# IN1000

Uke 11

# Plan for dagen

- Litt repetisjon / oppvarming med for-løkker
- Eksamensoppgaver
- Fortsette på studentsystemet / Game of Life / trix

# For-løkker på mengder

```
transportmidler = {"buss", "båt", "sykkel", "bil"}

for transportmiddel in transportmidler:
    print(transportmiddel)
```

# For-løkker på lister

```
frukter = ["appelsin", "eple", "pære", "sitron"]

for frukt in frukter:
    print(frukt)
```

# For-løkker på ordbøker

```python
ages = {"Alice": 25, "Bob": 33}

for name in ages:
    print(name)
    print(ages[name])
```

# Hvordan iterere gjennom verdiene

```python
ages = {"Alice": 25, "Bob": 33}

for age in ages.values():
    print(age)
```

# For-løkker på range()

- Lag en for-løkke som printer ut alle tall fra 0 til og med 9

```
for i in range(10):
    print(i)
```

# For-løkker på strings

```
hilsen = "hallo"

for bokstav in hilsen:
    print(bokstav)
```

# Hvordan opprette et objekt

```
class Person:
    def __init__(self, navn, alder):
        self._navn = navn
        self._alder = alder

kalle = Person("Kalle", 55)
```

# Hva er feil her?

```
class Person:
    def __init__(self, navn, alder):
        self._navn = navn
        self._alder = alder

    def hent_navn(self):
        return self._navn

kalle = Person("Kalle", 55)
hent_navn()
```

# En *metode* må alltid kalles på av et objekt!

```python
class Person:
    def __init__(self, navn, alder):
        self._navn = navn
        self._alder = alder

    def hent_navn(self):
        return self._navn

kalle = Person("Kalle", 55)
kalle.hent_navn()
```

# Oppgave 2a, prøveeksamen 2022

```python
class Klasse:
    def __init__(self, tall):
        self._tall = tall

    def sett_tall(self, tall):
        self._tall = tall

    def skriv_tall(self):
        print(self._tall)

x = Klasse(3)
y = Klasse(4)
z = x
u = Klasse(5)
u = z
z.sett_tall(2)
u.skriv_tall()
```

Hva skrives ut i terminalen?

# Oppgave 3b, prøveeksamen 2022

Skriv en funksjon tell_grader(fag, bsc, msc) som tar inn tre parametre av datatypen tekst (str) og sjekker om en Bsc-grad og/eller en Msc-grad er innen et bestemt fag. Dersom parameteren fag har samme verdi som både parametrene bsc og msc, skal funksjonen returnere 2. Dersom parameteren fag har samme verdi som én av parametrene bsc eller msc skal funksjonen returnere 1. Dersom parameteren fag hverken er lik verdien til parametrene bsc eller msc, skal funksjonen returnere 0.

For eksempel skal kallet *tell_grader("informatikk", "informatikk", "informatikk")* returnere 2, mens kallet *tell_grader("historie", "informatikk", "informatikk")* skal returnere 0.

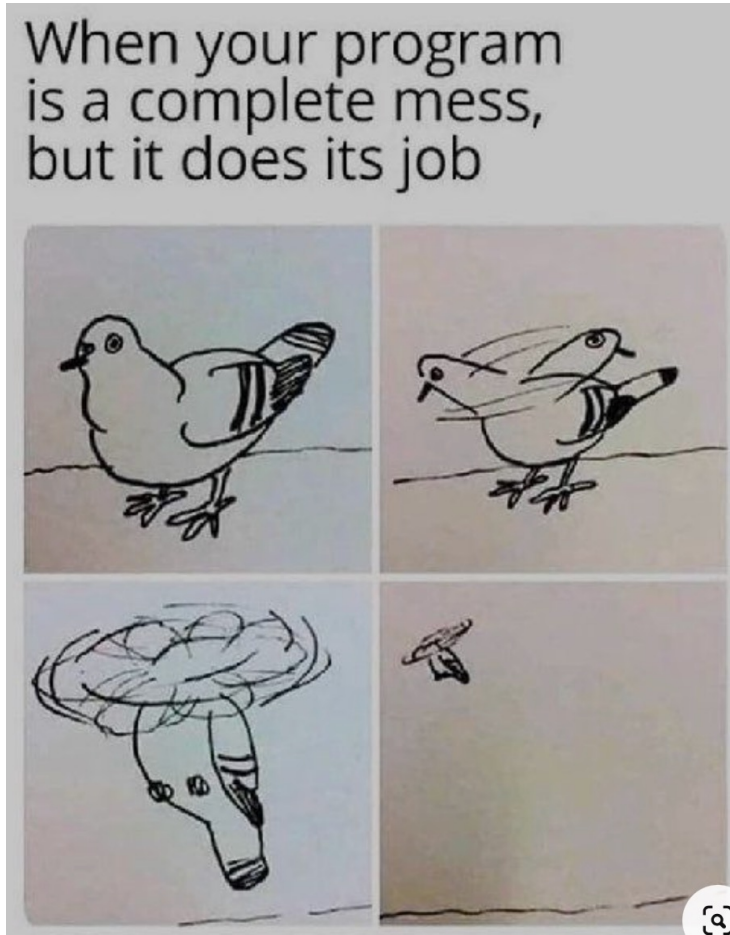
**Skriv ditt svar her**

# Fortsettelse på Studentsystem

- Vi har allerede laget Fag-objekter som holder oversikt over sine Studenter, og Student-objekter som holder oversikt over sine Fag. Nå skal vi lage en klasse “Studentsystem” som skal knytte alle fag sammen og holde oversikt over alle studenter og alle fag, samt å tilby en del operasjoner for brukeren.

Programmet skal tilby følgende operasjoner:

- Legge til ny student.
- Legge til nytt fag.
- Legge til et nytt fag til en student/legge til en ny student til et fag.
- Skrive ut alle studenter som tar et fag.
- Skrive ut alle fag en student tar.
- Finne ut hvilket fag som blir tatt av flest studenter.
- Finne ut hvilken student som tar flest fag.
- Fjerne en student fra et fag/fjerne et fag fra en student.
- Avslutt program.



# Fortsettelse på Studentsystem

- Programmet skal lese inn fra en gitt fil; emnestudenter.txt. Filen er på følgende format(forkortet versjon):

```
*MAT1001
Emilie
Jonathan
Henrik
Mari
Esben
Gard
Fridtjof
*MAT1100
Mathias
Ole Petter
Elsie
Andreas
Martin
```

# Fortsettelse på Studentsystem

**Oppgave 1**

- 1.1 Lag en klasse “Studentsystem”. Skriv konstruktøren for Studentsystem, denne skal ved opprettelsen ikke ta inn noen parametre, men skal lage en tom liste som skal holde på studenter og en tom liste som skal holde på fag.

- 1.2 Lag en metode i Studentsystem som håndterer innlesning fra fil, hvor filnavnet sendes med som parameter. Alle studenter skal legges inn i listen over Studenter og alle Fag inn i listen over Fag. Pass på så du ikke legger til duplikater av studenter (anta at alle studenter har unikt navn). Her kan du ha bruk for metoden du skriver i 1.3.
  - *Merk: studentene må i tillegg meldes opp til faget de ligger under (legges til i Fagets liste over studenter) og studentene må legge til faget i sin oversikt/liste over fag.*

- 1.3 Lag en metode i Studentsystem som finner og **returnerer** en Student basert på navn. Metoden skal returnere None dersom studenten ikke finnes i listen.
  *Merk: her må du bruke Student-objektenes “hent_student_navn()-metode.*

- 1.4 Lag en metode i Studentsystem som finner og **returnerer** et Fag basert på navn/fagkode.

# Fortsettelse på Studentsystem

**Oppgave 2**

- 2.1 Lag en metode som ber om et studentnavn (input fra bruker) og skriver ut alle fagene som den aktuelle studenten tar. Dersom studenten ikke finnes skal en beskjed skrives ut til brukeren. *Hint: Bruk metoden du skrev i 1.3 samt Student sin metode "skriv_fag_paa_student()"*

- 2.2 Lag en metode som ber om et fagnavn (input fra bruker) og skriver ut alle studentene som tar det aktuelle faget. Dersom faget ikke finnes skal en beskjed skrives ut til brukeren.

- 2.3 Lag en metode som finner ut hvilken student som tar flest fag. Resultatet kan skrives ut i terminalen. *Hint: Bruk studentenes hent_antall_fag()-metode.*

- 2.4 Lag en metode som finner ut hvilket fag som blir tatt av flest studenter. Resultatet skrives ut i terminalen.

# Fortsettelse på Studentsystem

**Oppgave 3**

- 3.1 Lag en metode som legger til en ny student i systemet, husk å sjekke om studenten allerede finnes.
- 3.2 Lag en metode som legger til et nytt fag i systemet, husk å sjekke om faget allerede finnes.