Velkommen til gruppetime i IN1000 ☺

29. september 2021

Jessie Yue Guan

Planen for i dag

- Filinnlesing og filutskriving
- Litt forskjellig…
- Litt repetisjon…

Filinnlesing & Filutskriving

Hvorfor lese inn data fra filer?

- Man kan jobbe strukturert med større datamengder
- Man skiller mellom data (hva) og kode (hvordan)
- Man kan teste programmene sine mye lettere
- Man slipper å skrive dataene på nytt fordi de blir lagret og kan hentes fram igjen

Hvordan lese inn linjer fra filer?

- Steg 1: Åpne filen (og velg les)
- Steg 2: Gå gjennom hver linje i filen
- Steg 3: Gjør noe med hver linje i filen
- Steg 4: Lukk filen

```
min_fil = open("mittFilNavn.txt")
for linje in min_fil:
    min_liste.append(linje.strip())
min_fil.close()
```

Hvordan lese inn tabeller fra filer?

- Steg 1: Åpne filen (og velg les)
- Steg 2: Gå gjennom hver rad i filen
- Steg 3: Splitte raden etter kolonne
- Steg 4: Gå gjennom hver rute
- Steg 5: Gjøre noe for hver rute
- Steg 6: Lukk filen

```
min_fil = open("mittFilNavn.csv")
for linje in min_fil:
    rad = linje.strip().split(";")
    for kol in rad:
        min_liste.append(kol)
min_fil.close()
```

Hvordan skrive ut linjer til filer?

- Steg 1: Åpne filen og velg skriv
- Steg 2: Skriv det du vil i filen
- Steg 3: Lukk filen

- **min_fil = open("mittFilNavn.txt", "w")**
- **min_fil.write("La la la\nHa ha ha")**
- **min_fil.close()**

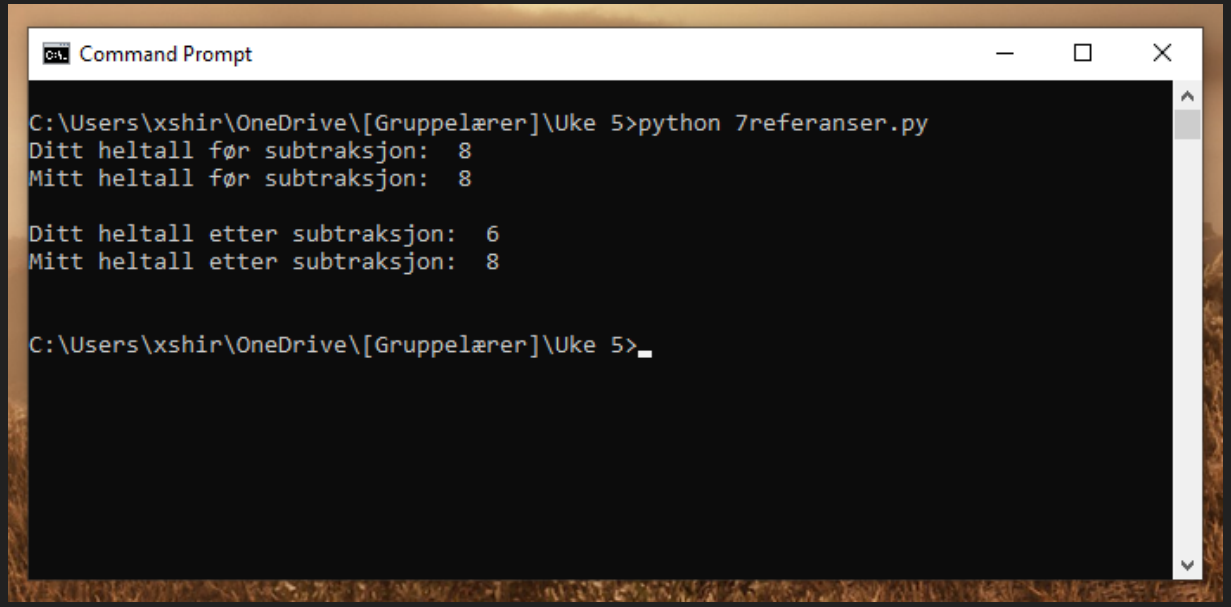
Funksjonene strip() og split()

- Whitespace er en samlebetegnelse for mellomrom, ny linje, tab, og andre "usynlige" tegn
- Disse tegnene blir lest inn fra filen og oversatt som " ", "\n", "\t", osv. i Python
- Du kan fjerne dem på begynnelsen og slutten av en streng ved hjelp av strip()
- Du kan splitte en streng inn i en liste av strenger på dem ved hjelp av split()
- Lag noen test-strenger og prøv deg fram litt selv, bruk print() for å sjekke resultatet

Referanser til primitive variabler

- Hvis du setter en primitiv variabel lik en annen primitiv variabel, og endrer den ene primitive variabelen så vil ikke den andre primitive variabelen endre seg

```
1  mitt_tall = 8
2  ditt_tall = mitt_tall
3
4  print("Ditt heltall før subtraksjon: ", ditt_tall)
5  print("Mitt heltall før subtraksjon: ", mitt_tall)
6  print()
7
8  ditt_tall -=2
9
10 print("Ditt heltall etter subtraksjon: ", ditt_tall)
11 print("Mitt heltall etter subtraksjon: ", mitt_tall)
12 print()
```



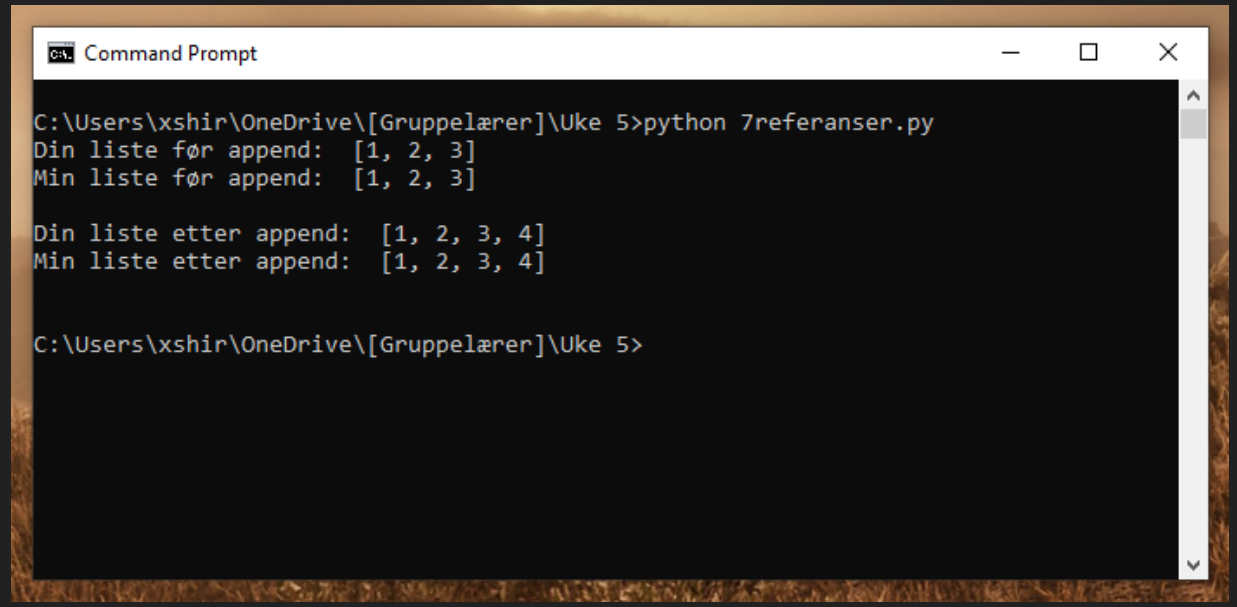
Referanser til primitive variabler

- Forestill deg at minnet til PCen består av et rutenett
- Hver rute er en plass i minnet som har en adresse og kan huske på en verdi om gangen
- Primitive variabler lagrer verdien som ligger inni ruten og ikke adressen til ruten
- Når du aksesserer den primitive variabelen så aksesserer du verdien og ikke adressen
- Hvis du setter en primitiv variabel X til å være lik en annen primitiv variabel Y, også gjør du noen endringer på X men ikke på Y, så vil X endre seg men ikke Y
- Dette er fordi du gjør endringer på verdien og ikke adressen

Referanser til kolleksjoner

- Hvis du setter en kolleksjon lik en annen kolleksjon, og endrer den ene kolleksjonen så vil den andre kolleksjonen også endre seg

```
1   mine_tall = [1, 2, 3]
2   dine_tall = mine_tall
3
4   print("Din liste før append: ", dine_tall)
5   print("Min liste før append: ", mine_tall)
6   print()
7
8   dine_tall.append(4)
9
10  print("Din liste etter append: ", dine_tall)
11  print("Min liste etter append: ", mine_tall)
12  print()
```



Referanser til kolleksjoner

- Forestill deg at minnet til PCen består av et rutenett
- Hver rute er en plass i minnet som har en adresse og kan huske på en verdi om gangen
- Primitive variabler lagrer verdien som ligger inni ruten og ikke adressen til ruten
- Når du aksesserer den primitive variabelen så aksesserer du verdien og ikke adressen
- Hvis du setter en primitiv variabel X til å være lik en annen primitiv variabel Y, også gjør du noen endringer på X men ikke på Y, så vil X endre seg men ikke Y
- Dette er fordi du gjør endringer på verdien og ikke adressen

Hva kan man gjøre med for-løkker?

- For eksempel: Man kan lage et program som oversetter en setning til røverspråk
- (Den dobler hver konsonant i ordet og legger til bokstaven o mellom dem)

```
konsonant = "bcdfghjklmnpqrstvwxyz"
vokal = "aeiouyæøå"

konsonanter = ["b", "c", "d", "f", "g", "h", "j", "k", "l", "m", "n",
"p", "q", "r", "s", "t", "v", "w", "x", "y", "z"]
vokaler = ["a", "e", "i", "o", "u", "y", "æ", "ø", "å"]

setning = input("Skriv inn en setning: ")
roversprak = ""

for bokstav in setning:
    if bokstav in konsonant:
        bokstav = bokstav + "o" + bokstav
    roversprak += bokstav

print(roversprak)
```

Hva kan man gjøre med while-løkker?

- For eksempel: Man kan sørge for at brukerinput er tall

```
1  tall = ""
2
3  while tall.isdigit() == False:
4      tall = input("Skriv inn et tall: ")
5
6  print("Det var et fint tall! :)")
7
8  #Man kan også bruke try-except
9
10 try:
11     tall = int(input("Skriv inn et tall: "))
12     print("Det var et fint tall! :)")
13 except ValueError:
14     print("Dette er ikke et tall")
15
```

Hva kan man gjøre med while-løkker?

- For eksempel: Man kan lage et program som tenker på et tilfeldig tall mellom 1 og 10 som skal la brukeren gjette hva tallet er og gi tilbakemelding om gjetningen er for høyt eller lavt

```
1  import random
2
3  tall = random.randint(1, 11)
4
5  svar = 0
6
7  while (svar != tall):
8      try:
9          svar = int(input("Skriv inn et tall: "))
10         if (svar < tall):
11             print("Tallet er for lite")
12         elif (svar > tall):
13             print("Tallet er for stort")
14     except ValueError:
15         print("Dette var ikke et tall!")
16
17 print("Det er helt riktig!")
```

Hvor får man prosedyrer/funksjoner fra?

- Prosedyrer/funksjoner som er innebygd i Python
  - Print, input, type, osv.
- Prosedyrer/funksjoner som er fra Python Standard Library (MEN bare hvis du vet hva du gjør)
  - Sqrt fra math, ctime fra datetime, randint fra random, osv.
- Prosedyrer/funksjoner som du lager selv

Hva brukes parametere/argumenter og returverdier til?

- Prosedyrer uten parametere
  - Kan bare få inndata vha. input()
  - Dvs. lite kontroll og mye arbeid
  - Kan bare gi utdata vha. print()
  - Dvs. lite gjenbrukbarhet og mye rigiditet
- Prosedyrer med parametere
  - Kan få inndata vha. parametere
  - Dvs. mye kontroll og lite arbeid
  - Kan bare gi utdata vha. print()
  - Dvs. lite gjenbrukbarhet og mye rigiditet

Hva brukes parametere/argumenter og returverdier til?

- Funksjoner uten parametere
  - Kan bare få inndata vha. input()
  - Dvs. lite kontroll og mye arbeid
  - Kan gi utdata vha. return
  - Dvs. mye gjenbrukbarhet og lite rigiditet
- Funksjoner med parametere
  - Kan få inndata vha. parametere
  - Dvs. mye kontroll og lite arbeid
  - Kan gi utdata vha. return
  - Dvs. mye gjenbrukbarhet og lite rigiditet

2-dimensjonale lister

listeception.py

```
hamsterbur = [
    ["Hamtaro", "Spencer", "Max"],
    ["Bobby", "Cupcake", "Cookie"],
    ["Angel", "Per", "Olav"]
]


for liste in hamsterbur:
    for navn in liste:

        print(navn)
    print()

print("Listeindeks 1, navneindeks 2: " + hamsterbur[1][2])

```