

# PYTHON KOMMANDOKART

PROFAG  
UNIVERSITETET I OSLO

## GRUNNLEGGENDE PYTHON:

### - Tips til feilsøking:

- Les og prøv å tolke feilmeldingen! De to nederste linjene viser ofte plassering av feilen og type feil.
- Det kan være feil i linja over den feilmeldingen foreslår.
- Skriv ut variabler underveis – da ser du hva du har å gjøre med.
- Kommenter ut snutter i programmet for å kjøre utvalgte deler.
- Spør noen andre om å lese gjennom programmet ditt.
- Bruk dokumentasjon på nettet flittig.

### - Generelt:

# kommentar	Alt etter # vil bli oversett av programmet
Triple anførselstegn på hver side av koden '\n'	Kommentere ut kode over flere linjer linjeskift

### - Datatyper:

Tilordning av verdi til en variabel er gitt ved:

a = 1	heltall
b = 1.0	flyttall
c = "hei" eller c = 'hei'	streng (tekst)
d = True eller d = False	boolsk

For å omgjøre en variabeltype til en annen:

float(1)	output: 1.0
int(1.0)	output: 1
string(1.0)	output: '1.0'

Husk at det er forskjell på teksten '1' og tallet 1.

### - Output og input:

print('hei')	output: 'hei'
a = input()	lagrer input som tekst i a
a = float(a)	Konverterer a til flyttall

## PROGRAMFLYT

### - Beslutninger

```
if <betingelse>:  
    <gjør dette>  
elif <betingelse>:  
    <gjør noe annet>  
else:  
    <gjør noe annet>
```

Bytt ut det som står inni <betingelse> med en logisk betingelse (som f.eks.  $a < 1$ ), og <gjør dette> med en beslutning (f.eks. `print(a)`).

### - While-løkker:

```
while <betingelse>:  
    <gjør dette>
```

### Eksempel:

```
i = 0  
while i < 5:  
    print(i)  
    i = i + 1
```

### - For-løkker:

```
for <tellevariabel> in range(fra, til, steglengde):  
    <gjør dette>
```

### Eksempel:

```
a = 2  
for i in range(10):  
    print(a**i)
```

Hvis vi bruker henholdsvis to argumenter eller ett argument, fungerer `range` som `range(fra, til, steglengde = 1)` og `range(fra = 0, til, steglengde = 1)`

### - Funksjoner:

```
def funksjonsnavn(parameters1, parameters2,...):  
    return verdi
```

### Eksempel:

```
def f(x):  
    return x**2
```

## LISTER OG ARRAYER

### - Lister

Lister lages ved bruk av klammeparentes: [element1, element2].

### Legge til

Legge til element	listenavn.append(element)
Eksempel:	tall.append(3)

### Slette

Sletter første element i lista med gitt navn	listenavn.remove(element)
Slette element nr. a	del listenavn[a]
Slette element nr. a	listenavn.pop(a)

### Telle

Lengden på lista	len(listenavn)
Antall av et visst element	count(listenavn(elementnavn))

### Sortere

Reversere lista	listenavn.reverse()
Sorte lista fra minst til størst	listenavn.sort()

### Finne og bruke elementer

Hente element nummer a (fra 0)	listenavn[a]
Hente element a til b	listenavn[a:b]
Returnerer True hvis element er i liste	element in listenavn

### - Arrayer

For å lage arrayer, må et nytt bibliotek importeres:  
`from pylab import *`

### Opprette array

Fra liste til array	array(listenavn)
array med n 0-er	zeros(n)
array med n jevnt fordelte tall fra a til b	linspace(a,b,n)

### Finne og bruke elementer

Hente element nummer a (fra 0)	arraynavn[a]
Hente element a til b	listenavn[a:b]

### Matematiske operasjoner

Skalarprodukt av array A og B	dot(A,B)
Kryssprodukt av A og B	cross(A,B)

### - tupler

Tupler lages med vanlig rund parentes.  
Tupler er statiske datasamlinger som ikke kan endres.  
Eksempel: `tuppel = (1, 2, 'a', 'b')`

# PYTHON KOMMANDOKART

PROFAG  
UNIVERSITETET I OSLO

## OPERATORER

Eksemplene bruker  $a=10$  og  $b=20$ :

### - Aritmetiske operatører

Op.	Beskrivelse	Eksempel
+	Addisjon	$a + b$ gir output: 30
-	Subtraksjon	$a - b$ gir output: -10
*	Multiplikasjon	$a * b$ gir output: 200
/	Divisjon	$b / a$ gir output: 2
%	Modulus	$a \% b$ gir output: 0
**	EkspONENT	$a ** b$ gir output: $10^{20}$
//	Heltallsdivisjon	$9 // 2$ gir output: 4

### - Logiske operatører

$10 == 10$	Logisk lik
$10 != 11$	Ikke lik
$12 > 10$	Større enn
$10 >= 10$	Større enn eller lik
$10 < 11$	Mindre enn
$a < b$ and $a > c$	begge kriterier sanne
$a < b$ or $a > c$	én av kriteriene sanne
$a \text{ not } b$	gir True hvis $a$ ikke er $b$

### - Tilfeldige tall

Tilfeldige tall trenger biblioteket *pylab*:

```
from pylab import *
```

<code>randint(a, b)</code>	Tilfeldig heltall fra og med $a$ til $b$ .
<code>uniform(a, b)</code>	Tilfeldig flyttall mellom $a$ og $b$

## PLOTTING OG DATABEHANDLING

### - Plotting

Plotting trenger biblioteket *pylab*:

```
from pylab import *
```

Funksjon	Beskrivelse
<code>plot(x,y)</code>	Plotter $y$ mot $x$ .
<code>show()</code>	Datamaskinen viser plottet
<code>title('tittel')</code>	Tittel på plottet
<code>xlabel('x-aksetittel')</code>	Beskrivelse av $x$ -aksen
<code>ylabel('y-aksetittel')</code>	Beskrivelse av $y$ -aksen
<code>legend(['graf1', 'graf2'])</code>	Liste med merkelapper på grafene
<code>xlim(fra, til)</code>	Definisjonsverdi ( $x$ -aksen)
<code>ylim(fra, til)</code>	Verdimengde ( $y$ -aksen)
<code>grid()</code>	Skru på rutenett
<code>axhline(y=0, color='black')</code>	$x$ -akse
<code>axvline(x=0, color='black')</code>	$y$ -akse

`subplot(rader, kolonner, nr.)` Separate plott i samme bilde

### - Lese fil

<code>fil = open("filnavn.type", "r")</code>	Åpner og leser ( $r = \text{read}$ ) fila
<code>innhold = fil.read()</code>	Lagrer innholdet som streng
<code>readline()</code>	Leser/hopper over ei linje
<code>fil.close()</code>	Lukker fila etter bruk

`loadtxt('filnavn.type', float)` Lager array av en fil vha. *pylab*

## STRENGER

Strenger og output av strenger kan manipuleres med en rekke operasjoner. Eksemplene nedenfor benytter denne variabelen: `tekst = 'Hei, verden!'`

### Finne og bruke elementer

<code>tekst[2]</code>	output: 'i'
<code>tekst[4]</code>	output: ''
<code>tekst[1:3]</code>	output: 'ei'

Husk at mellomrom også er et tegn, og husk at indeksering starter på 0!

### Dele opp strenger

<code>tekst.split(' ')</code>	output: ['Hei,', 'verden!']
<code>tekst.split(',')</code>	output: ['Hei', 'verden!']

Tegnet i parentes er *separatoren* vi ønsker å splitte med.

### Operatører på strenger

Vi bruker følgende strenger til eksemplene nedenfor:

```
a = ['Hallo'] og b = ['verden!']
```

Oper.	Beskrivelse	Eksempel
+	Trekker sammen strengene	$a + b$ output: 'Halloverden'
*	Repeterer strengen	$a * 2$ output: 'HalloHallo'
in	Inneholdes i lista	$r \text{ in } b$ output: True
%	Formatering av strenger	Eksempler følger nedenfor

### Strengformatering

Symbol	Betydning
<code>%f</code>	formatering av flyttall
Generell syntaks:	'tekst %<ant. desimaler>f' % (tall)
Eks.: <code>'%.2f' % (3.14159)</code>	output: 3.14
<code>%d</code>	formatering av heltall
Generell syntaks:	'tekst %d' % (tall)
Eksempel: <code>'%d' % (42)</code>	output: 42
<code>%s</code>	formatering av streng
Generell syntaks:	'tekst %<plass mot høyre>s' % (streng)
Eks.: <code>'%10s' % ('Knut')</code>	output: '    Knut'
Eks.: <code>'%-10s' % ('Knut')</code>	output: 'Knut    '

Et alternativ til strengformatering av tall er funksjonen *round(tall, antall desimaler)*.