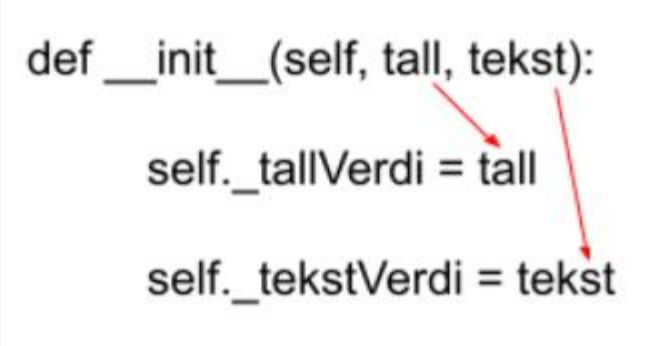
UKE 9 SEMINAR

MAGISKE METODER

__INIT__

- Instansvariablene kan settes fra parameterne:



- Hvis en instansvariabel alltid skal starte med samme verdi, trenger **ikke** denne å være med som parameter

```
def __init__(self, tall, tekst):
    self._tallVerdi = tall
    self._tekstVerdi = tekst
    self._startVerdi = 0
```

- **Navnet på et parameter** har **ikke** noen sammenheng med **navnet på instansvariabelen**

OPPGAVE 1

- Hvilken verdi får instansvariablene?

```
test_person = Person(13, "Kari")

# 1
def __init__(self, alder, navn):
    self._alder = 0
    self._navn = " "

# 2
def __init__(self, alder, navn):
    self._alder = navn
    self._navn = alder
```

```
# 3
def __init__(self, alder, navn):
    self._alder = alder
    self._navn = navn

# 4
def __init__(self, a, b):
    self._alder = a
    self._navn = b

# 5
def __init__(self, alder, navn):
    self._a = alder
    self._n = navn
```

OPPGAVE 2

- Opprett tre instanser av hver klasse med fornuftige verdier

```
class Bil:
def __init__(self, regnr, type, aarsmodell):
        self._regnr = regnr
        self._type = type
        self._aarsmodell = aarsmodell

class Koffert:
    def __init__(self, toalettsakerliste, klesliste):
        self._toalettsaker = toalettsakerliste
        self._klaer = klesliste

class Hund:
    def __init__(self, a, b, c):
        self._alder = a
        self._rase = b
        self._bjeffer_mye = c
        self._metthet = 10
```

«REGLER» FOR NAVNGIVNING

- Liten forbokstav på variabel- og metodenavn
  - Må være i ett ord
  - Bruk _ mellom ord
  - Eks: dette_er_variabelnavnet
  - Eks: dette_er_metodenavnet()
- Stor forbokstav på klassenavn
  - Må være i ett ord
  - Eks: Klassenavn
- Små bokstaver på filnavn
  - Må være i ett ord
  - Eks: filennavn.txt

PRINT VS RETURN

- Print og return har ingenting med hverandre å gjøre

- Eksempel fra oblig
  - "Lag en funksjon/metode som … og returnerer svaret"
    - Hva skal brukes her (print/return)?

- Hvorfor trenger vi return? Kan vi ikke bare bruke print?
  - Det er ikke alt man ønsker å skrive ut på skjermen!
  - Ved print så får man ikke tak i verdien (til å bruke til andre ting senere)!

LØKKER OG RETURN

- Hva gjør return når du har den i en løkke?
- Hva skjer i følgende kodesnutt? Blir dette riktig?
- Poenget med denne funksjonen er at den skal returnere om en liste inneholder et negativt tall.

```
def liste_har_negativt_tall(liste):
    for tall in liste:
        if tall < 0:
            return True
        else:
            return False

liste = [-10, 4, 3, -9, 100]
print(f"liste_har_negativt_tall på listen: {liste} returnerer {liste_har_negativt_tall(liste)}")

liste = [10, 4, 3, -9, 100]
print(f"liste_har_negativt_tall på listen: {liste} returnerer {liste_har_negativt_tall(liste)}")

liste = [10, 4, 3, 9, 100]
print(f"liste_har_negativt_tall på listen: {liste} returnerer {liste_har_negativt_tall(liste)}")
```

MAGISKE METODER

```
def __init__(self):
    pass

def __str__(self):
    pass

def __repr__(self):
    pass

def __eq__(self, other):
    pass

def __ne__(self, other):
    pass

def __hash__(self):
    pass
```

- Det finnes mange magiske metoder, men det er kun __eq__ og __str__ som er pensum

__EQ__

- Hvordan kan vi vite at to tall er like?
  - 42 == 42
  - 42 != 43

- Hvordan kan vi vite at to strenger er like?
  - «hund" == «hund"
  - «hund" != «katt"

__EQ__

- Hvordan vi vite at to objekter er like?

- For eksempel, hvordan kan vi vite at to katte-objekter er like?
  - Er to katte-objekter like hvis de har samme navn?
  - Er to katte-objekter like hvis de har samme alder?
  - Er to katte-objekter like hvis de har samme farger?
  - Er to katte-objekter like hvis de har samme kjønn?

__EQ__

- Svaret er at vi bestemmer, det er opp til programmereren å lage gode ID-er
- Vi kan definere hva som skal til for at to objekter regnes som like ved hjelp av __eq__
- Dette er en spesiell metode som kjøres automatisk når vi bruker == mellom to objekter
- NB!!! __eq__ må alltid ta imot objektet vi skal sammenligne med og returnere True/False!!!

__EQ__

- Hvis vi skal holde styr på et lite antall katter kan vi f.eks. sammenligne navnene

```
class Katt:
def __init__(self, n, a, f, k):
    self._navn = n
    self._alder = a
    self._farger = f
    self._kjonn = k

def hent_navn(self):
    return self._navn

def __eq__(self, annen):
    return self._navn == annen.hent_navn()
```

__EQ__

- Hvis vi skal holde styr på et stort antall katter kan vi bruke et ID-nummer:

```
class Katt:
def __init__(self, id):
    self.id = id

def hent_id(self):
    return self._id

def __eq__(self, annen):
    return self._id == annen.hent_id()
```

__STR__

- Hvordan kan vi vite hvordan vi skal printe et tall?
  - tall = 42
  - print(tall) #42
- Hvordan kan vi vite hvordan vi skal printe en streng?
  - tekst = «katt"
  - print(tekst) #katt

__STR__

- Hvordan kan vi vite hvordan vi skal printe et objekt?

- Hvilke instansvariabler er naturlig å printe for katte-objekter?
  - Hva med navnet til katte-objektet?
  - Hva med alderen til katte-objektet?
  - Hva med kjønnet til katte-objektet?
  - Hva med rasen til katte-objektet?

__STR__

- Svaret er at vi bestemmer, det er opp til programmereren å lage gode print-setninger
- Vi kan definere hvilke instansvariabler som skal vises når vi printer et objekt ved hjelp av __str__
- Dette er en spesiell metode som kjøres automatisk når vi bruker print() eller str() på et objekt
- NB!!! __str__ tar ikke imot noen parametere (bortsett fra self) og må returnere en streng!!!
  - Metoden skal ikke printe noen ting!

__STR__

- Hvis vi skal holde styr på et lite antall katter kan vi returnere bare navnet:

```
class Katt:
def __init__(self, n, a, f, k):
    self._navn = n
    self._alder = a
    self._farger = f
    self._kjonn = k

def __str__(self):
    return self._navn
```

__STR__

- Hvis vi skal holde styr på et stort antall katter kan vi returnere navn og et ID-nummer:

```
class Katt:
def __init__(self, n, id):
    self._navn = n
    self._id = id

def __str__(self):
    return f"{self._id}: {self._navn}"
```