

# DE24- Programming 40 Yhp

OOP

# OOP

- Objektorienterad programmering
  - Paradigm
    - cirka 1960-talet
  - Ett sätt att tänka
  - Systemdesign / arkitektur utifrån objekt
  - Återanvändbar kod+data(state) genom objekt
- OBS: OOP är inte en viss typ av programmeringsspråk
  - det är ett sätt att tänka, lösa / bryta ner problem

# Vad är en klass .. och vad är ett objekt?

- En klass är en mall som man bygger upp kring ett informationsobjekt som man hanterar. Det är alltså något man kan hantera information om tex en kund, order, produkt.
- Den hanterar både informationen och funktionalitet kring informationen.
- När man skall använda en klass skapar man en sk instans av klassen, man kan säga att det är en kopia av mallen som man använder. En instans av klassen kallas även för ett objekt. Därifrån kommer begreppet objektorienterad programmering

# Python class

- I Python definierar man:
  - Funktioner med def
  - Klasser med class
- Objekt (instanser) av en class skapas med hjälp av “()”

```
class Person:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age
```

```
p = Person("Kalle", 23)  
print(p.name)
```

# Python: metoder

- I klassen kan man definiera ett antal metoder
- Varje metod definieras som en vanlig funktion innanför klassen (dvs: med **def**)
- Varje metod kan anropas i en objekt med hjälp av “.” variabelnamn.metod

```
class Person:  
    def sleep(self, minutes):  
        pass
```

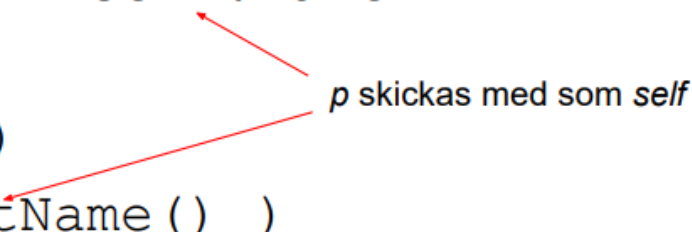
```
mark = Person()  
mark.sleep( 60 )
```

# Python: self

- Vad betyder self?
- Self är Pythons sätt att “hacka” in OOP
- Varje medlem som behöver ha tillgång till objektets data gör det via *self*
- **Som en pekare till objektet (instansen)**
- Python skickar med *self* automatiskt som en parameter när du anropar metoden med “*objekt.function()*”

```
class Person:
    def getName(self):
        return self.name

p = Person()
print( p.getName() )
```



*p* skickas med som *self*

# Python: constructor

- Klassens constructor i Python är en vanlig metod som
  - heter `__init__`
  - tar valfri antal parametrar
  - måste ha *self* som första parameter
- To ensure valid state
  - Går inte att skapa ett objekt utan att vissa saker initialiseras

```
class Person:  
    def __init__( self,name,age)  
        self.name = name  
        self.age = age  
  
p = Person("Stefan", 48)  
print(p.name)
```