Repetisjon

Skrevet av: Ole Kristian Pedersen, Kodeklubben Trondheim

Oversatt av: Stein Olav Romslo

Kurs: Python
Tema: Tekstbasert
Fag: Programmering
Klassetrinn: 8.-10. klasse

Introduksjon

I denne oppgåva skal me repetere litt Python-syntaks, det er ei god blanding av alle tinga du har lært i Python til no.

Kodeblokker

I oppgåvene er nokre døme så korte at me kan skrive dei direkte inn i Python. Her er ei kodeblokk som illustrerer eit kort døme:

```
>>> 1 == 2
False
```

Her tyder `>>>` at Python er klar til å ta imot kode, `1 == 2` er koden, og `False` er svaret.

I andre døme er det ikkje noko svar, men ei utskrift av tekst i staden:

```
>>> print("Hei!")
Hei!
```

Når me ikkje brukar `>>>` i kodeblokkene er det fordi koden er fleire linjer lang. Då er det betre å bruke ei fil:

```
for i in range(5):
    if i == 3:
        print(i)
    else:
        print(2*i)
```

Og når me køyrer denne fila i IDLE får me utskrifta:

```
>>>
0
2
4
3
8
```

Nokre gonger har me døme med input frå brukaren. Då vil teksten brukaren skriv vere grøn, medan det programmet skriv ut vil vere svart:

```
>>>
Kva heiter du? Ada
Hei, Ada!
```

Input og output

Me kan bruke `print()` når me skal skrive ut tekst til brukaren. Koden etter `>>>` er kode me skriv inn i til dømes IDLE, og som køyrast med ein gong.

```
>>> print("Hei, verd")
Hei, verd
```

`input()` brukast når du ynskjer å la brukaren gi input til programmet ditt.

```
>>> number = input("Skriv inn eit tal: ")
Skriv inn eit tal: 15
>>> print("Du skreiv inn: " + str(number))
Du skreiv inn: 15
```

Sjekkliste

Skriv eit program som spør om namnet til brukaren, og så skriv ut ei helsing til brukaren. Det kan til dømes fungere slik:

```
>>>
Hei! Kva heiter du?
Per
Hyggeleg å treffe deg, Per!
```

Dette må du gjere:

- [ ] Spør om namnet til brukaren.

- [ ] Lagre namnet i ein variabel.
- [ ] Skriv ut ei helsing til brukaren som inneheldt namnet brukaren skreiv inn.

if-elif-else

Me brukar `if`, `elif` og `else` for å bestemme kva som skjer i eit program. Etter `if` og `elif` kjem ein test og så `:`, men etter `else` kjem alltid `:` utan nokon test. På linja under `:` skrivast kodeblokka som skal køyrast viss testen er sann (`if` eller `elif`), eller viss alle testane er usanne (`else`).

Hugs at du alltid må starte med ei `if`-setning, og må ha alle `elif`-setningane før ei `else`-blokk. Du *treng ikkje* å bruke verken `elif`-setningar eller ei `else`-blokk viss du ikkje vil.

Til dømes slik:

```
name = "Ada"
if name == "Per":
    print("Per er eit gutenamn")
elif name == "Ada":
    print("Ada er eit jentenamn")
elif name == "Kim":
    print("Kim kan vere både eit gutenamn og eit jentenamn.")
else:
    print("Eg veit ikkje om " + name + " er ein gut eller ei jente.")
```

Sjekkliste

No skal du lage eit program som finn ut kva aldersgruppe brukaren er i: born, ungdom, vaksen eller pensjonist. Du kan sjølv bestemme kor aldersgrensene skal gå. Det kan til dømes fungere slik:

```
>>>
Hei! Kor gamal er du?
77
Du er visst ein pensjonist.
```

Det du må gjere er:

- [ ] Spør om alderen til brukaren.
- [ ] Lagre alderen til ein variabel.
- [ ] Test om alderen er born, ungdom, vaksen eller pensjonist.
- [ ] Skriv ut kva aldersgruppe brukaren er i.

Løkker

for-løkker

Me brukar `for` -løkker når me vil gjere ting fleire gonger.

```
# print Hello three times
for i in range(3):
    print("Hello")
```

Då får me ut:

```
>>>
Hello
Hello
Hello
```

Me kan bruke `for` -løkker når me vil gå gjennom ei liste:

```
# print all elements in the list food_list
food_list = ["eggs", "ham", "spiced ham", "jam"]
for food in food_list:
    print(food)
```

Dette programmet vil skrive ut:

```
>>>
eggs
ham
spiced ham
jam
```

Sjekkliste

No skal du lage ei liste med namn og skrive ut alle namna i lista. Resultatet kan sjå slik ut:

```
>>>
Per
Ada
Kim
```

Dette må du gjere:

- [ ] Lag ei liste med namn.
- [ ] Bruk ei løkke for å gå gjennom lista med namn.
- [ ] Skriv ut kvart namn.

range()

`range()` lagar ei rekkje med tal. Rekka kan brukast til å gjere noko mange gonger ved hjelp av ei `for` - eller `while` -løkke. `range()` tek inn tre argument: `start`, `stop`, `step`.

- `start` fortel kva me skal telje frå.
- `stop` fortel kva me skal telje til, merk at me **ikkje** tel med slutt-talet.
- `step` fortel kor store steg me skal telje med. Me kan til dømes telje med steg på `2` eller steg på `100`.

Sidan rekkja blir laga etter kvart som me tel over den, må rekkja konverterast til ei liste viss me ynskjer å sjå tala i rekkja. Rekkja konverterast til ei liste med `list()`. Her er nokre døme:

```
>>> list(range(1, 10, 1))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(200, 500))
[200, 201, 202, ..., 497, 498, 499]
>>> list(range(0, 50, 5))
[0, 5, 10, 15, 20, 25, 30, 35, 40, 45]
```

`range()` kan brukast på mange måtar, me kan til dømes gå gjennom den og summere alle tala frå 1 til 100:

```
sum = 0
for number in range(1, 101):
    sum += number
print(sum)
```

while-løkker

`while` -løkker har mange ulike bruksområde. Dei kan til dømes brukast når du vil køyre kode inntil noko bestemt skjer:

```
word = ""
while word != "exit":
    print(word)
    word = input("Please write a word: ")
```

Den same løkka kan skrivast slik:

```
while True:
    word = input("Please write a word: ")
    if word == "exit":
        break
    print(word)
```

Sjekkliste

Skriv eit program som summerer alle tala frå `1` til `100` ved hjelp av ei `while` -løkke. Pass på at du får `5050` som svar.

Dette må du gjere:

- [ ] Lag ein variabel som inneheldt summen.
- [ ] Lag ein teljevariabel som inneheldt talet du har kome til.
- [ ] Så lenge teljevariabelen ikkje er større enn 100:
    - [ ] Oppdater summen.
    - [ ] Auk teljevariabelen din med 1.

Funksjonar

Funksjonar let oss gjenbruke kode, og er svært nyttig når me skal programmere meir enn nokre få linjer. Ein funksjon er på forma:

```
def greet(name):
    print("Hei, " + name + "!")

greet("Per")
```

Her har me ein funksjon med namn `greet`, som skriv ut ei helsing. `name` er ein **parameter**, det vil seie at `name` er ein variabel som funksjonen `greet` tek imot. Når me **kallar** funksjonen `greet`, med `greet("Per")` er `"Per"` eit **argument** til funksjonen. Eit argument er den variabelen me gir til funksjonen når me kallar den.

Me kan òg lage funksjonar som returnerer ein verdi. Det vil sjå slik ut:

```
def multiply(x, y):
    product = x*y
    return product
```

Sjekkliste

No skal me lage ein funksjon som adderer to tal. Test at funksjonen din fungerer som dette:

```
>>> sum = add(3, 4)
Fikk inn 3 og 4
>>> print(sum)
7
```

Dette må du gjere:

- [ ] Definer ein funksjon som tek inn to tal som parametrar.
- [ ] Skriv ut tala du fekk inn.
- [ ] Rekne ut summen.
- [ ] Returner summen.