, men bättre än idag!

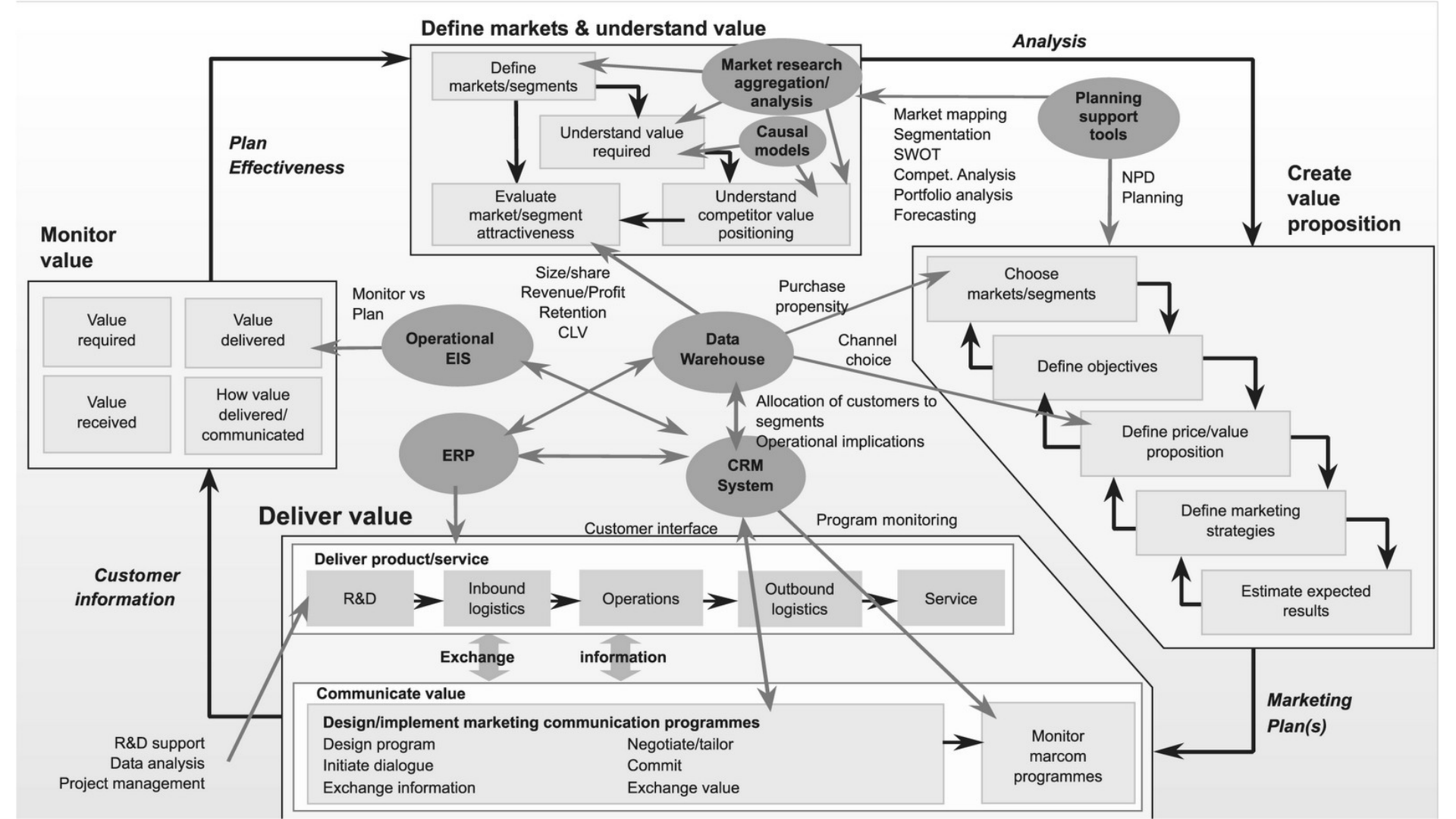
MALMÖ UNIVERSITET

Utvecklaren

Skapar lösningar. Utvecklar.

En ingenjörstyp:

- Metodisk och strukturerad
- Grupporienterad
- Kvalitetstänkande
- Hållbar


Define markets & understand value
Define markets/segments
Market research aggregation/ analysis
Understand value required
Causal models
Evaluate market/segment attractiveness
Understand competitor value positioning
Analysis
Market mapping
Segmentation
SWOT
Compet. Analysis
Portfolio analysis
Forecasting
Planning support tools
NPD
Planning
Create value proposition
Choose markets/segments
Define objectives
Define price/value proposition
Define marketing strategies
Estimate expected results
Plan Effectiveness
Monitor value
Value required
Value delivered
Value received
How value delivered/ communicated
Monitor vs Plan
Operational EIS
Size/share
Revenue/Profit
Retention
CLV
Data Warehouse
Purchase propensity
Channel choice
Allocation of customers to segments
Operational implications
ERP
CRM System
Deliver value
Customer interface
Program monitoring
Deliver product/service
R&D
Inbound logistics
Operations
Outbound logistics
Service
Exchange
information
Communicate value
Design/implement marketing communication programmes
Design program
Initiate dialogue
Exchange information
Negotiate/tailor
Commit
Exchange value
Monitor marcom programmes
Marketing Plan(s)
Customer information
R&D support
Data analysis
Project management


Solitaire
Game Help
Score: 72 Time: 49

Kursplanen

MALMÖ UNIVERSITET

Kursens syfte

Kursens syfte är att studenten ska utveckla kunskaper och färdigheter inom **objektorienterad programutveckling och -design**. Därigenom ska studenten även vidareutveckla sina programmeringsförmågor.

MALMÖ UNIVERSITET

Innehåll

- Från strukturerad till objektorienterad programmering
- Design och analys med principer för objektorientering
- Objektorienterad programmering (OOP)
- Unified Modeling Language (UML)
- Objektorienterad systemanalys och design (OOAD)
- Objektorienterad programmering i Python

Objektorienterad systemanalys och design

“Object-oriented analysis and design (OOAD) is a popular technical approach for **analyzing, designing an application, system, or business** by applying the object-oriented paradigm and **visual modeling** throughout the development life cycles to foster better stakeholder communication and product quality.”

MALMÖ UNIVERSITET

Objektorienterad programmering

“Object-oriented programming (OOP) is a programming paradigm based on the concept of **objects**, which are data structures that contain **data**, in the form of fields, often known as **attributes**; and code, in the form of procedures, often known as **methods**.”

Kursplanering

Vecka	Datum	Tid	Vad
36	2/9 5/9	13 – 15 13 – 17	Kursintroduktion Föreläsning – Intro till UML, OOAD och OOP
37	9/9 11/9 12/9	13 – 15 08 – 12 13 – 17	Föreläsning – Intro till UML, OOAD och OOP Föreläsning/Labb – Python, återblick Labb – UML & OOAD
38	16/9 19/9	13 – 15 13 – 17	Föreläsning – Intro till UML, OOAD och OOP Labb – Grunder i OOP
39	23/9 26/9	15 – 17 13 – 17	Föreläsning – OOP i Python Labb – OOP i Python
40	30/9 3/10 6/10	13 – 15 13 – 17	Föreläsning – OOP i Python Labb – OOP i Python ***Deadline: inlämningsuppgift 1***

MALMÖ UNIVERSITET

Kursplanering

Vecka	Datum	Tid	Vad
41	7/10 10/10	13 – 15 13 – 17	Kursintroduktion Föreläsning – Intro till UML, OOAD och OOP
42	14/10 17/10	15 – 17 13 – 17	Föreläsning/Labb – Mer OOP i Python Föreläsning/Labb – Mönster och datastrukturer
43	21/10 24/10	13 – 17 13 – 17	Föreläsning/Labb – Mönster och datastrukturer Workshop – Case
44	31/10 1/11	13 – 17	Workshop – Case ***Deadline: Inlämningsuppgift 2***
45	6/11 7/11	08 – 13 13 – 17	Salstentamen: ordinarie tillfälle Extratillfälle

MALMÖ UNIVERSITET

Kurslitteratur

- Think Python (O'Reilly)

  ISBN: 1491939362

  Finns gratis här: http://greenteapress.com/wp/think-python-2e/

- Object-Oriented Systems Analysis and Design Using UML (2010)

  ISBN: 9780077125363

- Problem Solving with Algorithms and Data Structures Using Python

  ISBN: 9781590282571

  Finns gratis här: http://pythonbooks.revolunet.com/

MALMÖ UNIVERSITET

Kursmaterial

- Canvas - inlämningar, resultat, meddelanden
  - https://canvas.mau.se
- Mau Webb - all annan information
  - http://da361a.ia-mau.se/
- Kursplan
  - http://edu.mah.se/sv/Course/DA361A?v=1#Syllabus

MALMÖ UNIVERSITET

Examination

Inlämningsuppgifter, 2 st, 3,5 hp

U – G

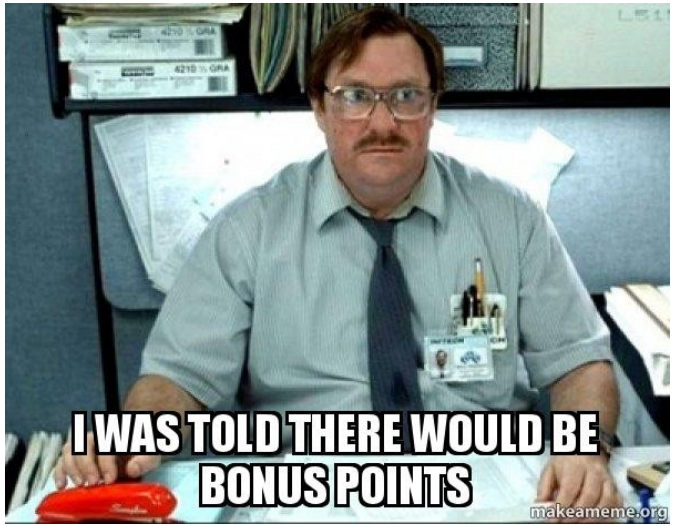
Skriftlig tentamen, 4 hp

U – VG

MALMÖ UNIVERSITET

Extra poäng till tentamen

- För varje laboration som du aktivt deltar i, kan du tillgodoräkna dig **0,5 p** på **ordinarie tentamen**
- Om du deltar aktivt i alla laborationer får du totalt **4 p** som du kan tillgodoräkna dig på **ordinarie tentamen**



MALMÖ UNIVERSITET

Bedömningformer

- Betygsskala U, G, VG
- För betyget godkänd krävs att samtliga inlämningsuppgifter blivit godkända
- För betyget väl godkänd krävs ett VG på tentan

MALMÖ UNIVERSITET

FUSK!

Var noga med att följa uppgiftsanvisningarna vad gäller samarbete. Otillåtet samarbete och delande av lösningar klassas som fusk och leder till tråkiga repressalier. Vi lärare är ganska bra på att känna igen delade lösningar. Har du problem med en uppgift – prata med en lärare istället för att “låna” en kursares lösning.

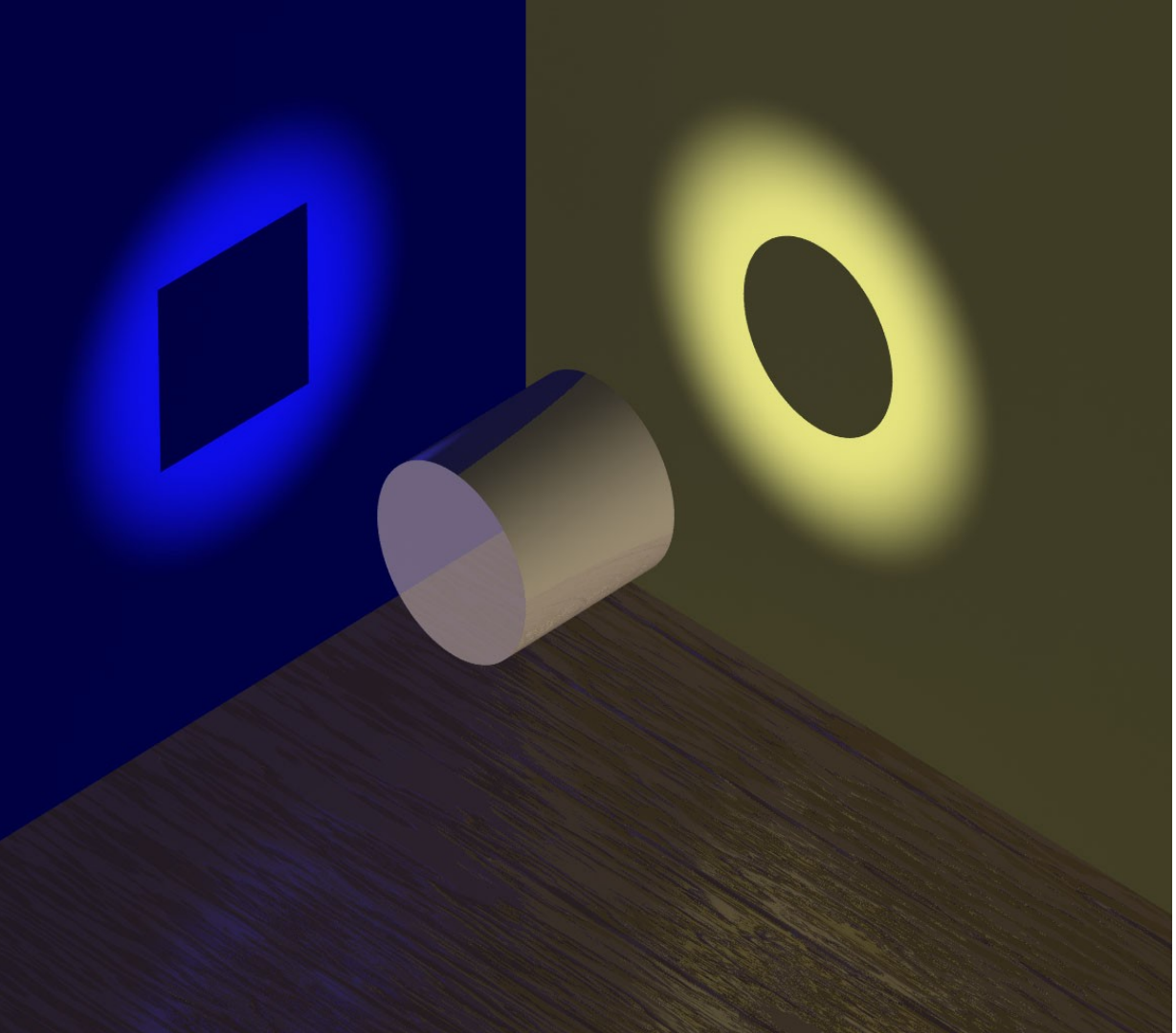
Förväntningar?

- Det här förväntar vi oss av er:
  - Läs kurslitteraturen
  - Var aktiva på föreläsningarna
  - Kom förberedda till övningar/laborationer
  - Gör uppgifterna i god tid – det går utmärkt att lämna in dem innan deadline!
  - Sträva mot ett VG – även om ni inte tar det i slutänden
- Vad förväntar ni er?

MALMÖ UNIVERSITET

Introduktion till objekt-orienterad programmering, analys och design

MALMÖ UNIVERSITET



Programmerings-paradigmer:

Olika sätt att beskriva världen

MALMÖ UNIVERSITET

Världen består av sekventiella händelser

I en sekventiell världsbild händer saker i steg, eller i en given ordning. Det som finns i världen påverkas av händelser, som kan namnges.

Om vi skriver ett program med denna världssyn arbetar vi framförallt med med **procedurer** eller **funktioner**.

Detta sätt att programmera kallas **procedurell programmering**.

MALMÖ UNIVERSITET

Världen är regler och sammanhang

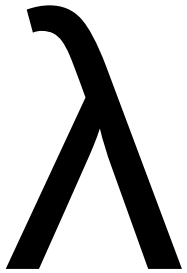
I en värld där allt ordnas enligt regler och sätts i sammahang är det (relativt) enkelt att ställa en fråga om tingens begivenheter och egenskaper.

En programmerare som arbetar efter detta synsätt **deklarerar logiska predikat** och **ställer sedan frågor** till programmeringsmiljön.

Den här paradigmen kallas **logisk programmering**.

MALMÖ UNIVERSITET

Världen kan beskrivas matematiskt



I en matematiskt färgad värld beskrivs skeenden och fenomen som matematiska formler. Här består allt av **funktioner**, vilka matas med **data** och **utvärderas**.

En programmerare som använder detta synsätt är mer intresserad av att beskriva för datorn **vad** som ska utföras, snarare än **hur** detta ska lösas.

Detta sätt att programmera kallas **funktionell programmering**.

MALMÖ UNIVERSITET

Världen består av objekt

Ett fjärde sätt att beskriva världen är att tolka allt som objekt – en stol, en fågel eller en tanke. Dessa objekt har alla olika **egenskaper** och kan **interagera med varandra**.

En programmerare som arbetar med med objekt spenderar mycket tid på att **modellera** och **analysera** innan någon kod skrivs.

Om du som programmerare ser världen som en samling av objekt har du antagit en **objekt-orienterad** världsbild.

MALMÖ UNIVERSITET

Vad är ett objekt?

Filosofen Charles S. Pierce definierade ett objekt som någonting som vi kan tänka eller tala om. I vårt dagliga språk motsvaras detta av våra subjektiv och pronomen.

Inom objekt-orienterad programmering är ett objekt en identifierbar samling data, knuten till funktioner som kan användas på denna data.

Man säger att objektet har **attribut** (data) och **metoder** (funktioner).

Bakgrund

I tidigare kurser har ni arbetat med variabler.

Vi kan exempelvis beskriva en kurs med hjälp av Python på följande vis:

```
university_location = "Malmö"
number_of_teachers = 2

teacher1 = {
"name": "Johan",
"age": 38,
"department": "IT"
}
teacher2 = {
"name": "Anton",
"age": 30,
"department": "DVMT"
}

print(teacher1)
print(teacher2)
print(university_location)
print(number_of_teachers)
```

MALMÖ UNIVERSITET

Ponera att vi vill upprepa detta för en annan kurs…

… för ett annat program…

… för ett annat universitet…

… i ett annat land…

…

MALMÖ UNIVERSITET

Övning!

1. Para ihop dig med personen som sitter bredvid dig.
2. Beskriv varandra i Python-kod. Använd vilka delar av språket du vill.
3. Beskriv någon annan – en förälder, en kursare, Batman – på samma sätt som ovan

Börjar det bli repetitivt?

MALMÖ UNIVERSITET

En potentiell lösning

```
# One way to describe a friend
friend1 = "My first friend is Romina. She is a female and she is 38 years old."
friend2 = "My second friend is Anton. He is a male. He is 30 years old."
```

Ponera att vi vill få ut åldern från de båda variablerna. Hur gör vi det?

3 …

2 …

1 …

^\s*(\w+)\s*\(\s*(\d+)\D+(\d+)\D+\)\s*$

MALMÖ UNIVERSITET

Ett lite bättre sätt

Med objekt i Python!

Ponera att vi vill få ut åldern från de båda variablerna. Hur gör vi det?

```
print(friend2['age'])
```

```
friend1 = {
"name": "Romina",
"gender": "female",
"age": 38
}

friend2 = {
"name": "Anton",
"gender": "male",
"age": 30
}
```

MALMÖ UNIVERSITET

Problem med den här metoden

- Koden blir snabbt **repetitiv**
- Vi måste **komma ihåg** väldigt många variabler
- **Att förstå** vårt program blir väldigt svårt
- **Att bygga ut** våra objekt med ett extra **attribut** skulle kräva mycket jobb

MALMÖ UNIVERSITET

Ännu bättre: objekt-orienterad programmering

Ett sätt att beskriva objekt **från verkliga världen** i kod är att beskriva dem som **instansierade objekt**!

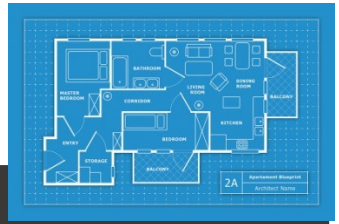
+

```
first_friend = Friend("Romina", "female", 38)
second_friend = Friend("Anton", "male", 30)
```

Klass = Mall

Objekt = Instanser av mallen

MALMÖ UNIVERSITET




```python
class Friend(object):
'''Det här är mallen som beskriver en vän'''

def __init__(self, friend_name, friend_gender, friend_phd):
'''Den här metoden anropas när en ny vän skapas'''
self.name = friend_name
self.gender = friend_gender
self.phd = friend_phd

def __str__(self):
'''Den här metoden anropas när en vän skrivs ut'''
return "This is {n} and their gender is {g}.".format(n=self.name, g=self.gender)
```

MALMÖ UNIVERSITET

Class

Definition of objects that share structure, properties and behaviours.

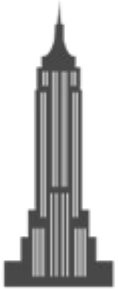
Building
class

Dog
class

Computer
class

Instance

Concrete object, created from a certain class.



Empire State
instance of Building

Lassie
instance of Dog

Your computer
instance of Computer

MALMÖ UNIVERSITET