IN1000

Høsten 2025

Gruppe 6

- Bli kjent
- Viktig info
- Skrive og kjøre første programmer
- Oblig-quiz

Gruppetimene

Forventninger

Obligatoriske innleveringer

Les oppgavene og også hva du skal levere inn **nøye**

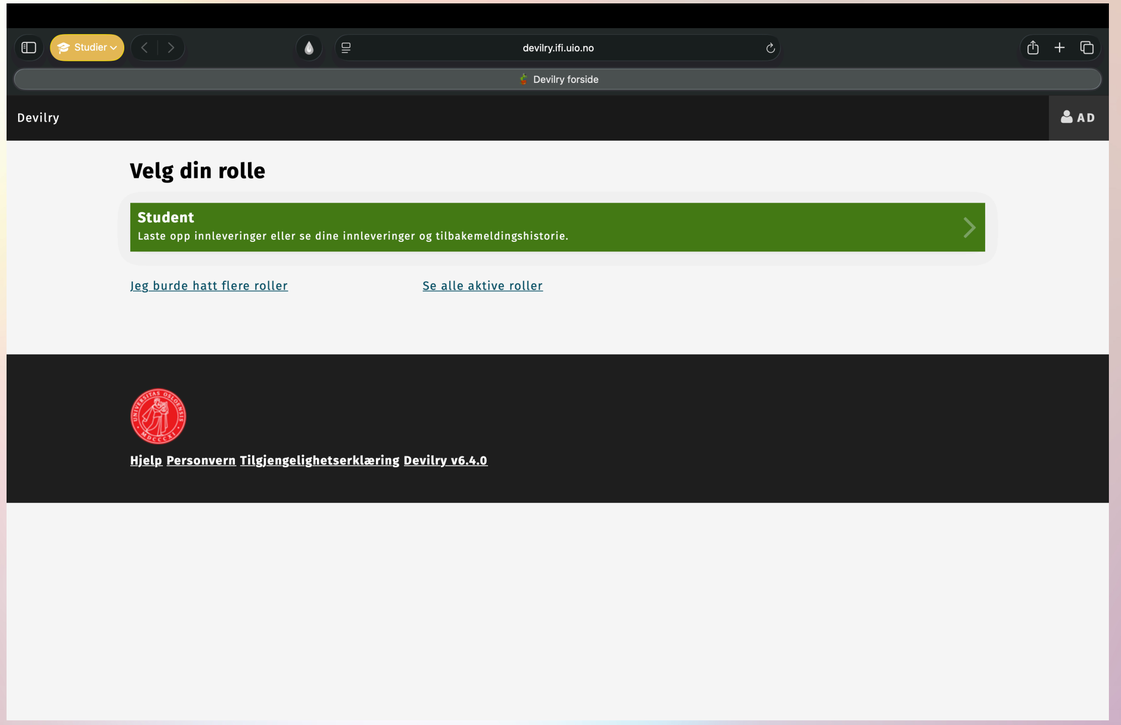
Leveres i Devilry

https://devilry.ifi.uio.no/

Du kan levere så mange ganger du vil fram til fristen har gått ut, det er det siste du leverer inn som vil bli rettet

Sjekk e-post jevnlig for tilbakemelding

Tre kalenderdager utsettelse kan du få ved å fylle ut skjema på semestersiden før fristen



Syk eller noe har skjedd

Studieadministrasjonen

studieinfo@ifi.uio.no

Kontakt her hvis du ikke får levert inn oblig ved f.eks. sykdom eller andre alvorlige ting som tar mer enn de tre egenmeldingsdagene. Ikke send inn til retteren din, gruppelæreren eller Siri/Geir Kjetil

Jobbe sammen, alene og AI

På obligene, og eksamen, så må du gjøre alt alene

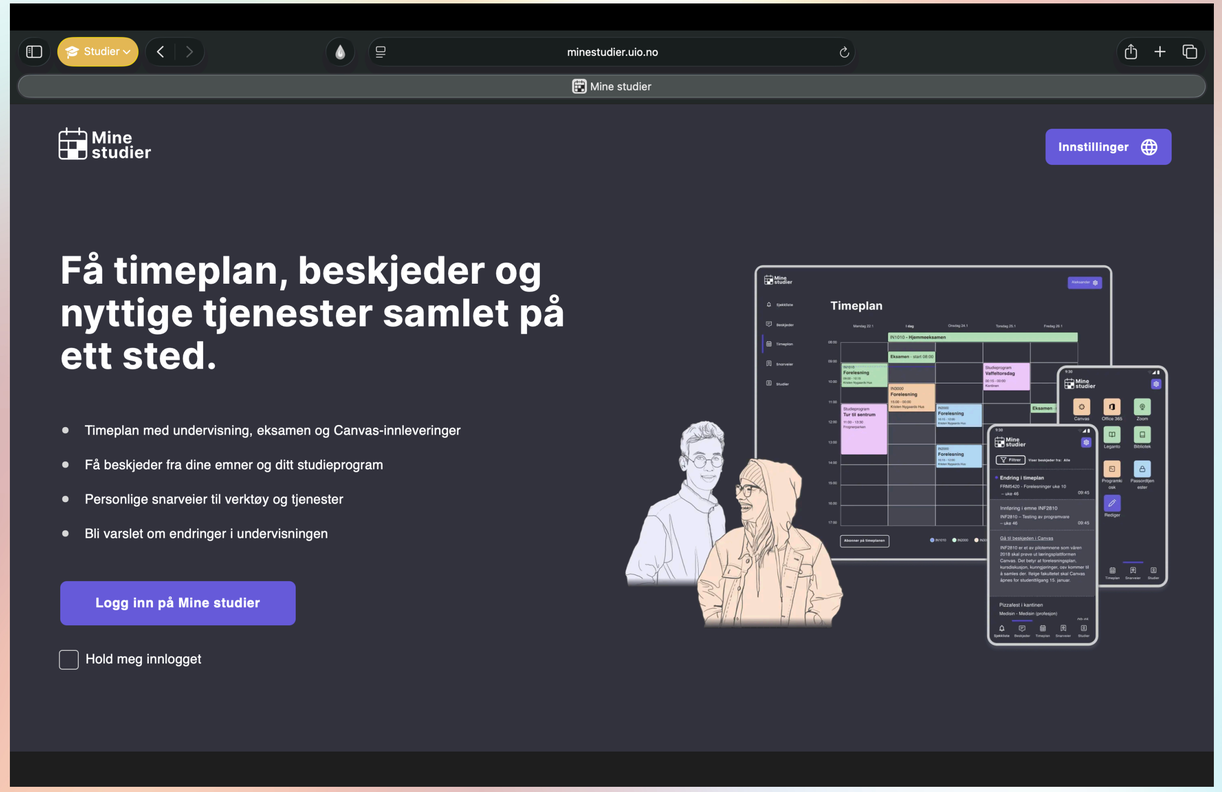
Programmering krever at man er både teknisk og kreativ

Samarbeid gjerne for å lære, og spør om hjelp

Men du lærer mest når du selv tenker ut løsninger, prøver og feiler



Mine studier



For å få oppdatert info om emnene dine, og kalenderen din kan du bruke Mine Studier, som er både en web-side og også som app

Discourse



Hvis du er usikker på noe, så kan du spørre på Discourse-forumet vi har for IN1000

https://astro-discourse.uio.no/c/in1000-25h/

Du kan poste både med brukernavn, eller som anonym

Som regel raskeste måte å få svar om ting som pensum og andre ting du lurer på

Men ikke post oblig-kode.

Trix

https://trix.ifi.uio.no/

Mange små oppgaver som som oftest også har et løsningsforslag

Hver uke så er det smart å løse alle oppgavene som tilhører denne uken

Pythontutor

https://pythontutor.com/python-compiler.html

For å visuelt se hvordan et program kjører, og med hvilke verdier og innhold variabler, objekter osv. har

Kan brukes til å debugge koden du har skrevet

Introduksjon til Python

Problemløsning

Ta vare på verdier med variabler

Skrive ut og lese inn fra terminalen

Bruke beslutninger for å bestemme programlinjer som kjøres

Forstå feilmeldinger og rette feil

Variabler

Variabler er "navn" som representerer en verdi

Det gir oss muligheten til å lagre og å bruke verdien senere

Python bruker "dynamisk" typing, og vi trenger ikke spesifisere hvilken type variablene skal ha

Variabelnavnene bør beskrive innholdet

I Python er det vanligst å bruke små bokstaver og _ (understrek) for mellomrom når vi lager variabelnavn

Variabler

```
alder = 28

fornavn = "Sara"

lengde = 128.0

bredde = 24.0

antall_appelsiner = 5

bok_tittel = "Ringenes Herre"
```

Input & Output

Nå bruker vi terminalen for gi informasjon til brukeren av programmet, og for at brukeren kan skrive noe inn

Vi bruker `input()` for input fra brukeren, og resulatatet kan vi lagre i en variabel

For å skrive noe ut så kan vi bruker `print()`

Input & Output

```
print("Velkommen til programmet")

print("Hva heter du? ")

fornavn = input()

print("Hvor gammel er du? ")

alder = input()

print("Hei " + fornavn + ", du er " + alder + " år")
```

Beslutninger

Vi kan bruke beslutninger når vi ønsker muligheten til å bestemme hvilken blokk med kode som skal kjøres

Nå skal vi se på `if`, `elif`, `else`

`If` bruker en logisk operasjon, og hvis den er sann, så vil blokken som tilhører bli kjørt.

Hvis vi har ulike operasjoner vi ønsker å teste, så kan vi også bruke `elif`.

Og for tilfellene som ikke blir sanne med de over, så kan vi lage en siste blokk ved å bruke `else`

Beslutninger

I Python så markerer vi en kodeblokk ved hjelp av innrykk. Man kan velge hvor mye, men man må bruke samme innrykk i hele programmet.

Mest vanlig så bruker man 2 eller 4 mellomrom for innrykket

Beslutninger

```
har_hest = input("Har du en hest? (j/n) ")

if har_hest == "j":

    hesten = input("Hva heter hesten? ")

    print("Du har en fin hest som heter " + hesten)

elif har_hest == "n":

    print("Du har ikke en hest")

else:

    print("Du må velge enten j eller n")
```

Feilmeldinger

Alle programmerere elsker bugs

I Python så er det viktig å kunne lese feilmeldingene slik at vi raskt kan finne ut hvor feilen er

Når en feil oppstår ved start eller under kjøring, så vil vi få en litt overveldende feilmelding i terminalen

Den består av en "traceback" og en feiltype og kort beskrivelse

Beslutninger

```
print(1/0)
```

```
$ python3 et-program-med-feil.py

Traceback (most recent call last):
  File "et-program-med-feil.py", line 1, in <module>
    print(1/0)
          ~^~
ZeroDivisionError: division by zero
```

Oppgaver til Problemløsning

Oppgave 1

Diskuter hvorfor det er nyttig å kunne problemløsning når vi programmerer. Del erfaringer dere har med problemløsning, både fra tidligere fag og ulike situasjoner i livet. Hvordan vil dere bruke problemløsning i IN1000 for å lære mest mulig?

Oppgave 2

1. Kjør programmet nedenfor og gi en detaljert forklaring av feilmeldingen.
2. Endre koden, slik at programmet kjører feilfritt. Skriv ned endringene steg for steg og diskuter hvorfor programmet kjører feilfritt etter endringene.
3. Lag en oppgave der dere bruker variabler og beslutninger. Legg en stegvis plan for hvordan dere vil løse oppgaven. Løs deretter oppgaven sammen.

```
busselskap = "Unibuss"
trikkeselskap = "Sporveien"
if busselskap == trikkeselskap:
    print("Samme selskap har både busser og trikker!")
    else:
        print("Buss- og trikkeselskapene er forskjellige!")
```

Oppgave 3

Lag et program med tre strenger. Brukeren skal få beskjed om hvilken av de tre vedkommende skriver inn. Hvis brukeren ikke skriver inn noen av de tre strengene, skal det gis beskjed om at ingen av dem ble skrevet inn.

Oppgave 4

Utvid oppgave 3, slik at brukeren får beskjed om hvilke to strenger vedkommende skrev inn. Dere skal løse utvidelsen på to forskjellige måter. Diskuter to mulige løsningsstrategier og skriv deretter kode for hver strategi. Hva kan være mulige fordeler eller ulemper ved å løse utvidelsen på den ene eller andre måten? Diskuter.

Oppgave 5

Nå skal dere lage et veldig forenklet pasientprogram. Det skal ta vare på en lege og en pasient. Hva slags informasjon programmet skal lagre om legen og pasienten velger dere selv. Brukeren skal først kunne skrive inn informasjon om pasienten, før informasjonen om pasienten skrives ut. Deretter skal brukeren skrive inn informasjon om legen, før informasjonen om legen skrives ut. Før programmet avslutter, skal det skrive ut at pasient (navn) er behandlet av lege (navn) for en diagnose dere velger.

Trix

Oppgave 01.07, 01.09, 01.11, 01.14, 01.18

evt. lag et program hvor en spiller kan gå øst, vest, nord, sør; og forskjellige ting skjer ettersom hvor han/hun går og gjør