Objektorientering

vt 25

Outline

Abstraktion av verkligheten

- Mycket i verkligheten kan delas upp i olika egenskaper.
- En person kan ha en ålder, längd och ett namn
- En bil har ett märke, modellnamn, topphastighet, ägare m.m.
- I programmering kan vi *modellera* detta med klasser

En klass

- Här är en representation av en person

```
1 class Person():
2     def __init__(self, name, age, length):
3         self.name = name
4         self.age = age
5         self.length = length
6 
7     def greet(self):
8         print(f"Hello my my is {self.name}")
```

Klassen person

- Vi kan nu skapa *instanser* av klassen `Person`

```
calle = Person("Calle", 32, 181)
enrique = Person("Enrique", 57, 175)
```

- Båda kan nu också hälsa

```
calle.greet()
enrique.greet()
```

Klassen person

- Vi kan komma åt deras egenskaper, eller klassvariabler

```
1 calle.name
2 enrique.age
3 calle.length
```

- Notera att när vi använde `greet()` så har vi paranteser, medan `name` är utan
- Vi kallar `greet()` för en metod.

Outline

Klassen Car

```
1  class Car():
2      def __init__(self, brand, year, color):
3          self.brand = brand
4          self.year = year
5          self.color = color
6      def drive(self):
7          print(self.brand + ": Kör framåt")
8      def honk(self):
9          print(self.brand + ": Tut tut!")
10     def breaking(self):
11         print(self.brand+ ": Bromsar...")
```

Klassen Car

Begrepp

- `class Car` är en ny *datatyp* som vi har skapat
- `__init__` är en *konstruktor*
- `def drive(self):` är en *metod*
- `self.brand` är en *instansvariabel*

Klassen Car

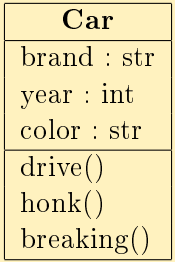
Skapa en instans

```
1 bil1 = Car("Volvo", 2018, "Vit")
2 bil2 = Car("BMW", 2005, "Black")
3
4 bil1.honk()
5 bil2.drive()
```

```
Volvo: Tut tut!
BMW: Kör framåt
```

Klassen Car

Klassdiagram



Klassen Car

self

Som du märkt inleds varje *metod* med parametern `self`. Exempelvis `honk(self)`

```
1     def honk(self):
2         print(self.brand+": Tut tut!")
```

Men `self` dyker inte upp i *metodanropet* senare.

```
1 bil1.honk() # Inget mellan paranteserna
```

Det är för att Python skickar med en *referens* till *instansen* varje gång man anropar en *metod*.

Klassen Car

self

Som du säkert också har märkt så står det `self.` framför alla *instansvariabler*.

```
1     def honk(self):
2         print(self.brand+": Tut tut!")
```

Outline

Övningar

- Ladda ner filen `Klasser 1.py` från classroom.

1. Skapa en ny instans av klassen `Person` med ditt eget namn, ålder och längd.
2. Utveckla metoden `greet` så att den tar emot ett annat namn och skriver ut något i stil med: `"Hello, XXX, my name is Calle"`
3. Utveckla den ytterliggare så att den tar emot en annan instans av klassen `Person` och hälsar på samma sätt.
4. Skriv av klassen `Car`.
5. Lägg till att den har en variabel med namnet `distance = 0`
6. Lägg till att metoden `drive` ökar distance med en godtycklig distans.
7. Lägg till en metod som skriver ut hur långt bilen har kört.
8. Gör uppgifterna 13.1–13.3 i boken. s. 235–