Uge 6

Tirsdag

Dagens tekst

Hvad ved vi allerede om objekter og hvorfor er de smarte?

Object centrisk vs. funktions centrisk

Objekter og klasser

Hvordan programmerer man en klasse?

Objektorienteret programmering

Data og funktionalitet følger hinanden.

Modularitet giver bedre struktur på kompliceret kode.

Functions vs. methods

```
s = “kasper”

# upper s!
u = upper(s)
```

```
# s, upper yourself!
u = s.upper()
```

Standard funktionalitet - under the hood

```
len(“kasper”)

len({1:51, 6:97})

len([4, 1, 3, 5, 7])
```

Overload operators - not your brain!

```
“kasper” + “munch”

4 + 6

[4, 1, 3] + [5, 7]
```

En punkt klasse

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def vector(self):
        return (self.x ** 2 + self.y ** 2) ** 0.5

    def __add__(self, other):
        new_x = self.x + other.x
        new_y = self.y + other.y
        return Point(new_x, new_y)
```

Uge 6

Torsdag

Dagens tekst

- Klasser
- Sidste uges afleveringsopgave
- Debugging

Standard funktionalitet - under the hood

```
class Codon:
    def __init__(self, triplet):
        self.triplet = triplet

    def isStartCodon(self):
        return self.triplet == 'ATG'

    def isSameTriplet(self, triplet):
        return self.triplet == triplet


myCodon = Codon('ATG')
print myCodon.isStartCodon()
print myCodon.isSameTriplet('TGA')
```