

# ◆ Enkle objekt

*Skrevet av: Ole Kristian Pedersen, Kodeklubben Trondheim*

*Oversatt av: Stein Olav Romslo*

*Kurs: Python*

*Tema: Tekstbasert*

*Fag: Programmering*

*Klassetrinn: 5.-7. klasse, 8.-10. klasse, Videregående skole*

## Introduksjon

I denne oppgåva skal me gi ei enkel innføring til klasser og objekt (orda forklarar me seinare i teksten).

## Ordbøker

Tenk deg at du skal lage ein sirkel, og ynskjer å lage variablar som beskriv radius og farge. Ved hjelp av ordbøker kan dette gjerast slik:

```
circle = {  
    "radius" : 3,  
    "color" : "red"  
}
```

Me kan hente ut og endre variablane:

```
# endrar radiusen til 5  
circle["radius"] = 5  
# skriv ut farga på sirkelen  
print(circle["color"])
```

Kva viss me vil lage ein variabel som reknar ut arealet til sirkelen? Me kan til dømes lage ein funksjon `circle_area()`:

```
import math
def circle_area(circle):
    radius = circle["radius"]
    # Formel:  $A = \pi * r * r$ 
    area = math.pi * radius * radius
    return area
```

Så kallar me funksjonen:

```
print(circle_area(circle))
```

## Klasser og objekt

Me ynskjer å bruke **objekt** i staden for ordbøker i dei neste spela våre. Me kan la eit Circle -objekt ha to variablar: `radius` og `color`. Ei **klasse** er ein slags mal for eit objekt. Klassa fortel oss kva verdiar objektet kan ha. Me kan lage ei Circle -klasse, og så lage eit objekt av typen Circle som me kaller `circle` basert på klassa.

Dette er enklere å forstå med eit døme:

```
# Me lagar Circle-klassa:
class Circle:
    radius = 3
    color = "red"
# Så lagar me circle-objektet
circle = Circle()
```

Me lagar fyrst Circle -klassa, før me så lagar eit Circle -objekt som me kallar for `circle`. Nøkkelordet `class` fortel datamaskina at me lagar ei klasse. Du kan samanlikne det med til dømes `def` som fortel datamaskina at det kjem ein funksjon.

Me kan hente ut og endre variablane til objektet:

```
# Endrar på radiusen
circle.radius = 5
# Skriv ut farga
print(circle.color)
```

No vil me lage ein funksjon som kan rekne ut arealet til sirkelen. Me kan lage denne funksjonen som ein del av klassa:

```
import math

class Circle:
    radius = 3
    color = "red"

    def area(self):
        area = math.pi * self.radius * self.radius
        return area
```

**Legg merke til innrykket av `area()` i dømet over!**

Så kallar me funksjonen:

```
print(circle.area())
```

Du lurar kanskje på kvifor me brukte `self.radius` i funksjonen `area()` ? Det er fordi me seier at me vil bruke `radius`-variabelen som er ein del av klassa. Du må alltid bruke `self` når du skal bruke funksjonar eller variablar du har laga i klassa.

Som me ser no er det ikkje stor skilnad mellom bruk av funksjonar og klasser:

Ordbøker

```
import math

circle = {
    "radius": 3,
    "color": "red"
}

def circle_area(circle):
    area = math.pi * circle["radius"] ** 2
    return area

circle["radius"] = 5
print(circle["color"])
print(circle_area(circle))
```

Klasser

```
import math

class Circle:
    radius = 3
    color = "red"

    def area(self):
        area = math.pi * self.radius ** 2
        return area

circle = Circle()

circle.radius = 5
print(circle.color)
print(circle.area())
```

Difor kjem me til å bruke klasser i dei neste oppgåvene - det er minst like enkelt som ordbøker, og ein kan gjere meir avanserte ting med klasser.

## Frå ordbok til klasser

No kjem det eit program som er skrive ved bruk av ordbøker. Du skal prøve å "omsette" dette til eit program som brukar klasser.

```
rectangle = {
    "width": 3,
    "length": 5,
    "color": "blue"
}

def rectangle_area(rectangle):
    area = rectangle["width"] * rectangle["length"]
    return area

rectangle["width"] = 10
print(rectangle["color"])
print(rectangle_area(rectangle))
```

---

## Test programmet ditt

Programmet over skriv ut det følgjande:

**blue**

50

Pass på at "omsettinga" di gjer det same.

Lisens: CC BY-SA 4.0 (<http://creativecommons.org/licenses/by-sa/4.0/deed>)