Innføring i programmering – Spyder (Python)

Del 1: La oss bli kjent med Spyder

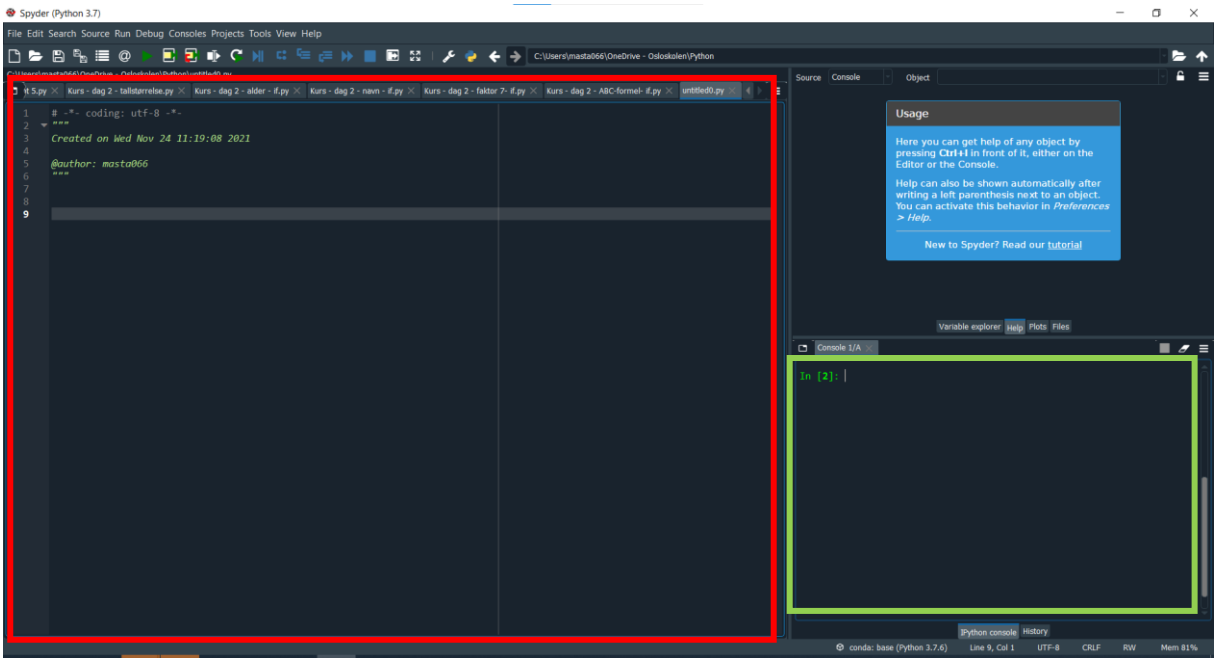
Åpne Spyder:

For å kunne programmere, trenger vi et program vi kan programmere i. Vi skal bruke et program som heter Spyder. For å åpne Spyder gjør følgende (hvis du ikke har lastet ned, last ned via www.spyder-ide.org):

1. Klikk enten på Windows-ikonet nederst i venstre hjørne eller klikk på «Windows-tasten» på tastaturet
2. Skriv inn «Spyder» og åpne

Oversikt over Spyder:

Når du har fått åpnet Spyder vil du bli møtt av tilsvarende tre vindu:



Det store vinduet til venstre (markert med rød ring rundt) er programmeringsvinduet vårt. Det er her vi skriver kodene våre. I dette vinduet vil det alltid stå noe når vi åpner et nytt vindu, denne teksten kan du bare fjerne når du åpner et nytt vindu.

Nede til venstre (markert med grønn ring rundt) er det som kalles «konsoll-vinduet», her får du ut eventuelt printede resultater fra kodingen, eller feilmeldinger om du har feil i det programmet du har skrevet.

Vinduet øverst til høyre (uten noen ring rundt) trenger du ikke å tenke noe på.

Hvordan regne i Spyder:

Å regne i Spyder er veldig likt det å regne i et hvilket som helst annet program på PC. Vi bruker de kjente tegnene + - * / for å gjøre de vanlige regnemåtene, = betyr er lik og > betyr større eler mindre (alt etter hvilken vei vi setter det).

For vår del er det to ting vi må være observant på, det første er at for å få et tall opphøyd i en eksponent (eks. 2^3), så skriver du ikke som i GeoGebra (2^3), men du skriver i stedet 2**3. Det andre er at av og til skal vi skrive dobbelt likhetstegn etterhverandre (==), men dette kommer vi tilbake til.

Hvordan fungerer Spyder:

En ting du må være klar over før vi starter kodingen i Spyder er hvordan Spyder, og alle andre programmer for koding, fungerer. Spyder vil alltid lese kode fra toppen og nedover. Ha dette i bakhodet til delkapittel hvor vi definerer variabler (del 3).

For å «kjøre» (eller starte) et program du har kodet må du alltid trykke på «play»-knappen som du finner øverst på verktøylinjen til Spyder:

Resultatet, om du har bedt om å få ut et resultat kommer i så fall i «konsoll-vinduet».

Del 2: «Print»

Hvis du ønsker å få ut et resultat fra kodingen vår i konsollvinduet bruker vi kommandoen *print*.

Eks. Hvis jeg skriver følgende i programmeringsvinduet:

Og kjører dette programmet (trykker «Play»-knappen) vil jeg få ut følgende i konsollvinduet mitt:

```
In [3]: runfile('C:/Users/masta066/OneDrive - Osloskolen/Python/
untitled0.py', wdir='C:/Users/masta066/OneDrive - Osloskolen/Python')
Hei
```

(Alt som står over «Hei» er bare plasseringen til filen jeg kjører, altså programmet jeg nå jobber i, vi trenger ikke å tenke over det – det vil alltid komme når vi kjører et program).

Ønsker jeg å få ut tekst i en *print*, må jeg huske å sette gåseøyne, «», rundt teksten.

Oppgave 1:

Lag et program som printer ut *Math is life* i konsollvinduet.

Oppgave 2:

Lag et program som printer ut summen av 2 + 3.

Del 3: Definere variabler

Når du programmerer er det veldig vanlig å bruke variabler, i stedet for tall, når du regner. La oss ta for oss et eksempel.

Eksempel: Du skal regne ut antall timer i et år.

```
1  #Utregning av antall timer i et år
2  dager = 365
3  timer = 24
```

Tips: Bruker du # framom tekst vil dette bli en kommentar (til deg selv eller lærer) i programmet, og som ikke blir en del av programmet når du kjører det.

En variabel kan nesten være hva som helst, men den kan ikke starte med et tall eller ha mellomrom.

Hvis jeg nå ønsker å regne ut antallet timer kan jeg bruke variablene *dager* og *timer* til å regne med:

```
1  #Utregning av antall timer i et år
2  dager = 365
3  timer = 24
4  totalt = dager * timer
5  print("Det er", totalt, "timer i et år")
```

Denne printen gir følgende i konsolvinduet:

```
In [4]: runfile('C:/Users/masta066/OneDrive - Osloskolen/Python/
untitled0.py', wdir='C:/Users/masta066/OneDrive - Osloskolen/Python')
Det er 8760 timer i et år
```

I printen min på linje 5 her ber jeg Spyder printe ut tre elementer:

- Første element: Teksten «Det er»
- Andre element: Variabelen *totalt* som er produktet av dager * timer
- Tredje element: Teksten «timer i et år»

Som da til sammen gir outputen *Det er 8760 timer i et år*.

Oppgave 4:

Du har en deltidsjobb og tjener 190 kr i timen. Du arbeider 26 timer per uke, og vi sier at det er fire uker i en måned.

Hvor mye tjener du ila en måned?

Lag et program som printer ut svaret.

Oppgave 5:

Her er et program for utregning av antall minutter per år.

```
#Utregning av antall minutter i et år
a = 365
a = 24
b = 60
minutter_pr_år = a*a*b
print("Det er", minutter_pr_år, "minutter i et år")
```

Hva er feil i programmet?

Oppgave 6

Du har satt inn 40 000 kr på en høyrentekonto hvor du får 2,3 % rente per år. Hvor mye vil du ha på kontoen etter 5 år?

Del 4: input

Det hender du ønsker å lage et program hvor tallene det skal regnes med varierer for hver gang du kjører programmet, vi skal putte inn nye tall hver gang, en ny input før programmet kjører. Det kan du løse med å bruke kommandoen *input*.

For eksempel kan du definere a som en input ved å skrive a = input(). Da kan du skrive inn inputen i konsollvinduet, og deretter vil Spyder regne med tallet du skrev inn for a.

Du ser at etter input i forrige avsnitt står det en tom parentes, (). Inne i denne parentesen kan du for eksempel sette inn en tekst du vil skal dukke opp i konsollvinduet i forbindelse med at du skal skrive inn en input. La oss vise med et eksempel.

Eksempel: En kilo godteri koster 120 kr. Lag et program hvor du kan variere mengden godteri du kjøper, og som gir ut prisen du må betale.

```
#Godteri-regneren

kilo = float(input("Hvor mange kg godteri kjøper du? Skriv inn:"))

kilospris = 120

print("Du må betale", kilo * kilospris, "")
```

To ting før jeg fortsetter forklaringen av selve eksempelet:

1. Før *input* har jeg skrevet *float*, grunnen til det er at hvis jeg ikke skriver det, så vil Spyder bare anse tallet jeg skriver inn etterpå som ren tekst og ikke et tall. Jeg kommer tilbake til dette etter eksempelet.
2. I dette eksempelet har jeg valgt å gjøre selve utregningen av prisen som må betales inne i *print*-kommandoen, i stedet for før, slik som jeg gjorde i eksempelet i del 3.

Når jeg kjører dette programmet så vil dette dukke opp i konsollvinduet:

```
Hvor mange kg godteri kjøper du? Skriv inn:
```

Her skriver jeg rett etter der det står «Skriv inn:» slik og trykker enter-tasten:

```
Hvor mange kg godteri kjøper du? Skriv inn:2
```

Da får jeg med en printet ut resultatet i konsollvinduet:

```
Hvor mange kg godteri kjøper du? Skriv inn:2
Du må betale 240.0 kr.
```

Tilbake til det jeg nevnte i eksempelet angående *float*. Når du bruker kommandoen *input,* og det er tall vi ønsker å sette inn, så må du enten skrive *float* før *input*-kommandoen eller *int*.

- *float* gir deg desimaltall av tallet du skriver inn som input
- *int* gir deg heltall av tallet du skriver inn som input

Oppgave 7

Du har en deltidsjobb hvor du ar en gitt timelønn på 185 kr per time.
Lag et program hvor du kan variere antall timer du jobber, og som automatisk gir deg tilbake lønnen du ville fått ved dette antallet timer jobbet.

Del 5: if

Det vi har sett på så langt i bruken av Spyder har ikke vært mer enn en vanlig kalkulator klarer, men nå skal vi se på nyttige koder vi kan benytte i programmering.

Koden vi skal se på er *if*, og hvis vi oversetter if til norsk får vi hvis. Når du bruker *if* i et program så setter du rett og slett opp en forutsetning. Du spør hvis *en eller annen forutsetning er sann* så skal Spyder printe ut et resultat, hvis ikke, printer den ut et annet resultat. La oss se på et eksempel.

Eksempel: Du ønsker å kjøpe deg en bolig. Du trenger 300 000 kr i egenkapital. Du sparer 50 000 kr hvert år. Lag en kalkulator hvor du kan variere antall år du har spart, og som svarer på om du har spart nok til å kjøpe bolig eller ikke.

```python
#Boligspareren

år = float(input("Hvor mange år har du spart? Skriv inn her: "))

sparebeløp = 50000

oppspart = år * sparebeløp

if oppspart < 300000:
    print("Du har ikke spart nok, spar mer!")

else:
    print("Du har spart nok, din gjerrikark!")
```

Hvis jeg da kjører programmet og for eksempel skriver inn 3, så får jeg dette:

```
Hvor mange år har du spart? Skriv inn her: 3
Du har ikke spart nok, spar mer!
```

Skriver jeg derimot inn 10 får jeg dette:

```
Hvor mange år har du spart? Skriv inn her: 10
Du har spart nok, din gjerrikark!
```

En ting du må kjenne til her er at hvis du skal skrive at forutsetningen din skal være lik noe, for eksempel hvis oppspart skal være lik nøyaktig 300 000 kr, må du bruke dobbelt likhetstegn:

```
if oppspart == 300000
```

Hvorfor bruker vi egentlig dobbelt likhetstegn (==)? Jo, fordi et likhetstegn gir bare Spyder beskjed om å sjekke om de to enhetene på hver side av likhetstegnet er lik hverandre i Spyders minne.

Eksempel: Skriver vi følgende linjer:

```python
a = 1

if k = a:
```

Så vil Spyder se etter om variabelen k er lik den lagrede variabelen a.

Hvis vi derimot setter dobbelt likhetstegn (==), så vil Spyder sammenligne for å se om de to verdiene på hver side av de to likhetstegnene er lik hverandre.

Eksempel: Skriver vi følgende linje:

```python
if k == 1:
```

Så vil Spyder se om k er lik verdien 1.

Oppgave 8

Du har 150 000 kr i et aksjefond og forventer 8 % årlig avkastning.

a) Lag et program hvor du kan variere antall år pengene har stått på konto, og hvor programmet forteller deg om beløpet har blitt doblet eller ikke.
b) Endre programmet ditt slik at det i tillegg til å fortelle deg om beløpet er blitt doblet eller ikke, også gir deg beløpet du har på konto etter gitte antall år.

Oppgave 9

Lag et program hvor du setter en bestemt pinkode. Deretter skal du få programmet til å be om at du skriver inn pinkoden, til slutt skal programmet fortelle deg om du har skrevet inn riktig eller gal pinkode.

Del 6: While-looper

I *if*-kommandoen så satte vi en forutsetning, og hvis det forutsetningen var oppfylt, så skulle vi gjøre det som stod inne i kommandoen én gang.

Når vi snakker om looper (eller ofte kalt løkker på norsk), tenker vi på en bit av koden vår som skal gjenta seg selv, helt til vår forutsetning ikke lengre er oppfylt, eller det vi kan tenke på som en terskel/grense er nådd. Vi skal nå se på to ulike looper, først *while*-looper og i neste del *for*-looper.

Nok en gang kan oversetter vi hva *while* betyr til norsk, og da ville vi sagt *mens*. Så *while* bruker vi som en loop for en utregning som skal foregå *mens* en verdi enten er høyere eller lavere enn en gitt terskel/grense. Du bestemmer hva denne terskelen/grensen i while-loopen skal være.

På samme måte som for if-kommandoen setter vi en forutsetning som må være oppfylt for at vi skal gjøre utregningene som står inne i *while*-kommangoen. I *while*-loopen kan vi tenke at vi setter en terskelverdi/grenseverdi, og så lenge terskelverdien/grenseverdien ikke er nådd gjentar den bare utregningene som står inne i loopen.

Her er et eksempel på hvordan det kan se ut:

```
1  startbeløp = 1000
2  verdi = startbeløp
3  spare = 500
4  år = 0
5  rente = 1.03
6
7  while verdi < 20000:
8      verdi = verdi * rente + spare
9      år = år + 1
10
11 print("Det tok", år, "år, og da er verdien på", verdi, "kr")
```

Dette programmet viser en sparing, hvor man starter med 1000 kr på konto og i tillegg sparer 500 kr ekstra hver år etter. Årlig rente er på 3 % (vekstfaktor 1.03), det betyr at det er penger du får av banken når du sparer på en konto hos dem.

Her er ser vi at *while*-loopen har en terskel/grense på 20 000 kr. Det betyr at så lenge «verdi», som er penger du har på konto, er mindre enn 20 000 kr skal man fortsette å legge på 3 % renter, pluss sparebeløpet på 500 kr.

NB: Inne i *while*-loopen ser du forresten to tilfeller av en veldig vanlig praksis innen programmering. Vi overskriver gamle variabler med en modifisering av seg selv. «verdi» overskrives ved å ta den gamle «verdi», øker den med 3% (ganger med «renten») og legger til 500 kr («sparing»). Dette gjør at man slipper å få en ny variabel for hvert år, man bare oppdaterer den gamle. Tilsvarende for «år». Der tar vi den gamle verdien av «år» og legger til 1. Dette gjør at vi får en tellemekanisme som egentlig teller hvor mange ganger vi har vært igjennom *while*-loopen vår.

Til slutt har vi en litt mer sammensatt *print* enn vi har sett tidligere. Her veksles det mellom tekst som skal printes ut (alt som står i gåseøyne «») og variablene år og verdi. Da får vi dette ut når vi kjører programmet:

```
Det tok 25 år, og da er verdien på 20323.410090557798 kr
```

Oppgave 10

Ta for deg sparingen i fond, som du gjorde i oppgave 8, men endre koden slik at du løser den ved hjelp av en *while*-loop i stedet for en *if*-loop.

Del 7: for-looper

Nå har vi sett på to kommandoer (*if* og *while*) som vi kan bruke når vi vil sette en terskel/grense for når kommandoene skal avsluttes. *for*-looper bruker vi derimot når vi vet hvor mange ganger vi vil kjøre loopen på forhånd, før den skal avsluttes.

Her er et veldig enkelt eksempel på hvordan *for*-loopen kan se ut:

```
1  år = 10
2  a = 0
3
4  for i in range(år):
5      a = a + 1
6
7  print(a)
```

I dette eksempelet ser vi på en enkel tellemekanisme, hvor vi øker verdien a med 1 per runde den kjører i *for*-loopen. Det er variabelen «år» som i dette tilfellet bestemmer hvor mange runder vi skal kjøre loopen, da *for*-kommandoen henviser til «år» i *range*-kommandoen som kommer på linje 4.

Kjører vi denne, vil loopen kjøres ti ganger og vi får printet ut verdien på a, som da blir:

```
10
```

for-kommandoen er veldig fin å bruke i for eksempel utregning av sparing når vi vet hvor mange år framover i tid sparingen skal foregå.

Oppgave 11

Ta for deg sparingen i fond, som du gjorde i oppgave 8, men endre koden slik at du løser den ved hjelp av en *for*-loop som skal gå i 20 år.