Python

◆ Enkle objekter

Skrevet av: Ole Kristian Pedersen, Kodeklubben Trondheim

Kurs: Python
Tema: Tekstbasert
Fag: Programmering
Klassetrinn: 5.-7. klasse, 8.-10. klasse, Videregående skole

Introduksjon

I denne oppgaven skal vi gi en enkel innføring til klasser og objekter (ordene forklares senere i teksten).

Ordbøker

Tenk deg at du skal lage en sirkel, og ønsker å lagre variabler som beskriver radius og farge. Ved hjelp av ordbøker kan dette gjøres slik:

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

Vi kan også hente ut og endre variablene:

```
# endrer radiusen til 5
circle["radius"] = 5

# skriver ut fargen på sirkelen
print(circle["color"])
```

Hva om vi ønsker å lage en variabel som regner ut arealet til sirkelen? Vi kan for eksempel lage en funksjon `circle_area()`:

```
import math

def circle_area(circle):
    radius = circle["radius"]

    # formula: A = pi * r * r
    area = math.pi * radius * radius
    return area
```

Så kaller vi funksjonen:

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

Klasser og objekter

Vi ønsker å bruke **objekter** i stedet for ordbøker i de neste spillene våre. Vi kan la et `Circle` -objekt ha to variable `radius` og `color` . En **klasse** er en slags 'mal' for et objekt. Klassen forteller oss hva slags verdier objektet kan ha. Vi kan lage en `Circle` -klasse, også lage et objekt av typen `Circle` som vi kaller `circle` basert på klassen.

Dette er enklere å forstå med ett eksempel:

```
# Vi lager Circle-klassen:
class Circle:
    radius = 3
    color = "red"

# Så lager vi circle-objektet
circle = Circle()
```

Vi lager først `Circle` -klassen, før vi så lager et `Circle` -objekt som vi kaller for `circle` . `class` -nøkkelordet forteller datamaskinen at nå lager vi en klasse, du kan sammenlikne det med f.eks. `def` som forteller datamaskinen at nå kommer en funksjon.

Vi kan hente ut og endre variablene til objektet:

```
# Endrer på radiusen
circle.radius = 5

# Skriver ut fargen
print(circle.color)
```

Nå ønsker vi å lage en funksjon som kan regne ut arealet til sirkelen. Vi kan lage denne funksjonen som en del av klassen:

```
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 eksempelet over!

Så kaller vi funksjonen:

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

Du lurer kanskje på hvorfor vi brukte `self.radius` i funksjonen `area()` ? Dette er fordi vi sier at vi vil bruke `radius` -variabelen som er en del av klassen. Du må alltid bruke `self` når du skal bruke funksjoner eller variabler du har lagd i klassen.

Som vi ser så er det ikke så stor forskjell mellom bruk av funksjoner 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())
```

Vi kommer dermed til å bruke klasser i de neste oppgavene - det er minst like enkelt som ordbøker, og man kan gjøre mer avanaserte ting med klasser.

Nå kommer det et program som er skrevet ved bruk av ordbøker. Du skal prøve å "oversette" dette til et program som bruker 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 skriver ut det følgende:

```
blue
50
```

Pass på at din "oversettelse" gjør det samme.

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