Python på 1-2-3

Skrive til skjerm

Vanlig tekst

```
print('<tekst>')
```

Eksempel:

```
1  print('Matematikk')
```

Resultat:

```
Matematikk
```

Tekst og variabler

```
print(f'<tekst> {<variabel>} <tekst>')
```

Eksempel:

```
1  tall = 2
2  print(f'Variabelen tall har\
   verdien {tall}.')
```

Resultat:

```
Variabelen tall har verdien 2.
```

Avrunding

```
print (f'<tekst> {<variabel>}:.\
<desimaler>f'.)
```

Eksempel:

```
1  svar = 11/7
2  print(f'Variabelen svar har\
   verdien {svar:.3f}.')
```

Resultat:

```
Variabelen svar har verdien 1.571.
```

Standardform

```
print (f'<tekst> {<variabel>:.\
<desimaler>e}.')
```

Eksempel:

```
1  svar = 7/111
2  print(f'Variabelen svar har\
   verdien {svar:.3e}.')
```

Resultat:

```
Variabelen svar har verdien 6.306e-02.
```

For hånd skriver vi $6{,}306 \cdot 10^{-2}$.

Hente inn opplysninger fra brukeren

Eksempler:

```
navn = input('Hva heter du?')
```

eller

```
tall = int(input('Skriv inn et heltall: '))
```

eller

```
tall = float(input('Skriv inn et tall: '))
```

Matematiske operatorer

Operator	Betydning	Eksempel	Resultat
+	addisjon	3 + 4	7
-	subtraksjon	3 - 4	-1
*	multiplikasjon	1.5 * 4	6.0
/	divisjon	5 / 2	2.5
**	potens	2 ** 3	8
//	heltallsdivisjon	34 // 7	4
%	rest	34 % 7	6

Variabler

Opprette variabel

```
tall = 4
```

Oppdatere variabel

```
tall = tall + 1
```

eller

```
tall += 1
```

- Variabler skal begynne med en vanlig bokstav.
- Variabler kan inneholde bokstaver, tall og understreking, _.
- Variabler kan ikke inneholde mellomrom eller noe annet.
- Noen ord er opptatt. Hvis du skriver inn et ord og det endrer farge, betyr det at ordet har en spesiell betydning i programmet, og at det derfor ikke kan brukes som variabelnavn.

Eksempler på akseptable variabler:
a, avvik, gjennomsnittlig_avvik, gjennomsnittligAvvik, avvik1, Avvik_2.

Datatyper

Kode	Datatype	Eksempel
str	streng (tekst)	`'programmering'`
int	heltall	`1153`
float	flyttall (desimaltall)	`2.7182818`
list	liste	`['a', 3, 5, 1, 1, 1, 'b']`
bool	sannhetsverdi (boolsk verdi)	`True, False`
np.array	np-array	`array([2, 1, 4, 3])`

Bygge opp utsagn

Relasjon	Betydning
==	er lik
!=	er ikke lik
<	er mindre enn
<=	er mindre enn eller lik
>	er større enn
>=	er større enn eller lik

Sammensettinger med «not», «and» og «or»:

Eksempel:

```
1  a > 3 or a < 7
```

Eksempel:

```
1  a == 34 and not a < 2
```

Vilkår

```
if    <betingelse>:
      <instruksjon>
elif  <ny betingelse>:
      <instruksjon>
else:
      <instruksjon>
```

Eksempel:

```
1  tall = int (input('Skriv inn\
             et heltall: '))
2  if tall > 0:
3     print ('Du tastet inn\
            et positivt tall.')
4  elif tall == 0:
5     print('Du tastet inn 0.')
6  else:
7     print ('Du tastet inn\
            et negativt tall.')
```

Løkker

for-løkker

Hvis du vet hvor mange ganger en løkke skal kjøre, bruker du for-løkke:

```
for <variabel> in range\
    (<fra>, <til>, <steglengde>):
    <instruksjon>
```

Eksempel:

```
for x in range(0, 10, 2):
    print(x)
```

Resultat:

```
0
2
4
6
8
```

Tabell over `range`-kommadoen:

Kommando	Verdi
`range(0, 5)`	0, 1, 2, 3, 4
`range(5)`	0, 1, 2, 3, 4
`range(2, 5)`	2, 3, 4
`range(0, 5, 2)`	0, 2, 4
`range(10, 5, -1)`	10, 9, 8, 7, 6
`np.arange(1, 3, 0.5)`	1., 1.5, 2., 2.5

while-løkker

Hvis du ikke vet hvor mange ganger en løkke skal kjøre, bruker du while-løkke:

```
<variabel til betingelse>
while <betingelse>:
    <instruksjon>
    <endre variabel>
```

Eksempel:

```
grense = 20
verdi = 1
while verdi < grense:
    print(f'{verdi} er mindre\
    enn {grense}')
    verdi = verdi * 2
```

Resultat:

```
1 er mindre enn 20
2 er mindre enn 20
4 er mindre enn 20
8 er mindre enn 20
16 er mindre enn 20
```

Funksjoner

```
def <funksjonsnavn>(<variabel>):
    return <verdi>
```

Eksempel:

```
def f(x):
    return x**2 - 8*x +1

tall = f(1)
print(tall)

print(f(4))
```

Resultat:

```
-6
-15
```

Lister

Opprette tom liste

```
liste = []
```

Legge til et element i lista

```
liste.append(tall)
```

Plukke ut elementer fra lista

```
liste = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

Kommando	Resultat	Forklaring
`liste[0]`	`'a'`	Første element er nr. 0
`liste[0:3]`	`['a', 'b', 'c']`	Tar ikke med nr. 3 (fjerde)
`liste[:3]`	`['a', 'b', 'c']`	Fra starten til nr. 3
`liste[-1]`	`'h'`	Siste element er nr. −1
`liste[3:6]`	`['d', 'e', 'f']`	Fra nr. 3 til nr. 6
`liste[3:-1]`	`['d', 'e', 'f', 'g']`	Tar ikke med siste element
`liste[3:]`	`['d', 'e', 'f', 'g', 'h']`	Tar med siste element

Eksterne bibliotek

NumPy

```
import numpy as np
```

Eksempel:

```
verdier = np.arange(1, 2, 0.2)
print(verdier)
[1.  1.2 1.4 1.6 1.8]

verdier = np.linspace(0, 8, 5)
print(verdier)
[0.  2.  4.  6.  8.]

verdier = np.zeros(6)
print(verdier)
[0.  0.  0.  0.  0.  0.]
```

Lage array

```
np.arange(<start>, <slutt>,
          <steglengde>)
np.linspace(<start>, <slutt>,
            <antall punkter>)
np.arange(<antall punkter>)
```

Eksempel:

```
import numpy as np
x = np.arrange(0, 1, 0.1)
y = np.linespace(0, 1, 11)
z = np.zeros(10)

print(f'x:{x}')
print(f'y:{y}')
print(f'z:{z}')
```

Resultat:

```
x:[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
y:[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ]
z:[0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
```

Kommandoer for π og kvadratrot

Kommando	Betydning	Eksempel
`math.pi`	$\pi \approx 3{,}141\,59\ldots$	`2 * math.pi * 6.5`
`math.sqrt()`	kvadratrot	`math.sqrt(88)`

Random

```
import random
```

Eksempel:

```
# Simulerer terningkast
terning = random.randint(1,6)
# Tilfeldig desimaltall\
mellom 0 og 1.
tilfeldig_tall = random.random()
```

Microbit

Eksempel:

```
# Importerer microbit.
import microbit as mb
# Viser smileansikt.
mb.display.show(mb.Image.HAPPY)
# Lagrer data fra\
# akselereometeret.
bevegelse_x =\
    mb.accelerometer.get_x()
# Lagrer temperaturen.
```

Matplotlib

Eksempel:

```
# Importerer matplotlib og numpy.
import matplotlib.pyplot as plt
import numpy as np
# Definerer funksjonen f.
def f(x):
    return -x**2 - 8*x - 10
# Oppretter x-verdier.
xstart = -10
xslutt = 10
steg = .1
x = np.arange\
(xstart, xslutt, steg)
# Tegner grafen.
plt.plot(x, f(x))
plt.show()
```