

# Forkurs i informatikk - Torsdag (Python)

## Dagens formål:

- Få mer trening i bruk av terminalen og hvordan man kan bruke den til å se på andre typer filer; nemlig Python-filer! Vi skal også se på hvordan vi kan skrive flere instruksjoner etter hverandre - som blir et eget program!
- Skrive og kjøre programmer i Python.

**Motivasjon:** I dag sikter øvelsene på å gjøre overgangen til programmering utover i semesteret enklere ved at dere blir kjent med hvordan Python-filer ser ut og hvordan vi skriver enkle programmer.

**Hjelp:** dersom det er noen ord eller uttrykk du ikke forstår står det en del forklart [her](#). Mye brukte kommandoer ligger [her](#).

*Ikke nøl med å spørre om hjelp underveis!*

## Programmeringsspråk og kildekode

Under dagens forelesning ble det snakket om hvordan alt er filer, og om hva kildekode er. Vi skal derfor i dag kikke på kildekode skrevet i programmeringsspråket Python, og lage enkle programmer selv.

Kildekode som er skrevet i Python-språket har filendelsen `“.py”`, som forteller datamaskinen om hvordan denne filen skal tolkes.

Aller først - forsikre deg om at du har forkursmappen som ble lastet ned fra tirsdagen ved å navigere deg til den i Terminalen (hint: **ls** og **cd**). Hvis du ikke har den, se [øvelser Dag 1](#) for hvordan dette skal gjøres. Når du er inne i mappen, bruk **tree**-programmet for å se hvilke filer som ligger der. Som du ser har du forskjellige filer:

Pythonfiler (.py)  
Tekstfiler (.txt)

I dag skal vi tittle nærmere på Python-filene. Åpne derfor kildekode-filen `“Dag2.py”` med **cat**-kommandoen for å se nærmere på den. Hva tror du Python-programmet gjør?

## Fra kildekode til kjøring

Teksten, eller kildekoden, som ligger i `Dag2.py` er det man trenger i Python for å kunne skrive ut til tekst til Terminal-vinduet. Nå skal vi gå videre med å kjøre programmet vårt. Dette gjør vi med programmet **python**, som tar inn navnet på filen som argument:

```
[bruker@maskin ~/MineFiler/Python] $ python3 Dag2.py
```

Nå har programmet (forhåpentligvis) kjørt ferdig. Gjorde programmet det du trodde?

## Endringer i kildekode

Nå som vi har kompilert og kjørt programmet, skal vi nå gjøre endringer i selve *kildekoden*. Åpne derfor Atom/Emacs med en av følgende programmer:

```
[bruker@maskin ~/MineFiler/Python] $ atom .
```

ELLER

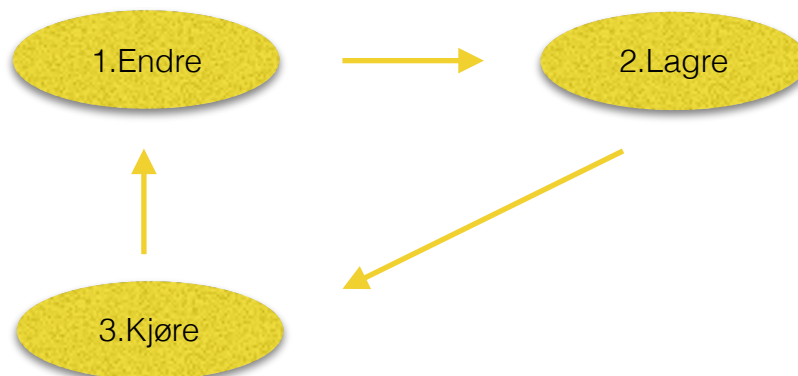
```
[bruker@maskin ~/MineFiler/Python] $ emacs &
```

Som du ser når vi åpner editoren, har vi fått fargelagt koden vår fordi den har en `.py`-filendelse. Fargene er for å kunne skille de forskjellige elementer i koden, slik at den blir mer lettlest for oss programmerere. Prøv nå å endre på programmet slik at den skriver ut "Hello world!" istedet. Når du har gjort endringene, husk å lagre. Kjør programmet som over, med **python**. Sammenlign utskriften til Terminalen med forrige gang du kjørte programmet. Ble det riktig? Kan du få programmet til å skrive det ut to ganger?

## Tekstvariabel

Som det ble snakket om på forelesning kan vi i dataprogrammene våre også ha variabler - lagringsplasser for ting som tekst og tall. Aller først skal vi se på et program som ligner på det fra sist øvelse, men som bruker variabler for lagring av tekst i utskriften. Åpne nå "*ProgOppgave2.py*" i Atom/Emacs, og se etter hva som er forskjellig fra forrige program. Hva tror du programmet skriver ut? Kompiler og kjør programmet for å se om det stemmer overens med det du trodde.

Endre nå programmet slik at variabelen navn inneholder ditt eget navn. Test deretter at det fungerer ved å *lagre og kjøre*.



*Programmeringsprosessen i Python*

## Andre variabler og if

Åpne nå “*ProgOppgave3.py*” og se på koden.

Kjør programmet. Hva blir utskriften? Nå skal du endre på programmet slik at utskriften blir noe annet uten å endre på noe annet enn tallet. Ble det riktig?

## Tekst over flere linjer

I Python kan man enkelt skrive ut tekster over flere linjer i programmet sitt ved å bruke

```
print("""Hei  
Hei  
Hei""")
```

eller

```
print("Hei\nHei\nHei")
```

Backslash'en (bakslask) i tekststrenger blir tolket som tegn som ikke skal vises direkte, men snarere indikere at usynlig tegn - her står **n** for det usynlige tegnet **newline**.

Skriv nå et program som skriver ut et skilt med navnet ditt på, ved hjelp av det nå har lært:

```
*****  
*                               *  
*   Ash Ketchum                 *  
*                               *  
*****
```

## Tekstformatering

Man kan også blande tall-variabler og tekster ved å *formatere* tekster. Formatering gjør det enklere å kunne bestemme *hvor* i teksten du vil ha visse variabler eller verdier. Se på programmet “*formatering.py*”, og kjør det. Forsøk nå å lage en egen tekst med så mange formateringer du vil, og skriv det ut til terminalen.

## Matematikk

Programmering egner seg godt for å skrive programmer som regner ut mer komplekse mattestykker, men dette skal dere lære mer om senere i semesteret. Ta en titt på “*utregning.py*” og prøv deg frem med litt med ulike tall.

## Et eget program

Til denne øvelsen skal du lage et program selv basert på det du har lært i de tidligere øvelsene. Lag derfor en ny fil i editoren din som du lagrer med filendelsen “.py”. Pass på å teste underveis mens du skriver.

Spesifikasjon av programmet:

- Lag en variabel for å ta vare på alder og legg inn en eksempel-alder

- Lag en variabel for å ta vare på et navn og legg inn et eksempel-navn
- Lag tester i programmet ditt som sjekker om en person er myndig eller ikke (dvs. er over 18 år gammel), og som skriver ut til terminalen:

*Hvis han ikke er myndig: "Kåre er ikke myndig, fordi han er X år gammel"*

*Hvis han er myndig: "Kåre er myndig, han er jo X år gammel!"*

## Et program for morroskyld

Helt til slutt skal vi leke oss litt med det siste programmet som ligger i mappen - "*Morro.py*". Åpne derfor filen i editoren du liker best. Hvis du vil gjette - hva kommer til å bli skrevet ut til Terminalen?

Kjør programmet. Hva skjer?

For deg som har lyst til å leke litt:

- Prøv å få programmet til å skrive ut alle tall mellom 0 og 1000000 (mill)
- Prøv å få programmet til å skrive ut alle partall mellom 0 og 1000000 (mill)
- Prøv å få programmet til å skrive ut hele nigangen (dvs.  $9*1$ ,  $9*2$ ,  $9*3$ ,  $9*4$ ). Et hint her er at du fint kan bruke gangetegn i programmet!

## For-løkke (frivillig/avansert)

Det forrige programmet kan også skrives på en annen måte. Åpne "*morro\_2.py*" i editoren din. Hvis du vil kan du prøve å gjøre det samme du gjorde med while-løkken.

## Oppgaver fullført

Bra jobbet, du er nå ferdig med alle forkursets øvelser! Husk at du ikke trenger å pugge kommandoene vi har vært borti i dag, og at dette mest er ment som øvelser som skal hjelpe deg på vei. Til neste uke er det fint om du husker på:

- Programmet **python3 [filnavn]** som kjører innholdet i en *“.py”*-fil

Synes du dette var gøy? Ta gjerne kontakt med et orakel på rommet ditt for litt mer avanserte øvelser. **Takk for at du har deltatt på forkurset - vi gleder oss til å programmere med deg utover i semesteret!**



## Guide til studier ved informatikk

Det finnes også en nettbasert introduksjon til studiet, som kan fungere som oppslagsverk for nye studenter ved IFI. Vi håper at alle tar seg en titt her enten du trenger repetisjon, ønsker litt mer stoff eller om du tør: Ta testen "Er du klar for Ifi"! Lenke:

<https://uio.instructure.com/courses/12798>

**PSST!** Se opp for Fagutvalgets kurs og bli gjerne med oss! Mer info på [fui.ifi.uio.no](http://fui.ifi.uio.no)