Python Sammendrag

Datatyper

Datatyper er typer med data som Python kan håndtere.

Norsk	Engelsk	Beskrivelse
Heltall	Integers (int)	Heltall
Flyttall	Floats (float)	Desimaltall
Sannhetsverdier	Booleans (bool)	True/False
Strenger	Strings (str)	Tekst
Ingen	None	Ingen verdi
Lister	Lists (list)	[Liste,med,ting]

Deklarere variabler

```
# Regne ut hypotenusen i en rettvinklet
#   trekant.
katet1 = 5
katet2 = 12
hypo = (katet1 ** 2 + katet2 ** 2) ** 0.5
print(hypo)
```

Om vi ønsker å forandre datatypen av en variabel kan vi bruke forkortelsen for datatypen. For eksempel om vi ønsker å gjøre heltallet 4 om til en streng kan vi skrive `str(4)`.

Lister

Lister blir brukt til å samle informasjon.

Liste

```
min_liste = ["Navn", 25, ["hei", 12]]
# Henter ut "Navn"
min_liste[0]
# Henter ut "hei"
min_liste[-1][0]
```

List Comprehension

List Comprehension er en kort syntaks for å lage en liste. For eksempel om vi ønsker en liste som inneholder alle kvadrattallene fra 1 til 100 kan vi skrive `[tall ** 2 for tall in range(1,11)]`.

Listemetoder

- `<liste>.pop()`
  - Fjerner det siste elementet i en liste.
- `<liste>.append(<element>)`
  - Legger til elementet bakerst i listen

Hvis vi ønsker å addere på en allerede eksisterende variabel kan vi bruke +=. For eksempel om `var` er en variabel kan vi skrive `var += 2` istedenfor `var = var + 2`.

Formaterte Strenger

Formaterte strenger er en måte å skrive strenger på som gjør det lettere å inkludere andre datatyper i strengen.

Formatert Streng

```
tall = 10
fav_tall = f"Mitt favorittall
    er {tall}."
```

Operasjoner

- Addisjon +
  - Eksempel: `5 + 2 >> 7`
- Subtraksjon -
  - Eksempel: `5 - 2 >> 3`
- Multiplikasjon *
  - Eksempel: `3 * 2 >> 6`
- Divisjon /
  - Eksempel: `3 / 2 >> 1.5`
- Potens **
  - Eksempel: `3 ** 2 >> 9`
- Modulo/rest %
  - Eksempel: `7 % 3 >> 1`

Vi kan også bruke + mellom strenger og lister. Da vil + slå sammen listene eller konkatenere strenger.
Eksempel:
`"Hei" + "sann" >> "Heisann"`

Funksjoner

- `print(<tekst>)`
  - Printer ut tekst.
- `input(<print-tekst>)`
  - Printer ut printtekst og tar inn svar fra brukeren.
- `round(<tall>, <antall desimaler>)`
  - Avrunder tall med et gitt antall desimaler.
- `type(<variabel>)`
  - Gir ut datatypen til en variabel.
- `len(<tekst/liste>)`
  - Gir ut lengden av en liste eller en streng.
- `range(<start>, <slutt>)`
  - Brukes i for-løkker for å iterere fra og med start til, men ikke med slutt. Om vi bare tar inn slutt vil range begynne på 0.

Bruker Input

```
# Tar inn tommer fra brukeren.
tommer = float(input("Oppgi tommer:"))
#  Konverterer til cm.
cm = round(tommer/0.3937, 3)
# Skriver ut resultatet i cm.
print("Dette er " + str(cm) + " i cm.")
```

Egendefinerte Funksjoner

Om vi gjentar biter av koden vår flere ganger kan vi samle dette opp med en funksjon.

Pseudokode Funksjon

```
def <funksjons navn>(<parametere>):
    '''<dokumentasjonsstreng>'''
    <gjoor ting>
    return <returner dette>
```

Funksjon

```
def boks(hooyde, bredde, lengde=1):
    '''
    Regner ut volum av en boks.
    Standardargument for lengde er 1.
    '''
    volum = hooyde * bredde * lengde
    return volum
```

Python vil hoppe ut av en funksjon når den møter på ett `return` nøkkelord.

If-Setninger

If-setninger blir brukt om vi ønsker at koden vår skal gjøre forskjellige ting i ulike situasjoner. Det er tre nøkkelord som er involvert:

- `if <kondisjon_if>`: - Hvis kondisjon_if er opprettholt, gjør det i if-blokken.
- `elif <kondisjon_elif>`: - Hvis kondisjon_if ikke er opprettholt, men kondisjon_elif er opprettholt, gjør det i elif-blokken.
- `else`: - Hvis ingen av kondisjonene over er opprettholt, gjør det i else-blokken.

If-Setning

```
# Tester hva resten er om
#   vi deler paa tre.
tall = input("Oppgi et heltall: ")
tall = int(tall)
if tall % 3 == 0:
    print("Tallet har rest 0.")
elif tall % 3 == 1:
    print("Tallet har rest 1.")
else:
    print("Tallet har rest 2.")
```

For-Løkker

For-løkker brukes når man ønsker å gjenta lignende kode flere ganger. For å lage en for-løkke bruker vi `for` nøkkelordet. For å iterere over tall bruker vi `range(<heltall>)`.

Fibonacci Tall: For

```
# Lager en liste med de fem foorste
#  Fibonacci tallene.
n = 5
fib = []
for i in range(n):
    if i <= 1:
        fib.append(1)
    else:
        fib.append(fib[-1]+fib[-2])
```

Med for-løkker kan vi iterere over lister og bokstaver i strenger. For eksempel:

- `for tall in [1, 3, 5, 6]:`
- `for bokstav in "en streng":`

While-Løkker

While-løkker blir brukt om man ønsker å kjøre kode så lenge en kondisjon er opprettholdt. For å lage en while-løkke bruker vi `while` nøkkelordet.

Fibonacci Tall: While

```
# Lager en liste med Fibonacci
#  tallene under 100
n = 100
fib_mindre_enn_n = True
fib = [1, 1]
while fib_mindre_enn_n:
    neste_fib = fib[-1] + fib[-2]
    fib.append(neste_fib)
    if neste_fib >= n:
        fib_mindre_enn_n = False
        fib.pop()
print(fib)
```

Importering av Moduler

Vi kan få tilgang til flere funksjoner og objekter ved å importere moduler. Moduler vi skal bruke i kurset er

- `math`
  - Inneholder de vanligste matematikkoperasjonene, f.eks. `cos`, `sin`, `sinh`, `exp`, `pi`, `sqrt`.
- `numpy`
  - Inneholder vektorer og matriser og de vanligste vektor og matrisefunksjonene.
- `matplotlib.pyplot`
  - Gjør at vi kan tegne grafer i Python.

Importering av Moduler

```
# Importer numpy ved aa bruke aliaset np.
import numpy as np
np.array([1,3,5])

# Importerer mattepakken.
import math
math.cos(math.pi)

# Importerer sqrt fra matte.
from math import sqrt
sqrt(4)
```

Numpy

```
# Importer numpy som np
import numpy as np
# Lager 2 vektorer
vektor1 = np.array([1, 2, 5])
vektor2 = np.array([0, 3, 5])
# Skalarmultiplikasjon med 3
3 * vektor1
# Summer de to vektorene
vektor1 + vektor2
# Lage en matrise
matrise = np.array([[1, 2], [3, 4]])
```

Matplotlib

```
# Importer matplotlib.pyplot som plt
import matplotlib.pyplot as plt
# Lager en vektor med tall fra 0 til 1
xliste = [0.1 * n for n in range(11)]
# Tar opphooyer verdiene i tredje
yliste = [tall ** 3 for tall in xliste]
# Lager en graf
plt.plot(xliste, yliste)
# Gjoor at pyton viser grafen
plt.show()
```