GRUPPETIME
UKE 1

IN1000 GRUPPE 4 - 25.01.22

PLAN FOR GRUPPETIMEN

- Praktisk info + bli kjent
- Læringsmål og moduler
- Variabler og typer
- Kommentarer
- Print
- Input fra bruker
- if-setninger og boolske uttrykk
- Feilmeldinger

BLI KJENT!

Siri Sollerud - prosa
Kontakt: @sirisoll på Mattermost eller sirisoll@uio.no

Mentimeter: 2642 8394

Breakout rooms

→ Navn
→ Hvilken hobby har du lagt flest timer i?
→ Hva er din #1 feriedestinasjon?
→ Hvorfor vil du ta IN1000?
→ Har du programmert tidligere?

Enkleste måten å kontakte meg er å sende meg en melding på Mattermost! Søk etter @sirisoll

MATTERMOST

Vi har vår egen kanal:
Søk på “Gruppe 4”
Server link: “mattermost.uio.no” må oppgis på første innlogging

UiO-EPOST

Sjekk hver dag!
Videresend til privat email

MINESTUDIER

Kan legges inn i kalender på mobil - guide finnes her

SEMESTERSIDEN

Timeplan, obliger, beskjeder, emneressurser, eksamen etc.
Vi har egen gruppeside

INNLEVERING OG OPPGAVER

Codegrade

- Kan levere flere ganger
- Kun nyeste filer som blir rettet
- Husk riktig filtype!
- Kan få samretting (rettes muntlig med retter/gruppelærer)

Trix

- Oppgaver basert på uke, tema etc.
- Godt hjelpemiddel hvis du står fast i en oblig
- Flott til samarbeid!

UKENS GANG

forberedelse

før tirsdag

Studer lysark/se på videoene og forsøk på noen Trix oppgaver, slik at du er kjent med stoffet før du møter. Pensum for uka finner du i oversikten

gruppetime

tirsdag

10:15-12:00

Ukens læringsmål gjennomgås med flere eksempler og praktisk info. Spør gjerne hvis du lurer på noe angående pensum!

lab

onsdag

12:15-14:00

Still spørsmål i plenum eller privat i breakout rooms. Her kan du få hjelp til trix oppgaver, pensum, obliger etc.

oblig

torsdag

Lever før 23:59 torsdag
Oblig 1-6 er poengbasert (krever 19/29 poeng totalt for godkjenning), mens oblig 7-8 er pass/fail. Leveres i Codegrade.

forelesning

fredag

12:15-14:00

Oppsummerende møte mellom studenter og faglærere, med vekt på refleksjon og repetisjon av det som var mest utfordrende med ukas stoff.

OBLIGER

1	2	3	4	5	6	7	8
POENGGIVENDE: 19/29 totalt for å bestå						BESTÅTT/ IKKE BESTÅTT	

- **1 - 6: ukentlig obliger → 3-6 poenggivende oppgaver**
  - Frist torsdager 23:59. Ingen utsettelser. Ett forsøk.
- **7 - 8: obligatorisk → utvikle større program i løpet av 2 uker**
  - Kan få utsettelse på 3 dager - send retter en e-mail før fristen.
  - Kan få nytt forsøk.
- **!NB PLAGIAT: ikke lov til å kopiere andres kode**
  → codegrade har plagiatkontroll

NB: Endringer kan komme ***Obliger levers i codegrade***	**KAN LEVERES FRA**	**SISTE FRIST**
1. innlevering (3 poeng)	**31. januar kl. 12.00**	**3. februar kl. 23.59**
2. innlevering (5 poeng):		10. februar 23.59
3. innlevering (5 poeng):		17. februar 23.59
4. innlevering (5 poeng):		24. februar 23.59
5. innlevering (5 poeng):		10. mars 23.59
6. innlevering (6 poeng):		17. mars 23.59

MODULER UKE 1-5

UKE	TEMA
1	**Innføring i python - deres første program!** - Et første Python-program - Variabler - Feilmeldinger - Input fra tastaturet - Beslutninger
2	**Dypere forståelse fra uke 1** - Datatyper - Evaluering av uttrykk - Kodeflyt - Prosedyrer
3	**Samlinger** - Objekter tilbyr tjenester - Samlinger og lister - Mengder (sets) - Ordbøker (dictionaries) - Nøstede samlinger

UKE	TEMA
4	**Løkker** - Løkker - Kombinere løkker med samlinger - For-løkker - Parametre i prosedyrer - Retur-verdier
5	**Filer** - Lese fra og skrive til filer - Eksempel på å kombinere data - Parameter-overføring og variabel-skop

FORVENTNINGER OG TIPS

- Mengdetrening med Trix er gull verdt og hjelper med å forstå obliger

  → Samarbeid gjerne med Trix og øv til du føler du mestrer det

- Ting bygger på hverandre: programmering krever innsats over tid

  → IN1000 er kjempeviktig for å forstå IN1010!

- Det er vanskelig å lære noe helt nytt!

  → Feilmeldinger og debugging er en del av prosessen

- Start tidlig med obliger: finn en god ukentlig flyt som passer deg
- Står du fast så spør meg eller en annen gruppelærer: vi er her for deg :)

SET UP YOUR WORKSPACE

1. Last ned **Python** og **Atom**
   NB!!! For de som bruker Windows, sørg for at "Add Python to Path" er avhuket under installeringen

2. Få i gang en **terminal**
   Linux: alt burde funke
   MacOS: det meste burde funke
   → Installer homebrew, og bruk det til å installere ting!
   Windows:
   → Bruk WSL (Windows subsystem for linux)
   → Eller Cygwin

→ Oppstartguiden & forkurs ressurser

JOBB MED TRIX OPPGAVER

Supert å hjelpe hverandre!

JOBB MED OBLIG 1

Pass på å ikke kopiere kode!

LINUX KOMMANDOER

pwd	print working directory
cd	change directory
mkdir	make directory
ls	list directory contents
touch	create empty file
gile	determine file type
cp	copy
rm	remove (pass på!)
cat	concatenate

python3 program.py

LÆRINGSMÅL UKE 1

- Kunne logge på en linux-tjener ved Ifi fra egen maskin, skrive, endre og kjøre et Python-program
- Programmering i Python:
  - Kunne **skrive ut til og lese inn fra terminalen**
  - Kunne ta vare på **verdier** med **variabler**
  - Kunne bruke **beslutninger** (if) for å avgjøre hvilke programlinjer som skal kjøres
  - Kunne lese en **feilmelding**
  - Kjenne til ulike verktøy for å **skrive og kjøre Python-programmer**

VARIABLER

- En variabel er et navn som representerer en verdi
- En variabel kan bare holde én verdi av gangen
- Man kan underveis i et program endre hvilken verdi en variabel representerer!

```
# Hva skrives ut?
pokemon = "Pikachu"
pokemon = "Charmander"
print(pokemon)
```

→ Charmander

TYPER

- Vi tilordner en verdi til en variabel
- Denne verdien kan være en tekst streng, et tall, et flyttall, etc.
- Altså, i python har vi forskjellige datatyper!

`2`	int - heltall
`2.2`	float - flyttall
`"2"`	str - tekst streng
`True/False`	boolean - sant eller usant

+++ finnes flere som vi skal se på senere i kurset

- Forrige lysark tilordnet vi verdien “Charmander” til variabelen pokemon

```
pokemon = "Charmander"
```

→ vi har dermed lagret en tekst streng (str) i variabelen pokemon

KODEFLYT

Koden kjøres i den rekkefølgen dere har skrevet den!

```
# Hva printes ut?
# Hva er verdien til alder etter følgende kode kjøres?
alder = 10
print("Alder er: ", alder)
alder = 5
```

→ 10
→ alder har verdi 5

Er du usikker på kodeflyten i et program? Prøv pythontutor!

PRINT

print() lar oss skrive ut til terminalen

```
# Skriv ut film, aar og sitat med print()
film = "The Fifth Element"
aar = 1997
sitat = "I am a meat popsicle"
```

PRINT EKSEMPEL

```
film = "The Fifth Element"
aar = 1997
sitat = "I am a meat popsicle"

print(film, "ble utgitt i", aar, "og et sitat fra filmen er:\n",sitat)

# Hvis print() er tom skrives en newline ut - det samme skjer med \n i strings
print()

# En annen løsning: f-string
print(f"{film} ble utgitt i {aar} og et sitat fra filmen er:\n{sitat}")
```

```
>>   The Fifth Element ble utgitt i 1997 og et sitat fra filmen er:
      I am a meat popsicle

     The Fifth Element ble utgitt i 1997 og et sitat fra filmen er:
     I am a meat popsicle
```

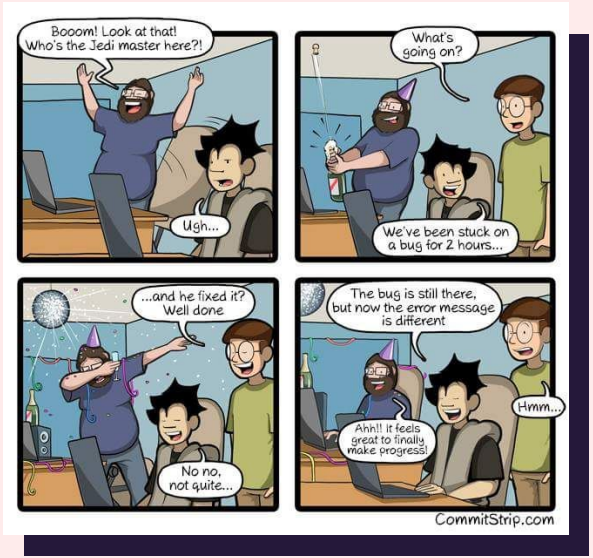
kommentarer.py

KOMMENTARER

```
# Dette er en komentar på en linje.
"""
Dette er en kommentar over flere linjer.
Kommentarer/dokumentasjon av kode er det man skriver i koden som ikke
leses/forsøkes tolkes av maskinen.

Det kan være nyttig å skrive litt kommentarer for å holde oversikt
over hva koden gjør.

Kommentarer gjør det enklere for andre å lese/forstå koden din
Det kan være viktig å forklare hvordan du har tolket en problemløsning
eller hvorfor du tok visse valg.
"""

print("Dette er det eneste som skrives ut!")
```

FEILMELDINGER

- Når noe går galt i et program får vi en feilmelding
- Feilmeldinger består av:
  - Hvor feilen lå (fil og linjenummer)
  - Hvilken type feil det var
  - Akkurat hva som var problemet
- feil.py



FEILMELDING EKSEMPLER

```
10 / 0
Traceback (most recent call last):
  File "feil.py", line 1, in <module>
    10 / 0
ZeroDivisionError: division by zero
```

```
4 + spam*3
Traceback (most recent call last):
  File "feil.py", line 1, in <module>
    4 + spam*3
NameError: name 'spam' is not defined
```

```
'2' + 2
Traceback (most recent call last):
  File "feil.py", line 1, in <module>
    '2' + 2
TypeError: can only concatenate str (not "int") to str
```

→ Nysgjerrig på mer? Se Python dokumentasjonen!

feil.py

PASS PÅ TYPEN!

```
'2' + 2
Traceback (most recent call last):
  File "feil.py", line 1, in <module>
    '2' + 2
TypeError: can only concatenate str (not "int") to str
```

Hvis ikke typene er like kan dette skape problemer fordi forskjellige typer har forskjellige egenskaper/regler.

Får å fikse feilmeldingen ovenfor må vi derfor passe på at typene er like.

```
svar = "2" + "2"        >> 22    → vi konkatenerer to strenger
svar = 2 + 2            >> 4     → vi adderer to integers
```

Vi kunne også ha gjort en typekonverterting

```
svar = "2" + str(2)     >> 22    → vi konverterer en int til str
svar = int("2") + 2     >> 4     → vi konverterer en str til int
```

Usikker på typen? Da kan du sjekke med type()

```
navn = "Siri"
print(type(navn))       >> <class 'str'>
```

ARITMETISKE OPERASJONER

```
1.  a = 4 + 5       >> 9
2.  a = 6 - 2       >> 4
3.  a = 4 * 2       >> 8
4.  a = 3/2         >> 1.5
5.  a = 11//2       >> 5
6.  a = 2**3        >> 8
7.  a = 3 % 2       >> 1
8.  a = 2 * 2 + 3   >> 7
```

5. → fordi // er heltallsdivisjon
6. → potens/eksponent $2^3 = 2 * 2 * 2 = 8$
7. → heter modulo: finner resten av hva som er igjen etter en divisjon
8. → følger renerekkefølgen

INPUT

- input() henter inputt fra bruker som lagres i variabelen som en streng
- Vi kan dermed lese inn hva brukeren skriver i terminalen med input()
- Husk å gi presise beskrivelser/promtps til brukeren

```
navn = input("Skriv inn navn: ")
print(navn)
```

→ Kan meldingen “Skriv inn navn” forvirre brukeren?

```
navn = input("Skriv inn fornavn etterfulgt av etternavn: ")
print(navn)
```

→ navn inneholder nå strengen “Siri Sollerud” feks.

BESLUTNINGER: IF-STATEMENTS

- If- statements sjekker booslke uttrykk
- Boolske uttrykk evaluerer til enten `True` eller `False`
  1. if boolsk uttrykk er `True`:
     2. utføres en kodeblokk
  3. deretter utføres resten av koden

```
if True:
    # Do this
elif True:
    # Do this
else:
    # Do this
# Continue
```

EKSEMPLER: IF-STATEMENTS

```python
# "Hade!" printes alltid
if(5 < 1):
    print("Hei på deg!")
else:
    print("Hade!")
```

```python
# "Hei på deg!" printes alltid
if(5 == 5):
    print("Hei på deg!")
else:
    print("Hade!")
```

BOOLSKE UTTRYKK

- Et boolsk uttrykk evaluerer til enten `True` eller `False`
- Alle disse evaluerer til en boolean:

```
1 != 2          >>True
1 < 1           >>False
1 >= 1          >>True
"a" == "a"      >>True
1 == "1"        >>False
```

!=	leses “er ikke lik”
<	leses “er mindre enn”
>=	leses “er større enn eller lik”
==	leses “er lik”

<, <=, >, >=, ==, != *heter comparison operators eller sammenligningsoperatorer på norsk*

hvis.py

INPUT/IF-STATEMENT

Hvorfor trenger vi typekonvertering her?

```
# Be brukeren om å oppgi alder
# Sjekk om brukeren er myndig (18 eller over)

alder = int(input("Hvor gammel er du?: "))
if alder >= 18:
    print("Du er myndig")
print("Da fortsetter vi videre!")
```

boolsk uttrykk

kodeblokk

tekst.py

DISKUTER: VERDIEN TIL TEKST?

```
# Hva er verdien til variabelen tekst
# etter at følgende kode er utført?

tall = 7
tekst = "a"

if tall > 10:
    tekst = tekst + "b"
elif tall < 5:
    tekst = tekst + "c"
else:
    tekst = tekst + "d"
print(tekst)
```

DISKUTER: HVA SKRIVES UT?

1.
```
if 5 < 10:
    print("Nr. 1")
elif 5 < 10:
    print("Nr. 2")
```

2. to if-sjekker
```
if 5 < 10:
    print("Nr. 1")
```
```
if 5 < 10:
    print("Nr. 2")
```

3.
```
if 5 < 10:
    print("Nr. 1")
else:
    print("Nr. 2")
```

4.
```
if 10 < 5:
    print("Nr. 1")
else:
    print("Nr. 2")
```

5.
```
if 10 < 5:
    print("Nr. 1")
elif 10 < 5:
    print("Nr. 2")
else:
    print("Nr. 3")
```

6. en if-sjekk
```
if 5 < 10:
    print("Nr. 1")
elif 2 < 5:
    print("Nr. 2")
else:
    print("Nr. 3")
```

Pass på at med en gang vi kommer til et boolsk uttrykk som evaluerer til True så hopper vi ut av if-sjekken!

AND

True	True	**True**
True	False	**False**
False	True	**False**
False	False	**False**

OR

True	True	**True**
True	False	**True**
False	True	**True**
False	False	**False**

NOT

True	**False**
False	**True**

→ Vi kan sette sammen boolske uttrykk med **and** og **or**

DISKUTER: TRUE/FALSE?

```
(5 == 1) and (5 > 20)
(5 == 1) and (5 > 1)
(5 == 5) and (5 > 1)
```

```
(5 == 1) or (5 > 20)
(5 == 1) or (5 > 1)
(5 == 5) or (5 > 1)
```

```
not("a" == "a")
not(1 != 2)
not(1 == "1")
```

DISKUTER: TRUE/FALSE?

```
x = 2

1 < x < 3       >>True

10 < x < 20     >>False

3 > x <= 2      >>True

2 == x < 4      >>True
```

JOBB SAMMEN: TRIX 01.14

Skriv et program der du ber om brukerinput på ett eller flere trivia-spørsmål. Hvis du ikke kommer på et spørsmål får du et gratis her: "Hva heter hovedstaden i Marokko?" (svaret er "Rabat").

Lagre det rette svaret i en tekststreng.

Skriv en if-test for å sjekke om brukeren har svart rett på spørsmålet. Hvis de har svart riktig skal programmet skrive ut "Helt rett!". Hvis ikke skal programmet skrive ut "Beklager, svaret var" og deretter det riktige svaret du har lagret.

Programmering er kreativt!

Det er som regel mange måter å løse et problem på med programmering.
Etter du har løst et problem, spør deg selv: kunne du ha løst det annerledes?
→ Feks. færre kodelinjer eller annen fremgangsmåte?
→ Diskuter med en venn! Hvordan løste de det?

KONTAKT

Spørsmål om faget:

Gruppelæreren din eller foreleser

Studieinfo: spørsmål, klager, utsettelse av frister

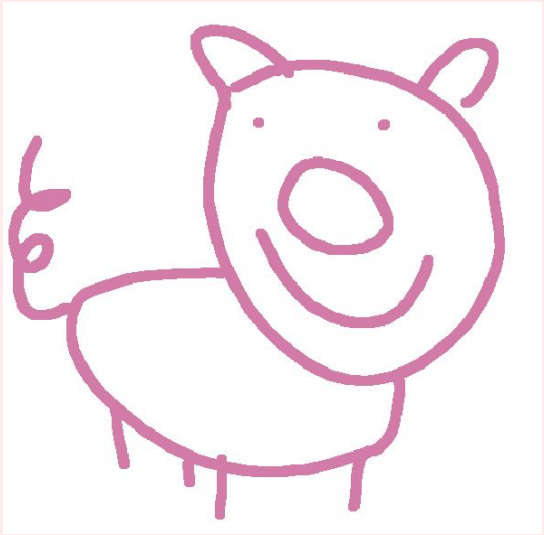
https://www.mn.uio.no/ifi/studier/kontakt/

UiO forvei: bekymringer, noen å snakke med

https://www.mn.uio.no/studier/forvei/

SiO: fysisk og psykisk helsehjelp

https://www.sio.no/helse
https://www.sio.no/helse/noen-%C3%A5-snakke-med



UKENS MESTERVERK: “Python Pig”