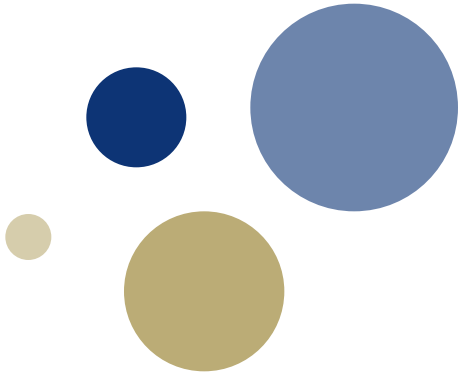
NTNU

Kunnskap for en bedre verden



Python: Variable og beregninger, input og utskrift

TDT4110 IT Grunnkurs

Professor Guttorm Sindre

Læringsmål og pensum

- Mål for denne uka:
  - **Vite litt om design av programmer (2.1, 2.2, 2.4)**
  - Kunne skrive ut til skjermen (2.3, 2.8)
  - Kunne bruke variable og vite om datatyper (2.5)
    - og gjøre enkle beregninger (2.7)
  - Kunne lese inndata fra tastatur (2.6)
- Pensum
  - […] Python Ch 2: Input, Processing, and Output

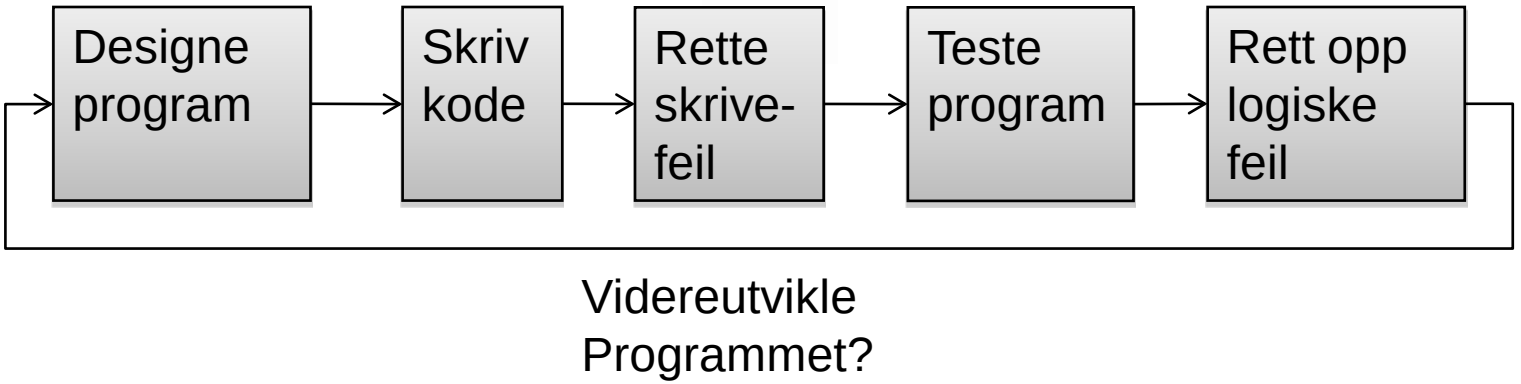
- Kahoot på fredag: 2.3 – 2.8
  - Vil lønne seg å ha lest + prøvd ut innlesing og utskrift, variabelkonvertering og beregninger i Python

Designe et program

Kapittel 2.1-2.2

Programutviklingssyklus

- Programutviklingssyklusen er prosessen man følger når man lager og utvikler programmer:



Programutviklingssyklus

1. Designe programmet:
   - Forstå oppgaven som programmet skal utføre
   - Avgjøre stegene som må tas for å utføre oppgaven
2. Skrive kode:
   - Må følge regler definert av programmeringsspråket
3. Rette opp skrivefeil:
   - Feilmeldinger fra tolker/kompilator må rettes opp for å kunne kjøre
     - NB: av og til kan skrivefeil likevel bli grammatisk korrekte
     - Dette er verre, da ses feilen først under kjøring
4. Teste programmet:
   - Når programmet kan kjøres må det testes for logiske feil i koden, for eksempel feil i beregninger, produserer feil resultat, osv.
5. Rette opp logiske feil (debugging)
   - Finner og retter opp logiske feil

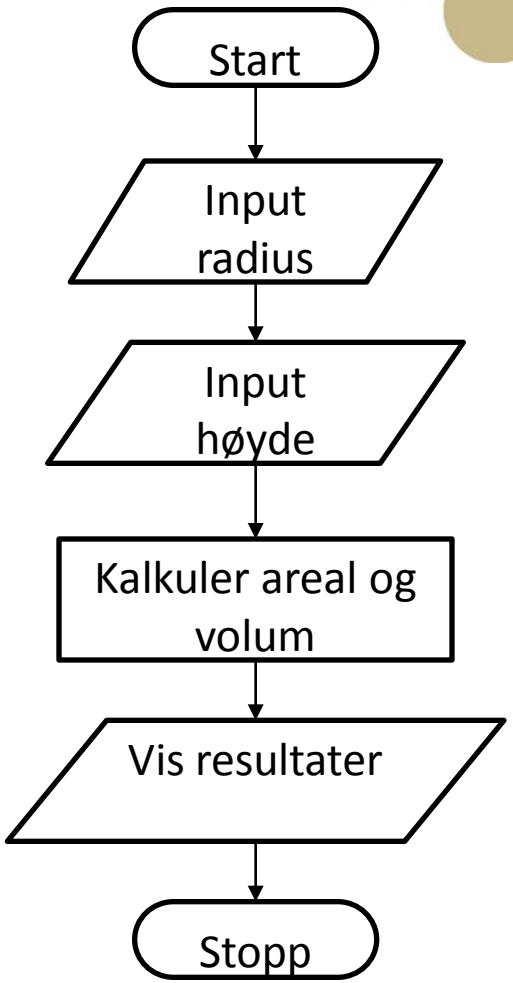
Hjelp til å designe programmer: Pseudokode

- Beskriver programmet med naturlig språk
- Pseudokode kan ikke forstås av en datamaskin
  - men kan oversettes av mennesker til ulike programmeringsspråk

```
Hent inn (input) radius (r) for sylinderen
Hent inn (input) høyde (h) for sylinderen
Kalkuler areal og volum av sylinderen
    a) kalkuler omkrets av sirkel (2*pi*r)
    b) areal av sirkel (pi*r**2)
    c) volum av sylinder (A_sirkel*h)
    d) areal av sylinder (2*A_sirkel + omkrets*h)
Vis (på skjerm) areal og volum
```

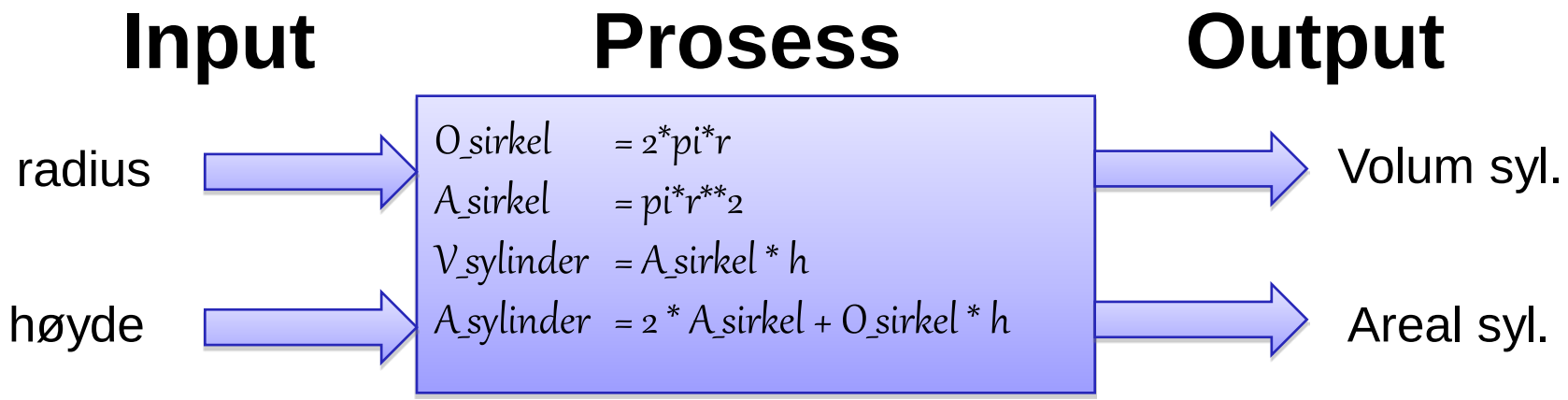
Hjelp til å designe programmer: Flytskjema

- Beskriver programmer grafisk:
  - Ovaler: *Terminalsymboler* som viser start og stopp
  - Parallellogram: *Input- og output-symboler* (hente inn fra bruker og vise til skjerm)
  - Rektangel: *Prosesseringssymboler* der man utfører noe på data (for eksempel beregninger)
- Blir mer nyttig hvis det også fins valg (romber) og løkker (piler tilbake)



Hjelp til å designe programmer: Input – prosessering - output

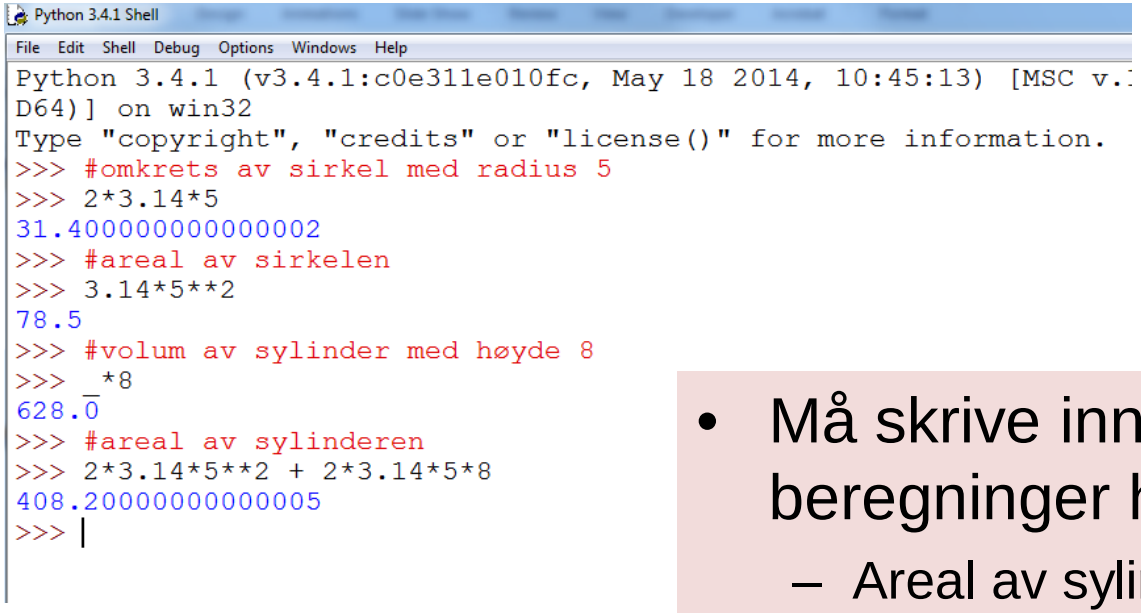
- Dataprogrammer utføres ofte som en trestegsprosess:
  1. Mottar input
  2. Utfører prosessering av input (gjøre noe med input)
  3. Produserer output



Brukervennlighet

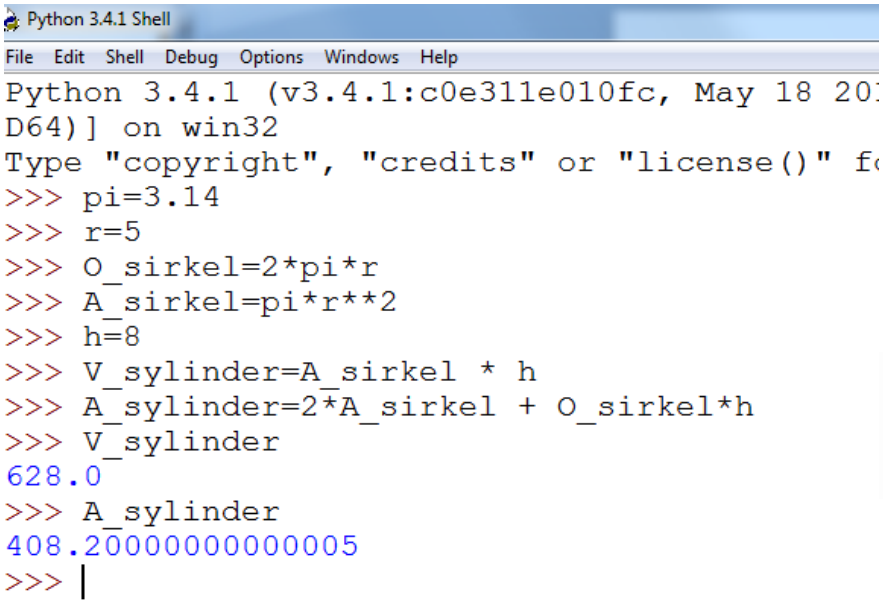
- Hvis programmet skal kunne brukes av andre:
  - Må være enkelt for brukeren
    - Ikke kreve programmeringskunnskaper
    - Ikke kreve kjennskap til interne beregninger
    - Lett å bruke nye inputverdier fra gang til gang
  - Forståelig
    - All input må være forklart
    - All output må være forklart
    - Alltid lett å skjønne hva som skal gjøres
  - Pluss flere kriterier som vi ikke går inn på her

Vi hadde først…



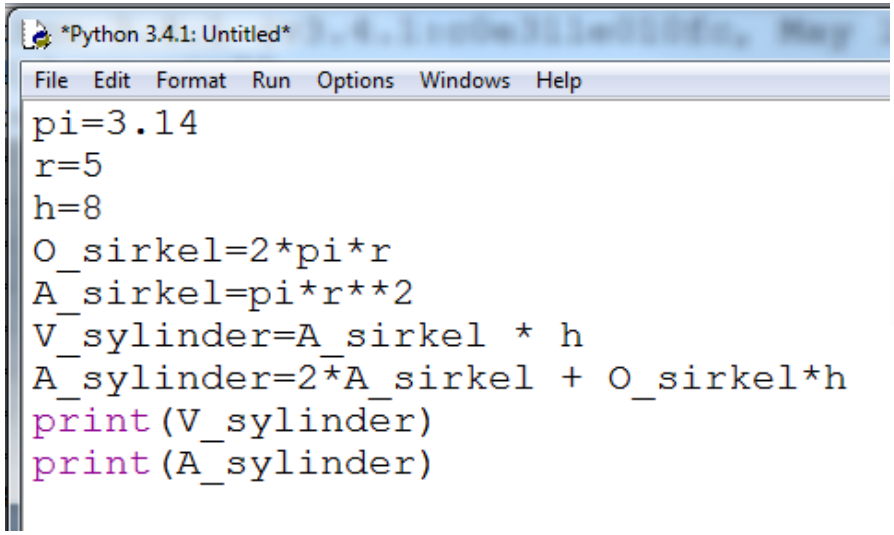
- Må skrive inn beregninger hver gang
  - Areal av sylinder: tungvint, må gjenta formler vi allerede skrev lenger oppe

Så…



- Bedre: variable
  - Husker verdier
  - Slipper å gjøre mellomregninger på nytt
- Må fortsatt programmere alt selv
  - Ingen lettelse av arbeid for andre

Lagret program:



- Bedre:
  - Slipper å skrive alt på nytt
- Men
  - Må i koden for å endre r, h
    - Tungvint
    - Uaktuelt for bruker som ikke kan programmere
  - Uforståelig output
    - Kun tall uten forklaring

```
>>> ================================ RESTART ================================
>>>
628.0
408.20000000000005
>>>
```

Bedre lagret program:

Python 3.4.1: sylinder2015.py - C:/Python34/sylinder2015.py

File Edit Format Run Options Windows Help

```
pi=3.14
r=float(input('Hva er radius til sylinderen? '))
h=float(input('Hva er høyden til sylinderen? '))
O_sirkel=2*pi*r
A_sirkel=pi*r**2
V_sylinder=A_sirkel * h
A_sylinder=2*A_sirkel + O_sirkel*h
print('Volumet til sylinderen:',V_sylinder)
print('Arealet til sylinderen:',A_sylinder)
```

- Bedre:
  - Nå kan en bruker gi inn ulike verdier for r og h UTEN å måtte endre koden
  - Brukeren får forklaring
    - Hva skal inn?
    - Hva kommer ut?

```
>>> ================================ RESTART ================================
>>>
Hva er radius til sylinderen? 5
Hva er høyden til sylinderen? 8
Volumet til sylinderen: 628.0
Arealet til sylinderen: 408.20000000000005
>>>
```

Kommentarer

Kapittel 2.4

Kommentarer i programmer

- Forklarer hva som blir gjort i programmet
- Blir ignorert av tolker / kompilator
- Blir ikke skrevet ut til skjerm.
- Startes i Python med tegnet #

- Kommentarer er viktigere
  - Jo større programmet er
  - Jo flere som skal samarbeide om å lage det
  - Jo lenger det skal brukes
- Men også ok i egen jobbing med små programmer:
  - Først skrive pseudokode som kommentarer
  - Så skrive programkoden mellom kommentarene

Læringsmål og pensum

- Mål for denne uka:
  - Vite litt om design av programmer (2.1, 2.2, 2.4)
  - **Kunne skrive ut til skjermen (2.3, 2.8)**
  - Kunne bruke variable og vite om datatyper (2.5)
    - og gjøre enkle beregninger (2.7)
  - Kunne lese inndata fra tastatur (2.6)
- Pensum
  - […] Python Ch 2: Input, Processing, and Output

Vise output med print-funksjonen

Kapittel 2.3

Skrive informasjon til skjerm

- For å skrive noe til skjerm brukes funksjonen print(uttrykk)
  - for eksempel

    ```
    print('Hei verden!')
    ```

  - skriver teksten ”Hei verden!” til skjerm
- Uttrykket kan være
  - Tall, utregning av tall, logiske uttrykk, tekst og/eller variabel
  - Ett eller flere elementer

Vise flere elementer med `print` funksjonen

- Kan vise flere elementer i samme print-setning
  - Elementene skilles med komma
  - Eksempel på å vise to elementer til skjerm:

```
print('Areal av sirkelen blir: ', 2*3.14*5)
```

- Tekstrenger har fnutter rundt seg
- Tall, aritmetiske uttrykk og variable har ikke fnutter
- print legger på et mellomrom mellom hvert element
  - Men kan endre dette med ekstra triks, mer om det senere

Variable og datatyper

Kapittel 2.5

Læringsmål og pensum

- Mål for denne uka:
  - Vite litt om design av programmer (2.1, 2.2, 2.4)
  - **Kunne bruke variable og vite om datatyper (2.5)**
    - og gjøre enkle beregninger (2.7)
  - Kunne skrive ut til skjermen (2.3, 2.8)
  - Kunne lese inndata fra tastatur (2.6)
- Pensum
  - […] Python Ch 2: Input, Processing, and Output

Hvorfor trenger vi variable?

- Kan beregne ved å bare skrive inn tallene direkte?
- F.eks. regne ut omkrets og areal for sirkel:

```
>>> 2*3.14*5  #omkrets hvis radius er 5
31.400000000000002
>>> 3.14*5**2  #areal hvis radius er 5
78.5
>>>
```

Ln: 44 Col: 4

- Men funker dårlig…
  - …hvis omkrets og areal trengs videre i andre beregninger
  - …hvis vi skal beregne for mange ulike sirkler
  - …hvis programmet skal bli nyttig for noen som ikke kan programmere

Variable

- Formål: huske data til bruk senere i programmet
  - Variabelen har et navn
  - Representerer en verdi lagret i datamaskinens minne
  - Innhold kan variere i løpet av programmet
  - Er ikke helt det samme som en variabel i matematikken
- Variabel er et sentralt konsept i programmering
  - trengs i utregninger og beslutninger
  - for å lagre og gjøre endringer på informasjon

Opprette variable

- Oppretting av variable gjøres ved tilordningsuttrykk:

  `alder = 23`

  – Etter dette vil `alder` referere til verdien 23 i minnet

  – verdien kan endres med nye tilordninger

  - f.eks `alder = 24` eller `alder = alder + 1`

- En tilordning skrives generelt på formen:

  `variabel = uttrykk`

  –Tegnet ”=” (er lik) er en tilordningsoperator

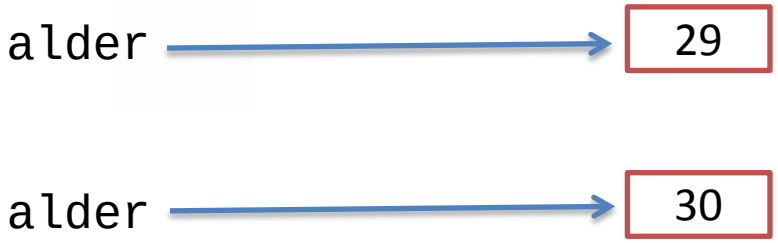
  –*variabel* er navnet på variabelen

  –uttrykk representerer en verdi

Endring av verdi i en variabel

- En variabel kan endre verdi i løpet av et program.
- Man endrer verdien av variabel ved å gjøre en ny tilordning.
- Eks:

```
alder = 29
print(alder)
alder = 30
print(alder)
```



Lagring av verdier

- Variablenes verdier lagres sammen med andre data og programmer i minnet (RAM på datamaskinen)
  - Tenk på RAM som en kommode med nummererte skuffer
  - I hver skuff kan du lagre en *byte* eller 8 *biter* (bits)
  - En byte er et tall i området 0-255 (8 sifre i 2-tallssystemet)
- Pythons tolker / kompilator finner selv hvor mye plass som trengs for å lagre en verdi
  - En enkelt byte: lite heltall eller bokstav
  - Flere bytes: store heltall, flyttall
  - Mange bytes: lange tekststrenger eller lister

Regler for variabelnavn

- Forbudt å bruke
  - Nøkkelord i Python (f.eks. def, if, …) som variabelnavn
  - Andre symboler enn bokstav eller _ som første tegn
  - Andre symboler enn bokstav, tall eller _ i påfølgende tegn
    - NB: Python skiller mellom store og små bokstaver
    - Dvs., pris, Pris, PRIS og PRiS vil være ulike variable
  - Et navn kan ikke brukes i programmet før variabelen er opprettet
- Anbefalinger
  - Bruk navn som er informative, ikke villedende eller intetsigende
  - Unngå variabelnavn som lett kan forveksles / feiltastes
  - Hvis variabelnavn består av flere ord, bruk _ mellom ordene
    - F.eks. areal_sirkel, pris_inkl_moms
    - Eller stor bokstav på nytt ord: arealSirkel, prisInklMoms

Datatyper

- I Python kan variable ta vare på ulike typer data (datatyper), som
  - Heltall (int): `antall = 7`
  - Desimaltall (float): `penger = 35.5` `# Bruker . i stedet for ,`
  - Tekststreng (str): `navn = 'Petter'` `# Bruker fnutter`
  - Sannhetsverdi (bool): `rykte = True` eller `rykte = False`
- Datatypen kan sjekkes med

  `type(uttrykk)`
- I Python får variable type ved tilordning
  - Dvs. at en variabel kan bytte datatype underveis i programmet

Oppgave: Datatyper

- Angi hvilken datatype disse variablene bør være av heltall (integer), desimaltall (float), streng (string), sannhetsverdi (boolean):
  - A) Variabel for å lagre et telefonnummer
  - B) Variabel for å lagre millimeter nedbør
  - C) Variabel for å lagre navn på et fag
  - D) Variabel for å lagre om en deltaker er påmeldt eller ikke
  - E) Variabel for å lagre pris på bensin
  - F) Variabel for å lagre personnummer
  - G) Variabel for å lagre en tekstmelding

Mer om variable og datatyper

- Noen programmeringsspråk har
  - En grense på hvor mange siffer en variabel kan takle
  - Forbud mot å plutselig endre type på en variabel
- Python har ikke dette
  - Kan skrive inn veldig lange tall
  - Kan tilordne en annen type verdi i en variabel enn den har
- Noen spesielle måter å skrive tall på
  - vitenskapelig notasjon: 5.9e8 betyr $5.9 * 10^{8}$; 3.2e-8 betyr $3.2 * 10^{-8}$
  - komplekse tall: 1j i Python betyr det samme som i matematisk
    - Det komplekse tallet (1+i) må skrives (1+1j)
    - Skriver du bare (1+j) tror Python at j skal være en variabel

Læringsmål og pensum

- Mål
  - ~~Lære om å designe et program~~
  - ~~Lære om skrive ut til skjermen (print)~~
  - ~~Lære om variable~~
  - **Lære om å lese fra tastatur**
- Pensum
  - […] Python: Chapter 2 - Input, Processing, and Output

Lese input fra tastatur

Kapittel 2.6

Hent input fra tastatur med funksjonen input

- I Python brukes funksjonen input for å få taste inn data:

  variabel = input(prompt)

  –promptet vises på skjerm når programmet venter på input

  –variabel er navnet på variabelen som får data som blir tastet inn

  –Eks:

  navn = input('Hva er navnet ditt? ')

  –Bruk mellomrom på slutten av prompt-teksten

  - Virker uten også, men…
  - inndata kommer da kloss etter promptet

Lese inn tall med input funksjonen

- input-funksjonen returnerer alltid en tekststreng
  - dvs. en tekst med fnutter rundt.
- Tekst funker dårlig i matematiske beregninger
- Hva hvis du ønsker å
  - hente inn tall fra brukeren?
  - gjøre beregninger på disse tallene?
- Må konvertere tekststrengen til et tall
- Mulige funksjoner for dette:

```
variabel = int(element)          # Konverterer element til et heltall
variabel = float(element)        # Konverterer element til et desimaltall
variabel = eval(uttrykk)         # Konverterer uttrykk til verdi
```

Bruk av int() og float()

- To måter å hente heltall fra input-funksjon:

  streng_verdi = input(’Hvor mange kuer har du ? ’)

  kuer = int(streng_verdi) # Oversetter streng til heltall

- Du kan gjøre begge deler i en setning:

  kuer = int(input(’Hvor mange kuer har du ? ’))

  –Kalles nøstet funksjonskall og fungerer på følgende måte:

  - Først utføres den innerste funksjonen, altså input(’Hvor mange..’)
  - Verdien til input-funksjonen brukes så i int-funksjonen
  - Resultatet fra int-funksjonen lagres i variabelen kuer

- Hente ut flyttall:

  cash = float(input(”Mye penger har’ru? ”))

Utføre kalkulasjoner

Kapittel 2.7

Operatorpresedens
(prioritert rekkefølge)

- Python har følgende matteoperatorer: + - * / // % **
- Rekkefølge av utregning avhenger av presedens:
  - Eksponent: **
  - Multiplikasjon, divisjon og rest av divisjon: * / // %
  - Addisjon og subtraksjon: + −
  - Dvs.: 12 + 6 / 3 blir 14 og ikke 6 (6/3 blir utført først)
  - Lik prioritet? Da utregnes det fra venstre mot høyre
- For å sikre korrekte beregninger brukes parenteser:
  - (12+6)/3 blir 6
  - 10/(5-3) blir 5 osv…
  - Nøstede parenteser løses innenfra og ut

Bruk av variable i utregninger

- Når du bruker flere variable i ett uttrykk, vil:
  - Variabelen på venstre side bli resultatet av utrekningen
  - Alle variablene på høyre side er verdier som brukes i utrekning.
  - Eks:

A=5

B=8

C = A + B + 9

Resultat lagres her

5 + 8 + 9

Vi prøver litt…

Programmere litt?

- Oppgave: Lag et program som
  - Henter inn tre måleverdier
  - Kalkulerer gjennomsnittet (summen delt på 3)
  - Viser resultat på skjerm
  - Eksempel på kjøring:

```
>>> ================================ RESTART ================================
>>>
Skriv inn måling 1: 2.43
Skriv inn måling 2: 5.67
Skriv inn måling 3: 6.2224
Måling 1: 2.43 Måling 2: 5.67 Måling 3: 6.2224 Snitt: 4.774133333333333
>>>
```

- Prøv å programmere dette selv!
  - Hvis du er så god at det blir enkelt: Lag penere utskrift
    - Bruk tabulator eller format
    - Vis at du ville klare å få tall i pene kolonner under hverandre hvis det skulle regnes ut flere gjennomsnitt

Mer om utskrift til skjerm

Kapittel 2.8

Endring av oppførsel for print

- Endre separasjon mellom elementer

```
a=5
b=7
print( a, b)                    # gir 5  7
print( a, b, sep='_')           # gir 5_7
print( a, b, sep='')            # gir 57
```

- Annen avslutning enn linjeskift

```
print('Her kommer en setning.', end='')
print('her kommer en setning til.')
```

  - Begge blir skrevet ut på samme linje!
  - Kunne ha satt **end='\n\n'** for å gå to linjer ned

Escape-karakterer

- Escape-karakterer er spesialkarakterer
  - kan skrives inn i tekststrenger for spesielle operasjoner:

\n Gir linjeskift

\t Hopper til neste tabulator

\’ Skriver ut tegnet ’

\” Skriver ut tegnet ”

\\ Skriver ut tegnet \

–Eks:

```
print('\tInnrykk er kult med\nNy linje')
```

Formattering av tall ved hjelp av funksjonen format

- Funksjonen format
  - tar inn et tall og en tekststreng som viser format
  - Returnerer tekststreng hvor tallet er konvertert til dette formatet

  format(tall, formattering)

  –formattering er en tekststreng for ulike valg, f.eks:

```
format(1/3, '.2f')      # 2 desimaler, f står for float
format(1/3, '10.1f')    # 1 desimal og setter av 10 tegn
format(1/3, 'e')        # vitenskapelig notasjon på tallet
format(1/3, '.0%')      # tallet i prosent med 0 desimaler
format(500, '10d')      # heltall der det settes av 10 tegn
```

  –Typisk bruk:

```
print(format(1/3,'10.5f'))
```

Oppsummering

- Utviklingssyklus:
  - Design, programmer, rett opp skrivefeil, test, rett logiske feil
- Skrive til skjerm: print(uttrykk)
  - Kan ha flere uttrykk etter hverandre
  - Formattering (pen utskrift): format(tall, 'formatstreng')
- Variabler: Referanser med navn til verdier i minnet
- Tilordning: Gi variable verdier: **variabel = uttrykk**
  - Evt. variable på høyre side av **=** bidrar med verdier i regnestykket
  - Resultatet blir lagret i variabel på venstre side
- Basis datatyper: int, float, str, bool
  - Bruk av feil type gir ofte kluss
- Lese fra tastatur: variabel = input(prompt)
- Operatorpresedens: Rekkefølge av matematiske operatorer
- Formattering av tall ved hjelp av funksjonen

Kahoot neste uke…

- Handler om neste ukes programmeringspensum:
  - Kapittel 3 ”Decision Structures and Boolean Logic” i Gaddis-boka
- Kommer til å stille modifiserte varianter av noen av disse spørsmålene fra Gaddis-boka
  - What is a Boolean expression? (Checkpoint 3.4)
  - Write an if statement that assigns 0 to x if y is equal to 20. (Checkpoint 3.6)
  - How does a dual alternative decision structure work? (Checkpoint 3.8)
  - What would the following code display? (Checkpoint 3.11)
  - Assume the variables a=2, b=4, … Circle the T or F for each of the following conditions… (Checkpoint 3.16)
  - Write an if-statement that displays ”The number is not valid…” (Checkpoint 3.19)
  - When a program compares characters, it actually compares… (Review Question Multiple Choice 4)
  - If the expression on the left side of the and operator is False… (Review Question Multiple Choice 6)
  - and, or and not are … operators (Review Question Multiple Choice 6)
  - A compound Boolean expression created with the or operator is true… (Review Question True or False 4)
  - When determining whether a number is inside a range (Review Question Short Answer 5)

  - Modifikasjoner kommer til å være oversetting til norsk og ”Kahootisering”
- Dessuten ett helt uannonsert spørsmål
  - Men relatert til kap.3 det også.