GRUPPE 7

Velkommen til
IN1000 - Uke 8

Læringsmål

- Forstå (mer av) hva som skjer bak kulissene når vi oppretter og bruker objekter
- Kjenne til forskjellen på å endre en referansevariabel og å endre objektet variabelen refererer til
- Kunne skrive programmer med samlinger av (referanser til) objekter
- Kunne sette seg inn i og skrive mindre programmer med flere klasser og med objekter som refererer til andre objekter

```python
variabel = None

if variabel is None:
    print("Variabel er None")
```

None

- Brukes som en tom verdi eller manglende data
- Dersom du har en prosedyre, returnerer den automatisk None
- Vi sammenligner ved bruk av “is” og ikke “=“

```python
class Rektangel:
    def __init__(self, lengde, bredde):
        self._lengde = lengde
        self._bredde = bredde

    def areal(self):
        return self._lengde * self._bredde

rek1 = Rektangel(5, 10)
rek2 = Rektangel(3, 7)

print(rek2.areal())
```

Hva er self?

- Sier python hvilket objekt som skal jobbes med

Oppvarmings oppgave

- Skrive en klasse 'Sirkel' i en fil sirkel.py
- Lag en konstruktør som oppretter radius (tas inn som parameter)
- Lag så følgende metoder:
  - En som returnerer diameteren til sirkelen (radius * 2)
  - En som returnerer omkretsen til sirkelen (diameter * pi)
  - En som returnerer arealet til sirkelen (radius^2 * pi)
    - For å lage en potens skriver man n ** m (2 ** 3 = 2^3)
- Opprett en test-fil der du oppretter to sirkel objekter. Skriv ut diameteren og omkretsen til den ene og arealet til den andre.

Løsningsforslag

```python
class Sirkel:
    def __init__(self, radius):
        self._radius = radius

    def diameter(self):
        return self._radius * 2

    def omkrets(self):
        return self.diameter() * 3.14

    def areal(self):
        return (self._radius ** 2) * 3.14

from sirkel import Sirkel

sir1 = Sirkel(5)
sir2 = Sirkel(10)

print(sir1.diameter())
print(sir1.omkrets())
print(sir2.areal())
```

Referanser (pekere)

- Vi har lært at variabler er noe som holder på en verdi:

- string = «Hei, hei»
- tall = 5
- float = 3.14
- boolean = False

- Men når vi snakker om klasser og objekter er variablene en referanse til objektet

Referanser

- Variabler kan sees på som bokser som holder på verdier, mens referanser er små bokser som inneholder adressen til objekter
- Variabler kan inneholde, strings, ints, floats eller booleans (immutable)
- Referanser peker til f.eks. lister, mengder, ordbøker, klasser man lager selv (mutable)

Hva blir konsekvensen av dette?

```
bil = Bil("Audi")
bil2 = bil

bil2.endre_merke("BMW")

print(bil.hent_merke()) # skriver ut BMW
```

Hva blir konsekvensen av dette?

```
tall = 5

tall2 = 5

tall2 += 2

print(tall) # skriver ut 5
```

```python
class Forelder:
    def __init__(self):
        self._barn = None


    def sett_barn(self, barn):
        self._barn = barn


class Barn:
    def __init__(self, fødselsdato, fødselsvekt):
        self._fødselsdato = fødselsdato
        self._fødselsvekt = fødselsvekt


def hovedprogram():
    forelder = Forelder()
    barn = Barn("01012024", 3500)
    forelder.sett_barn(barn)

hovedprogram()
```

Oppgave 1

- Gitt hovedprogrammet nedenfor, tegn datastrukturen slik den ser ut etter linjen `forelder.sett_barn(barn)`. En datastrukturtegning viser opprettede objekter og sammenhengen mellom dem. Tips: eksempler på datastrukturtegninger finnes blant annet i ukas forelesningsslides. Sammenlign deres tegning med Python Tutors.

```python
class Matbutikk:
    def __init__(self):
        self._varer = []
```

Oppgave 2

- Utvid klassen `Matbutikk`, slik at den får en metode for å hente lista `varer`.
- Skrive en metode for å legge til et `Vare`-objekt i lista `varer`.
- Trenger vi flere parametere enn `self` i konstruktøren over? Begrunn svaret.
- Skriv klassen `Vare`, med instansvariabler `vareid` og `varenavn`. init-metoden (konstruktøren) tar inn `vareid` og `varenavn`. `Vare` skal ha to metoder; én for å hente `vareid` og én for å hente `varenavn`.
- Skriv et hovedprogram med en `Matbutikk`, og opprett og legg til tre `Vare`- objekter. Skriv ut `vareid` sammen med `varenavn` for de tre varene på tre forskjellige linjer. Dere blir nødt til å kalle relevante metoder underveis.

```python
class Student:
    def __init__(self, navn, studieprogram):
        self._navn = navn
        self._studieprogram = studieprogram

    def hent_navn(self):
        return self._navn

    def endre_studieprogram(self, nytt_program):
        self._studieprogram = nytt_program
```

Oppgave 3

- Hva er det *self* refererer til i init-metoden (konstruktøren) og i metodene hent_navn og endre_studieprogram?
- Hva slags variabler er *navn*, *studieprogram* og *nytt_program*?
- Hvordan skiller variablene i b) seg fra variablene *_navn* og *_studieprogram*?
- Hvorfor trenger vi *self* i metodene *hent_navn* og *endre_studieprogram*?
- Hvorfor trenger vi ikke å inkludere *self* som argument når vi kaller metodene hent_navn og endre_studieprogram på et Student-objekt?

```python
class A:
    def __init__(self, a):
        self._a = a


class B:
    def __init__(self, b):
        self._b = b


class C:
    def __init__(self, c):
        self._c = c


def hovedprogram():
    c = A("B")
    a = B("C")
    b = C("A")
    b = a
    a = c
    c = b

hovedprogram()
```

Oppgave 4

- Hva refererer henholdsvis variablene *a*, *b* og *c* til i hovedprogrammet?
- Hvorfor bruker vi ikke *self* noe sted i hovedprogrammet?
- Er argumentene vi bruker gyldige? Forklar hvorfor/hvorfor ikke.

```python
class KaffeLatte:
    def __init__(self, ant_espressoshots):
        self._antall_espressoshots = ant_espresso


    def hent_antall_espressoshots(self):
        return self._antall_espressoshots

def hovedprogram():
    bestillinger = []
    for i in range(5):
        dobbel_latte = KaffeLatte(2)
        bestillinger.append(dobbel_latte)

    for bestilling in bestillinger:
        print(bestilling)


hovedprogram()
```

Oppgave 5

- Hva vil skrives ut i programmet? Kom opp med et svar og begrunn hvorfor før dere kjører programmet.
- Endre programmet, slik at linjen `dobbel_latte = KaffeLatte(2)` står under linjen `bestillinger = []` i stedet for inni for-løkka. Hva skrives ut nå? Kom opp med et svar og begrunn hvorfor før dere kjører programmet.
- Hvor mange objekter av klassen KaffeLatte finnes i hovedprogrammet i a), og hvor mange finnes det i hovedprogrammet i b)? Begrunn hvorfor.