Data- och informationsvetenskap: Objektorienterad programmering och modellering för IA

DA361A 7,5hp LP1

Lärare i kursen

Anton Tibblin

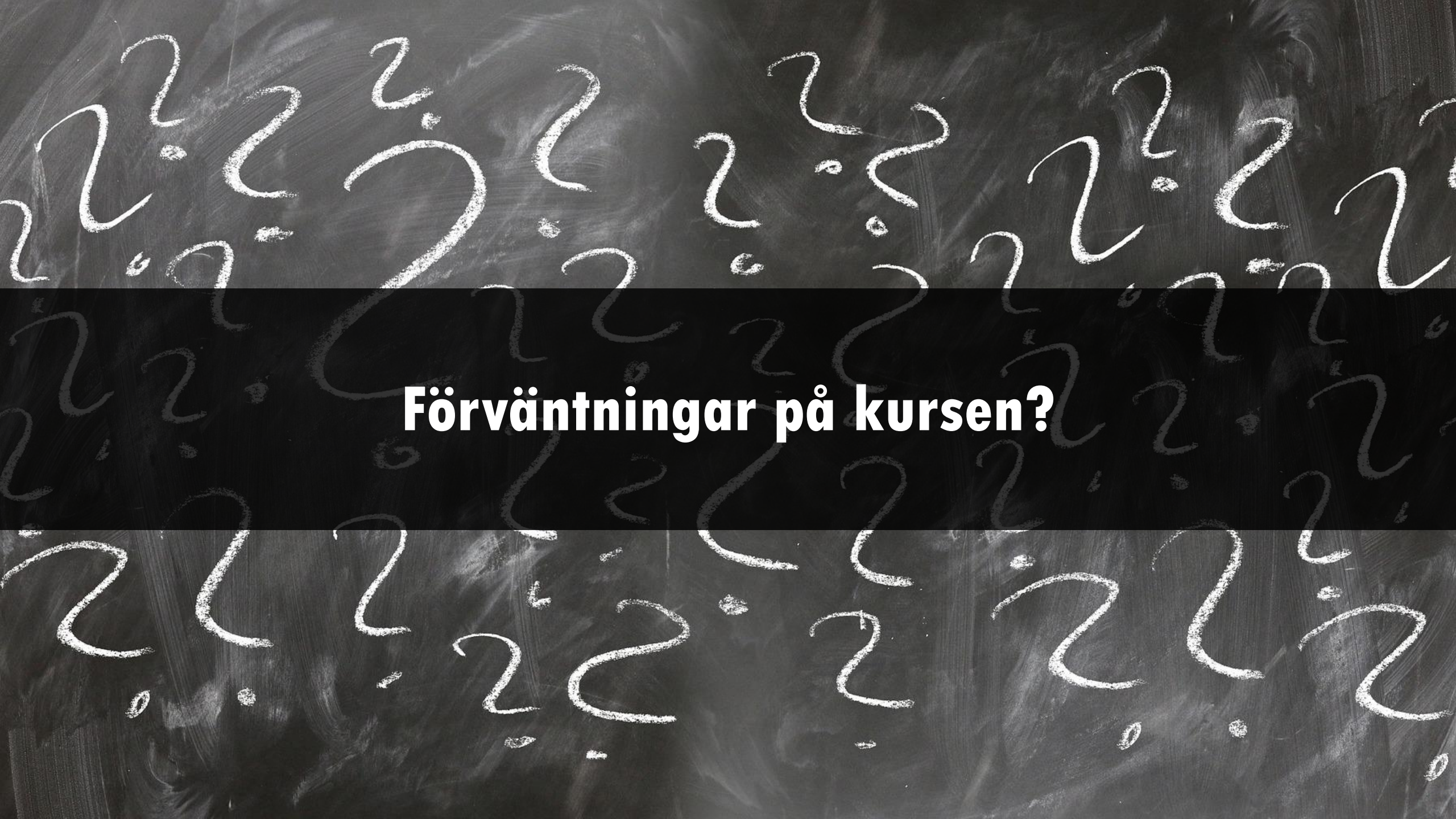
anton.tibblin@mau.se

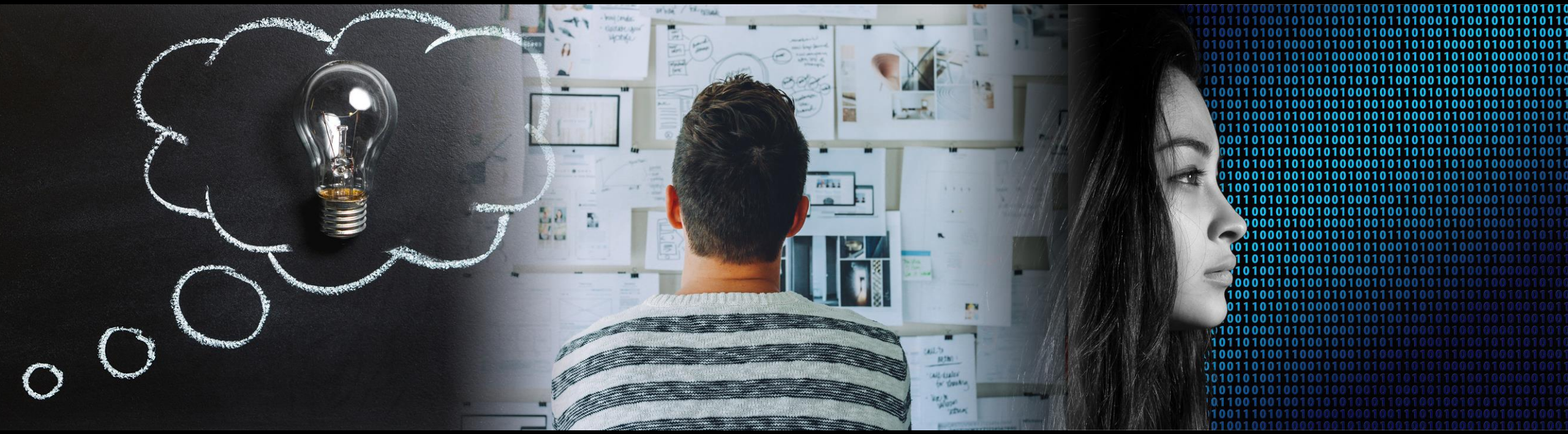
Aleksander Fabijan

aleksander.fabijan@mau.se

Närvaro!

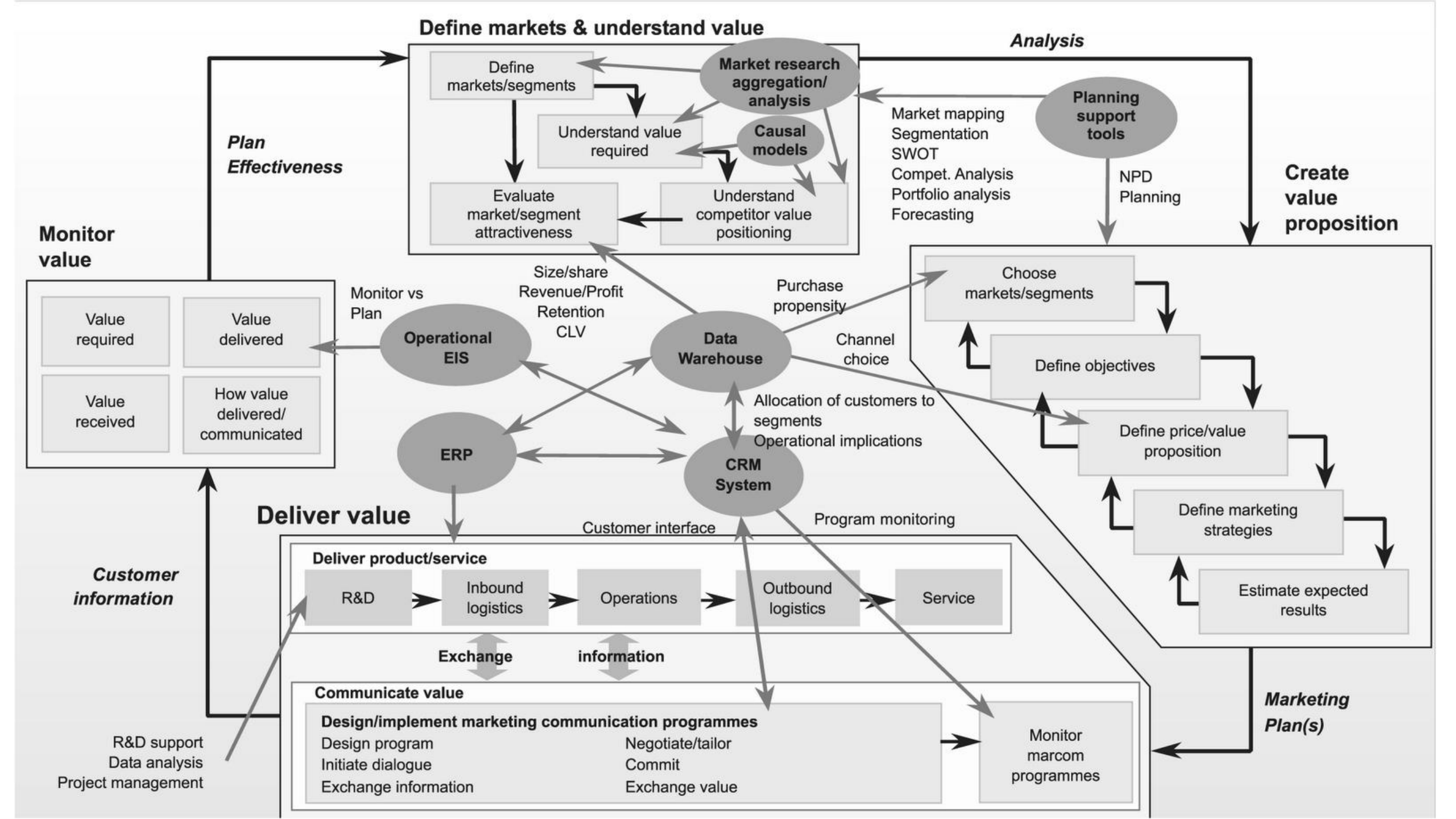

I
PYTHON


Förväntningar på kursen?



Hur går ni till väga idag?

När ni jobbar med era projekt?


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
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
Purchase propensity
Data Warehouse
Channel choice
Choose markets/segments
Define objectives
Define price/value proposition
Define marketing strategies
Estimate expected results
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
Customer information
R&D support
Data analysis
Project management
Marketing Plan(s)

Kursplanen

Kursens syfte

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

Innehåll

- Från strukturerad till objektorienterad programmering
- Design och analys med principer för objektorientering
- Objektorientering i det aktuella programspråket (Python)
- Unified Modeling Language (UML)

Innehåll

- Objektorienterad systemanalys och design (OOSAD)
- Objektorienterad programmering (OOP)

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."

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**."

Kursmaterial

- Canvas- inlämningar, resultat, meddelande
  - *https://mau.instructure.com/*
- Mah Webb - all annan information
  - *http://da361a.ia-mau.se/*
- Kursplan
  - *http://edu.mah.se/sv/Course/DA361A?v=1#Syllabus*

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
- http://pythonbooks.revolunet.com/

Schema

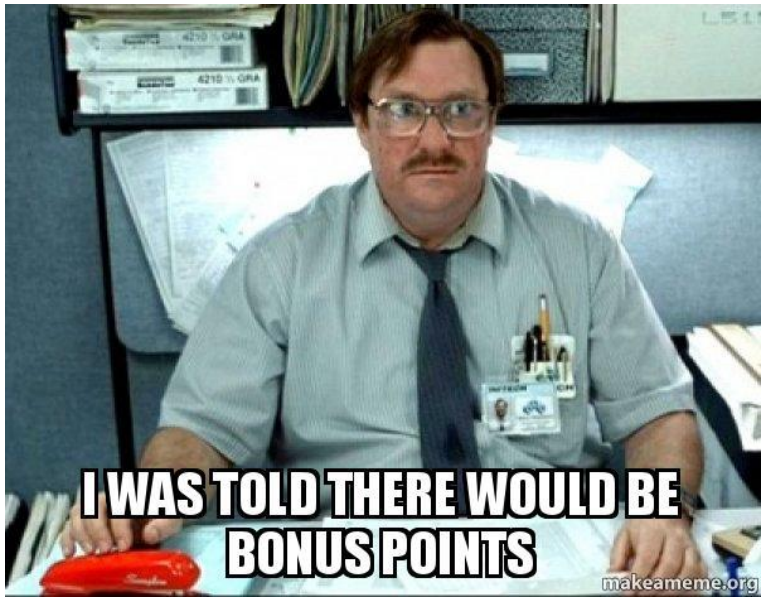
Vecka	Moment
36	Kursintroduktion + Föreläsning + Labb
37	Föreläsning*2 + Labb
38	Föreläsning + Labb
39	Föreläsning + Labb
40	Föreläsning + Labb
41	Workshop + Labb
42	Mer information kommer…
43	Extra föreläsning + labb
44	Extra labb
45	Tenta + extra labb

Bedömningsformer

- 2st Inlämningsuppgifter, 3.5 hp, U-G
- Tentamen, 4 hp, U-VG

Extra poäng till tentamen

- För varje laboration som man aktivt deltar i, kan man tillgodoräkna sig 0.5p till ordinarie tentamen
- Om man deltar aktivt i alla laborationer får man totalt 3p som man kan tillgodoräkna sig på ordinarie tentamen



Läshänvisningar

- Think Python
  - kap. 2-3 + 10-11 (repetition)
  - kap. 15-18 (OOP)

da361a.ia-mau.se

Frågor?

OOP Intro to Course

Aleksander Fabijan

Background

- In previous courses we learned about variables. For example, if you want to describe this course with a few variables in Python you could say:

```
city_of_course = "Malmö"
number_of_teachers = 2

teacher1 = {
"name": "Aleksander",
"age": 28,
"hasPhD": "Yes"
}
teacher2 = {
"name": "Anton",
"age": 28,
"hasPhD": "No"
}

print(teacher1)
print(teacher2)
print(city_of_course)
print(number_of_teachers)
```

- Now imagine we would want to describe another course like this…
  - Another program…
    - Another university…
      - …

Exercise 1

- Describe a friend of yours in Python code. Use any knowledge of python that you have previously obtained!
- Now **describe another friend** of yours following the same pattern.
- Do you see how much the code is starting to repeat itself?

One Solution

```
# One way to describe a friend
friend1 = "My first friend is Helena. She is a female and she is 30 years old."
friend2 = "My second friend is Anton. He is a male. He is nearly 29 years old."
```

Imagine I ask you to extract the age from the two variables. How would you do that?

…

…

…

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

Another Solution

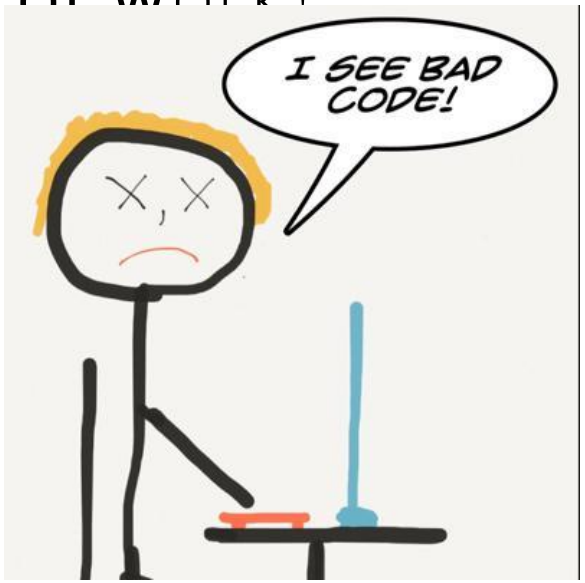
```
friend1 = {
"name": "Helena",
"age": 30
}

friend2 = {
"name": "Anton",
"age": 28
}
```

Imagine I ask you to extract the age from the two variables. How would you do that?

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

Challenge to go down this path…

- Code would start to **repeat itself** a lot,
- We would have to **remember the names of so many variables**!
- **Understanding our program** by others would be hard!
- Giving **one extra attribute** would require a **lot of work**!



Better: Object Oriented Programming

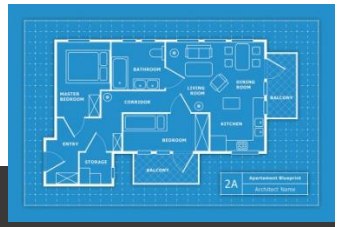
- One way to describe objects from real life in programming is to code them as objects!

+

```
first_friend = Friend("Helena", "female", True)
second_friend = Friend("Anton", "male", False)
```

Blueprint = Class

Objects = Instances from blueprint



```
class Friend(object):
'''This is a blueprint for a Friend'''

def __init__(self, friend_name, friend_gender, friend_phd):
'''This method is called when a new friend needs to be created'''
self.name = friend_name
self.gender = friend_gender
self.phd = friend_phd

def __str__(self):
'''This method is called when someone prints the friend'''
return "This is {n} and their gender is {g}.".format(n=self.name, g=self.gender)
```