Python Verktøykasse

kodeklubben[trondheim]_

Matte

Python kan brukes som en vanlig kalkulator:

Plus: `2 + 2`

Minus: `3 - 2`

Gange: `3 * 3`

Dele: `3 / 3`

Potens: `2 ** 2`

Vi kan også kombinere operatorer:

`2 + 3 - 1 * 2`

Dette er det samme som:

`2 + 3 - (1 * 2)`

Boolske verdier

Med boolske verdier kan vi representere om noe i vårt program er sant eller usant:

Og

`True and True` → `True`

`True and False` → `False`

`False and False` → `False`

Eller

`True or True` → `True`

`True or False` → `True`

`False or False` → `False`

Ikke

`not True` → `False`

`not False` → `True`

Likhet (`==`)

`True == True` → `True`

`True == False` → `False`

`"Hei" == "Hei"` → `True`

`(2 + 2) == 5` → `False`

Ulikhet (`!=`)

`True != True` → `False`

`True != False` → `True`

`"Hei" != "Hei"` → `False`

`(2 + 2) != 5` → `True`

Kommentarer

Ofte ønsker vi å forklare koden vår. Sånne forklaringer kalles for kommentarer, og lages ved å bruke `#` eller `"""`:

```
# Jeg er en kommentar!

print(10) # Kommentarer kan også skrives ved
# siden av kode.

"""
Jeg er også en kommentar.
Men fordi jeg ble laget med tre apostrof, kan jeg
skrives over flere
linjer.
"""
```

Vi kan bruke hvilke tegn vi ønsker i kommentarer. Selv om de inneholder gyldig Python-kode, blir ikke koden kjørt.

If-setninger

If-setninger lar programmene våre gjør forskellige ting avhengig av en påstand. Syntaksen til if-setninger står slik:

```
if <PÅSTANDEN>:
    <Koden å kjøre hvis påstanden er True>
```

For eksempel:

```
været = "sol"

if været == "sol":
    print("Nå kan vi spise is")
```

Siden `været` variabelen inneholder `"sol"`, er påstanden `været == "sol"` sant. Derfor blir koden kjørt. Her er Boolere-sekjsonen veldig nyttig!

Det er også mulig å kjøre kode hvis påstanden er usant (dvs. hvis den evalureres til `False`). Til dette bruker vi `else`-nøkkelordet:

```
været = "regn"

if været == "sol":
    print("Nå kan vi spise is")
else:
    print("Nå er det lurt med paraply!")
```

Print & Input

Med print-funksjonen kan vi skrive til skjermen:

```
print("Hei computer")
print(123)
```

Input-funksjonen lar oss hente tekst fra de som bruker programmet:

Strenger

Strenger er ord og setninger, og lages ved å bruke "":

```
"Jeg er en streng"
```

Finne lengden på en streng:

```
len("Hei") == 3
```

Gjør strengen større:

```
"hei".upper() == "HEI"
```

Eller mindre:

```
"HEI".lower() == "hei"
```

Sette sammen strenger med pluss:

```
"Kake " + "er deilig" == "Kake er deilig"
```

Vi kan også gange strenger med tall:

```
"Na" * 5 == "NaNaNaNaNa"
```

Lister

Lister bruker vi for å samle flere verdier sammen i et struktur:

```
dager = ["Mandag", "Tirsdag", "Onsdag",
        "Torsdag", "Fredag", "Lørdag", "Søndag"]
```

Lage en tom liste:

```
tom_list = []
```

Finne lengden på en liste:

```
len([]) == 0
len(["epler", "bananaer", "pærer"]) == 3
```

Legge til en verdi:

```
frukter = ["epler", "bananaer"]
frukter.append("appelsiner")
print(frukter) → ["epler", "bananaer",
"appelsiner"]
```

Finne en verdi på et bestemt sted:

```
frukter = ["epler", "bananaer", "pærer"]
frukter[0] == "epler"
frukter[1] == "bananaer"
frukter[2] == "pærer"
```

```
name = input("Hva heter du?")
...Brukeren skriver...
print(name) → Det brukeren skrev blir nå printet
```

Teksten blir alltid lest som streng:

```
ditt_tall = input("Hva er ditt tall?")
print(ditt_tall) → "8"
```

Hvis vi trenger et tall, eller en andre type, kan vi bruke spesielle funksjoner som forvandler verdier:

Lage tall: `int("8")` → `8`

Lage desimaltall: `float("8.5")` → `8.5`

Lage streng: `str(8)` → `"8"`

Lage Boolean: `int("true")` → `True`

Variabler

Med variabler kan vi gi et navn til verdier. Vi bruker `=` for å koble en verdi med et navn. Ikke forvirre det med `==` !

```
min_bursdag = "06/03/2010"
```

Etter at en variabel er blitt deklarert, kan den brukes når som helst, akkurat som et "rått" verdi:

```
melding = "hei"
name = "Jonas"
print(melding.title() + " " + name) → "Hei Jonas"
```

De er også nyttige når vi vil gjenbruke et verdi i flere programområder:

```
dag = "Tirsdag"
print("Programmeringskurset holdes på " + dag)
print(dag + " er ukas beste dag!") →
"Tirsdag er ukas beste dag!"
```

Vi kan også sette dem på nytt:

```
poeng = 5
...Etter en stund i et dataspill...
poeng = poeng + 1
print(poeng) → 6
```

For-løkker

Når vi vil gjøre ting et bestemt antall ganger, uten å måtte gjenta koden vår, kan vi bruke for-løkker. Syntaksen ser sånn ut:

```
for <VARIABEL> in <SEKVENS>:
    """...Gjenta koden som skrives i blokken
    for hver verdi i sekvensen..."""
```

Etter hver løkk går programmet tilbake til den første linjen i løkket. Løkken slutter når den siste sekvensdelen blir nådd.

Gjenta *n*-ganger:

Merk at vi teller fra null!

Erstatte en verdi:

```
frukter = ["epler", "bananaer", "pærer"]
frukter[1] = "jordbær"
print(frukter) -> ["epler", "jordbær", "pærer"]
```

Merk at vi bruker `=` istedenfor `==`.

While-løkker

Disse brukes for å gjenta koden når vi ikke vet hvor mange ganger koden bør gjentas.

Den viktigste delen er ekspressionen etter `while`. Så lenge ekspressionen evalueres til sant (`True`), skal løkken fortsette.

For eksempel, denne løkken kommer aldri til å slutte:

```
while True:
    print("Jeg skal printes ut i evighet")
```

Mens denne printer tall fra 0 til 9:

```
x = 0
while x < 10:
    print(x)
    x += 1
```

Kan du sammenligne den med den tilhørende for-løkken?

Når blir julematen ferdig? Hva skjer hvis `and`'en endres til `or`?

```
ribben_er_ferdig = False
potetene_er_ferdige = False

while not ribben_er_ferdig and
not potenene_er_ferdige:
    print("Julematen er ikke ferdig ennå!")
```

```
for i in range(9):
    print(i)
```

Løkke gjennom en liste:

```
for frukt in ["epler", "bananaer", "pærer"]:
    print(frukt)
```

Dictionary

Dictionary er en struktur som assosierer en "nøkkel" med en verdi:

```
{
    "navn": "Ola",
    "kjønn": "kvinne",
    "født": "2010",
}
```

(I dette eksempelet er `"navn"`, `"kjønn"`, og `"født"` nøkklene; og `"Ola"`, `"kvinne"`, og `"2010"` verdiene.)

De er veldig nyttige når vi vil samle forskellige datapunkter som handler om én ting.

Nøkkelene og valuatene kan være hva som helst!

Lage en tom dictionary:

```
{}
```

Få tilgang til en verdi:

```
min_dict = {"navn": "Ola"}
print(min_dict["navn"])
```

Erstatte en verdi

```
min_dict = {"navn": "Ola"}
min_dict["navn"] = "Jonas"
```

Legge til en verdi

```
min_dict = {"navn": "Ola"}
min_dict["alder"] = 11
print(min_dict["alder"]) → 11
```