

# Python

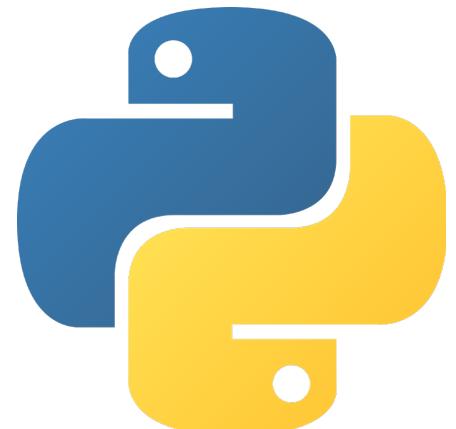
Objektorienterad programmering (och modellering)



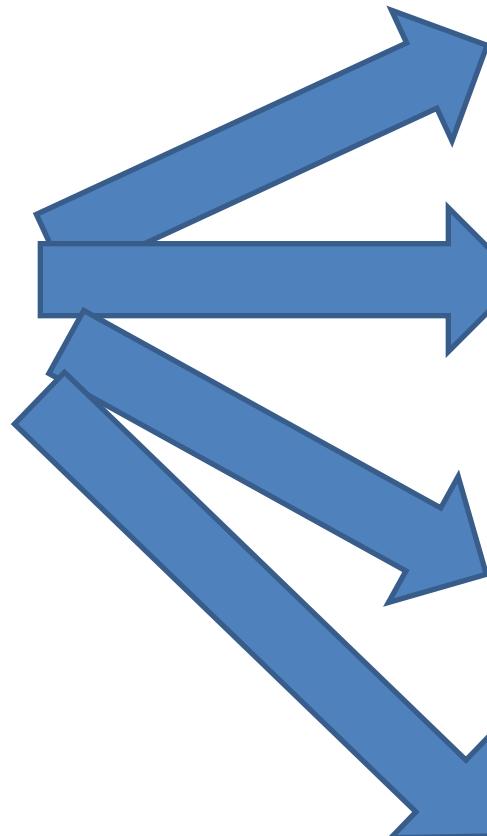
Några frågor innan vi kör igång dagens tillfälle?

# Dagens agenda

- Vad är objektorienterad programmering?
- Hur skiljer sig OOP från funktionsdriven programmering?
- Hur ser detta ut i Python?
- Lite snabba exempel!



# OOP



## Vår 2022 - Termin 2

- [Informationsarkitektur för flerplattformsapplikationer \(DA108A\), 7.5 hp,](#)
- [Systemutveckling och projekt I \(DA336A\), 15 hp,](#)
- [Databasteknik \(DA297A\), 7.5 hp,](#)

## Höst 2022 - Termin 3

- [Data- och informationsvetenskap: Objektorienterad programmering och modellering för IA \(DA361A\), 7.5 hp,](#)
- [Information bortom skärmar \(DA200A\), 7.5 hp,](#)
- [Informationsdesign \(DA456A\), 7.5 hp,](#)
- [Webbtjänster \(DA109A\), 7.5 hp,](#)

## Vår 2023 - Termin 4

- [Utvärderingsmetoder för användarupplevelse \(DA410A\), 7.5 hp,](#)
- [Informationssäkerhet \(DA222A\), 7.5 hp,](#)
- [Flerplattformsapplikationer med webbtekniker \(DA395A\), 7.5 hp,](#)
- [Forskningsmetodik \(DA333A\), 7.5 hp,](#)

## Höst 2023 - Termin 5

- [Examensprojekt: Informationsarkitekt och systemutvecklare \(DA485A\), 15 hp,](#)

## Vår 2024 - Termin 6

- [Data- och informationsvetenskap: Examensarbete \(DA462A\), 15 hp,](#)









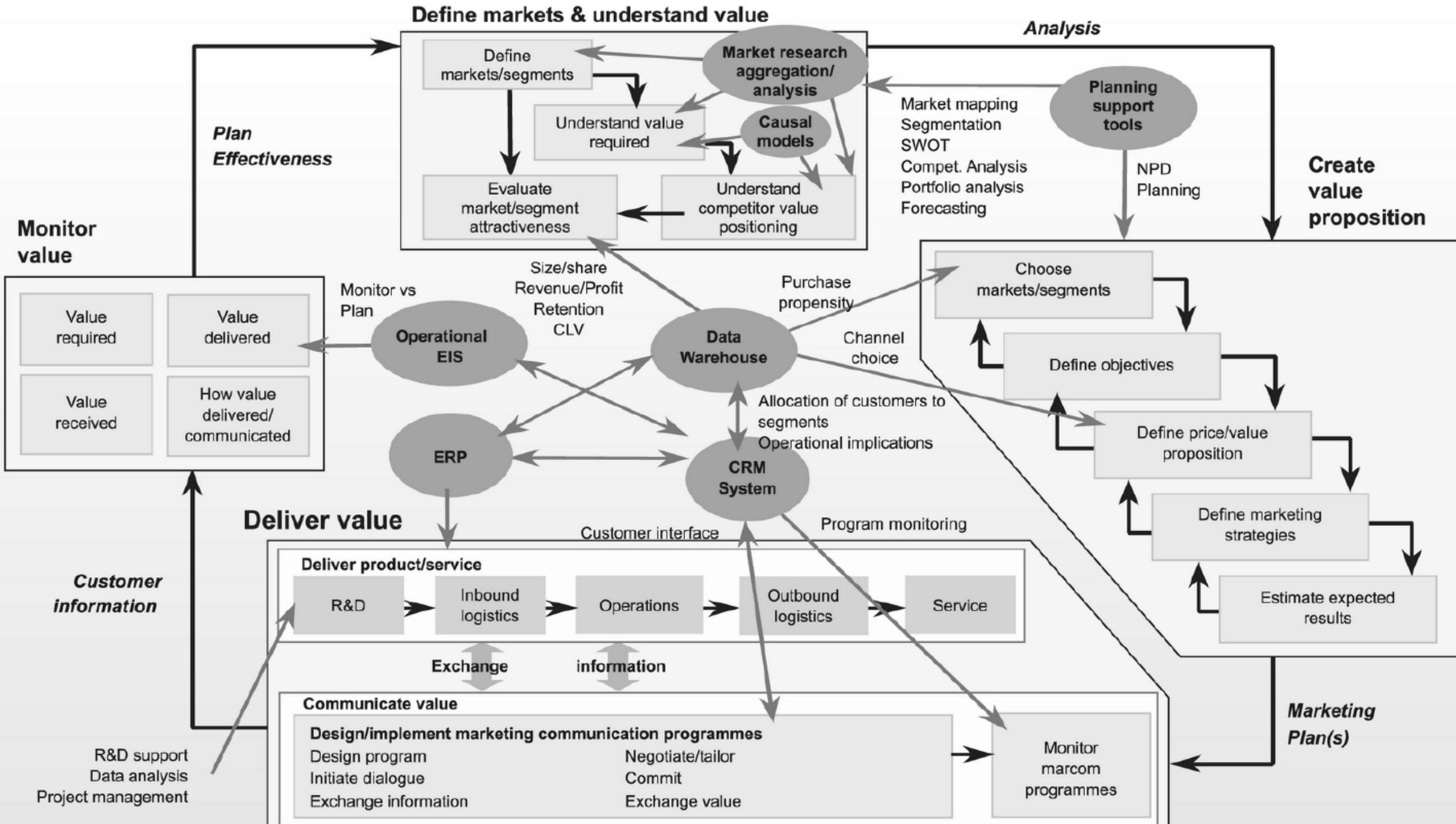


# Hur går ni till väga idag?

När ni jobbar med era projekt?









I Python-kod, beskriv en film?

I Python-kod,  
beskriv en film med skådespelare?

I Python-kod,  
beskriv en film med skådespelare,  
regissör, speltid, betyg, språk?

**THERE'S GOT TO BE A  
BETTER**

**WAY TO DO THAT**

# Datatyper?

# Vi vill modellera världen!

Objektorienterad programmering och modellering

# Vad är objektorienterad programmering

- En programmeringsparadigm, ett sätt hur man skriver & strukturerar kod.
- Vi vill kunna efterlikna den ”verkliga världen” så mycket som möjligt, genom att göra allt till objekt.
- Därför brukar man modellera sitt program innan man programmerar det.

# Att designa ett program

Varför inte en filmsamling?

Vad är en film?

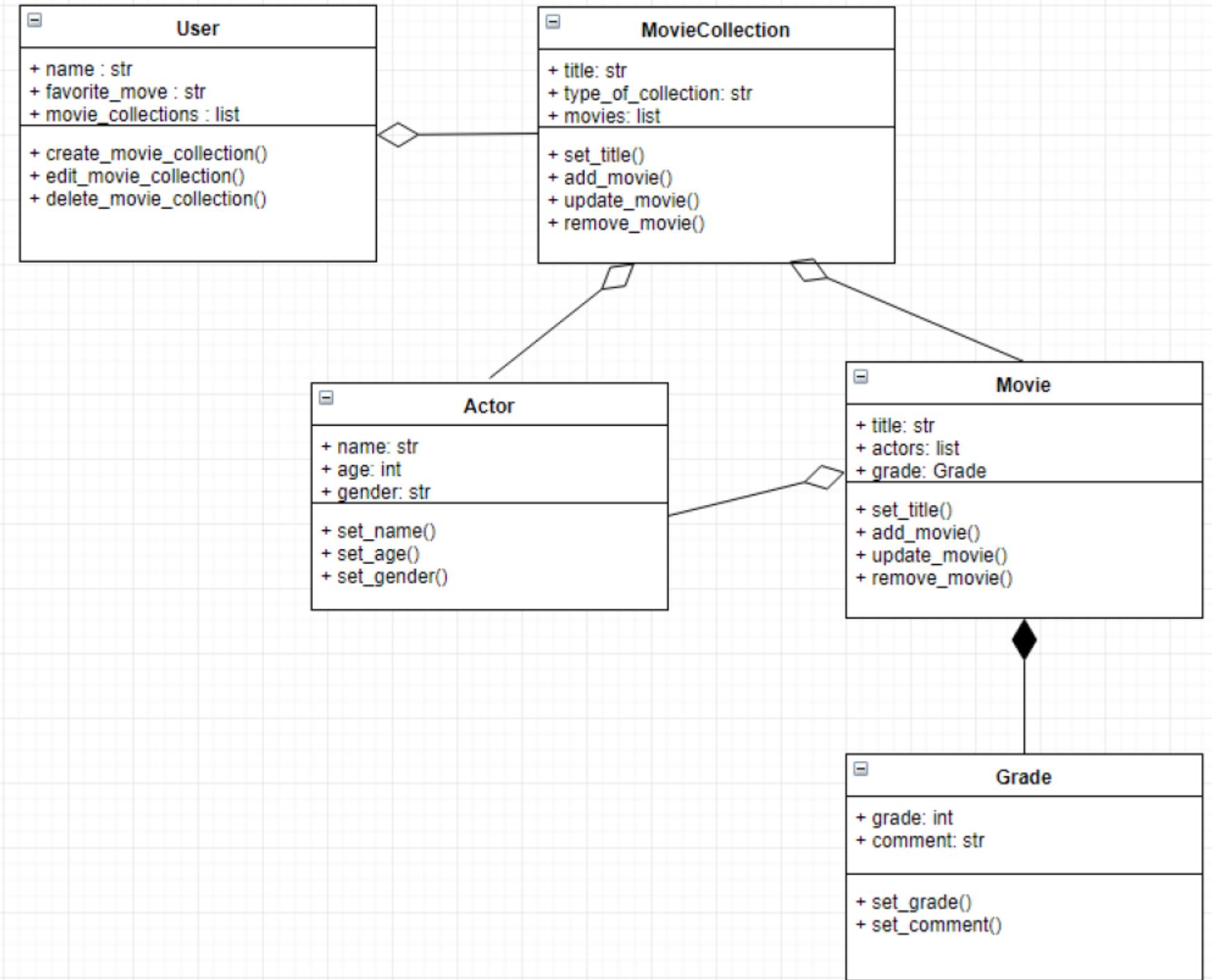
Vad är en  
skådespelare?

Hur relaterar en  
skådespelare till en film?

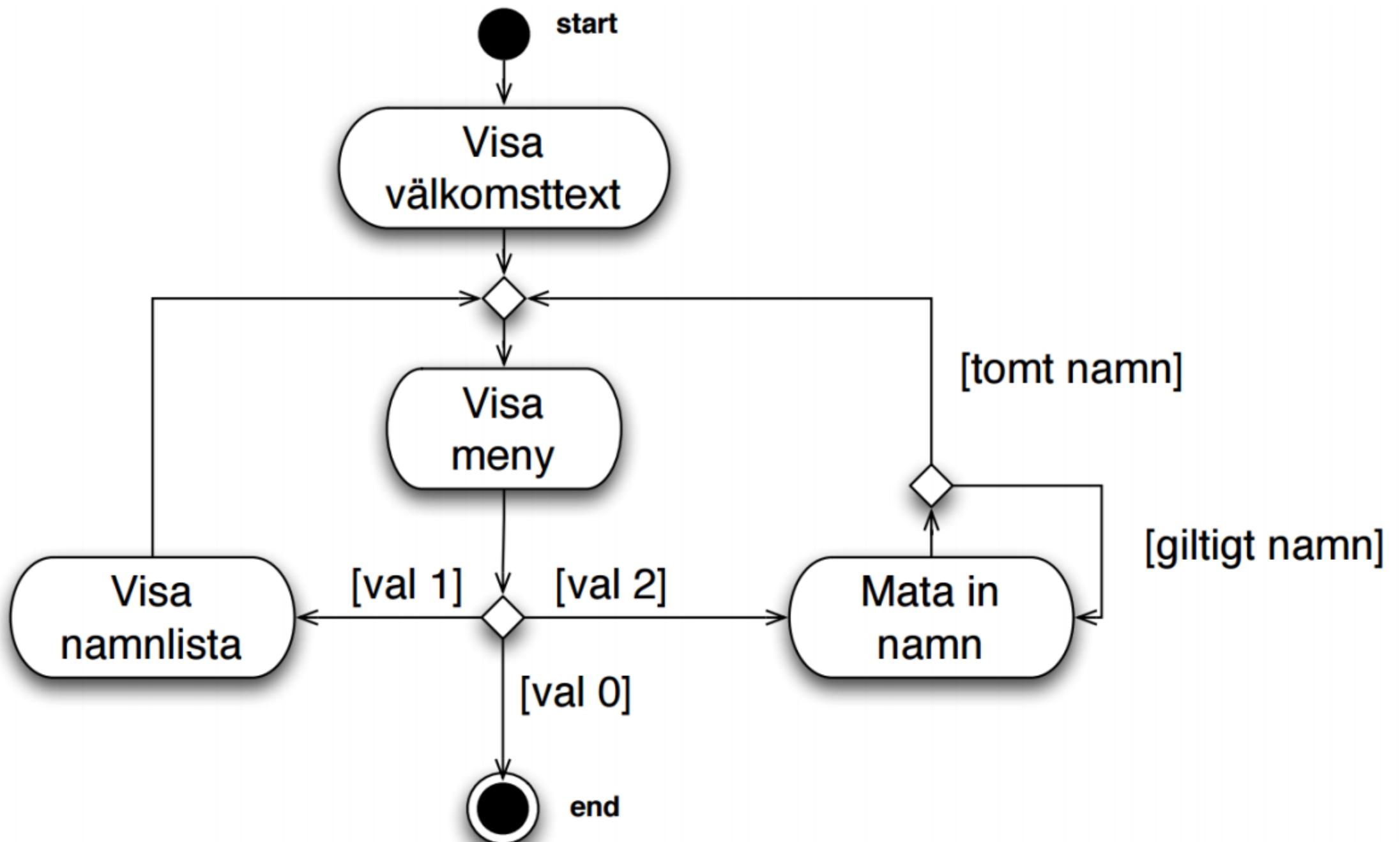
Vad är en  
filmsamling?

# Demo!





**Funktionsdriven programmering är en metod att skriva mjukvara. Den fokuserar på en mjukvarudesign som är centrerad på funktioner och händelser som sker i ett program.**



# Funktionsdriven programmering består av en eller flerafunktioner

- Funktioner arbetar med data som är fristående från funktionen.
- Data skickas mellan funktioner.
- Fokus är på att bygga funktioner som hanterar en mjukvaras data.

# Ex. Funktioner arbetar med data som är fristående från funktionen.

```
def add_movie(movies):
    """
    Lägger till en film i vår filmlista

    Args:
        movies (list) : En lista innehållandes lexikon (filmer)
    """
    print("\nLägg till en film")
    print("-"*40)
    movie = {}
    movie["title"] = input("Titel: ")
    movie["year"] = input("År: ")
    movie["director"] = input("Regissör: ")

    movies.append(movie)

def print_movies(movies):
    """
    Skriver ut alla filmerna i vår filmslista

    Args:
        movies (list) : En lista innehållandes lexikon (filmer)
    """
    print("\n{:<20}{:<6}{:<15}".format("Titel", "År", "Regissör"))
    print("-"*40)
    for movie in movies:
        print(f"{movie['title']:<20}{movie['year']:<6}{movie['director']:<15}")
```

# Ex. Data skickas mellan funktioner.

```
def main():
    """
    Huvudfunktionen i programmet som hantera välkomnande av användaren,
    inläsning av filmer från vår textfil "movies.txt" samt menyn i programmet.
    """
    # 1. Skriva ut en välkomstfras
    welcome()

    # 2. Läsa in alla filmer från en text-fil
    movies = read_movies_from_file("movies.txt")

    # 3. Skriver ut menyn - och beroende på vad användaren väljer - visar/lägger till/tar bort film
    while True:
        print_menu()
        user_choice = input("Ange val: ")

        if user_choice == "1":
            print_movies(movies)
        elif user_choice == "2":
            add_movie(movies)
        elif user_choice == "3":
            delete_movie(movies)
        elif user_choice == "4":
            search_movie(movies)
        elif user_choice == "5":
            save_movies_to_file("movies.txt", movies)
        elif user_choice == "0":
            break
        else:
            print("Du valde inte ett giltigt alternativ, försök igen.")

def save_movies_to_file(file_name, movies):
    """
    Sparar våra filmer till given fil

    Args:
        file_name (str) : Sökvägen till den fil som ska användas
        movies (list) : En lista innehållande lexikon (filmer) som ska sparas
    """
    my_file = open(file_name, "w")
    for movie in movies:
        my_file.write(f"{movie['title']};{movie['year']};{movie['director']}\n")
    my_file.close()
    print("\nFilerna är nu sparade i filen!")
```

**Objektorienterad programmering  
fokuserar på objekt. Objekt skapas från  
abstrakta datatyper och inkapslar data  
och funktioner tillsammans.**

# Objektorienterad programmering

- Ett objekt är en datatyp som innehåller både data och funktioner.
- Data som finns i ett objekt kallas för attribut (eller egenskaper). Funktioner som finns i ett objekt kallas för metoder.

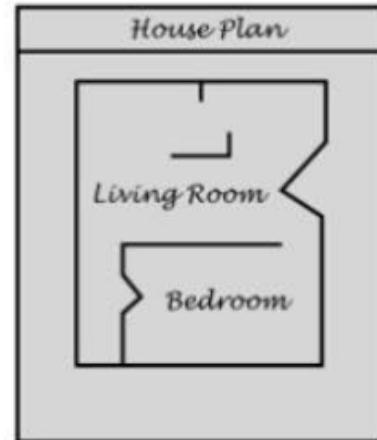
# OBJECTS

# EVERYWHERE

En klass är kod som  
specifierar en datatyp  
med attribut och metoder.

# Class

Blueprint that describes a house



3 objects /  
instances /  
individuals

Instances of the house described by the blueprint



Person

name

p1 : Person

name = “Jane”

p2 : Person

name = “John”

# Person

- name : str

+ get\_name : str

+ set\_name

+ say\_hello

+ \_\_str\_\_ : str

```
1 class Person(object):
2
3     def __init__(self, name):
4         self.name = name
5
6     def get_name(self):
7         return self.name
8
9     def set_name(self, name):
10        self.name = name
11
12    def say_hello(self):
13        print self.name, "says hello!"
14
15    def __str__(self):
16        return self.name
17
```

# Att modellera en stad

*Tänk er typ "sim city"*



TRUMNEY

# Vad hittar vi för saker?

Människa

*Attribut:*

Namn

Ålder

Kön

Pengar

*Metoder:*

Gå

Shoppa

Prata

Åka taxi

Byggnad

*Attribut:*

Gata

Nummer

Storlek

Våningar

Ålder

*Metoder:*

Bygga ut

Rasera

Renovera

Affär

*Attribut:*

Namn

Typ av affär

Adress

Telefonnummer

*Metoder:*

Öppna

Stänga

Sälja

Inventera



# Låt oss fokusera på taxibilen...

Taxibil

*Attribut:*

Förare

Aktiv

Passagerare

Plats

*Metoder:*

Hämta folk

Lämna folk

Byt chaufför

Byt passagerare



Vi har skapat datatypen:

*Taxi*

# Vad är då en klass?

- En klass är en beskrivning av hur ett objekt (t.ex. en taxi-bil, en affär, en människa) ser ut. Alltså:
    - *Vilka attribut som finns*
    - *Vilka metoder som finns*
    - Man brukar rita upp detta i ett så kallat klassdiagram.
      - Taxibil
      - *Attribut:*
        - Förare
        - Aktiv
        - Passagerare
        - Plats
      - *Metoder:*
        - Hämta folk
        - Lämna folk
        - Byt chaufför
        - Byt passagerare
- 

Taxi
+ driver_name: str + on_duty : bool + nr_of_passangers: int + location : list
+ get_driver_name() : return str + set_driver_name(driver_name) : return str + get_on_duty () : return bool + set_on_duty(bool) : return bool + get_nr_of_passangers(): return int + set_nr_of_passangers:(nr): return int + get_location(): return list + set_location(list): return list

# Hur skiljer sig detta från det vi gjort hittills?

- Funktionsdriven programmering
- *Skapar ett steg-för-steg program (funktioner som kallar på varandra i en viss ordning)*
- *Vi är väldigt öppna med vilken data som finns – och hur denna skickas runt i programmet*
- Objektorienterad programmering
- *Fokuserar på att modellera världen på ett sätt som är enkelt att förstå för människan*
- *Istället för att man skickar data mellan olika funktioner har varje objekt sina egna funktioner och attribut – som ibland är hemliga, och objekten visar bara det som de vill visa.*

Vi kollar hur detta ser ut!

```
class Taxi:

    def __init__(self, reg_nr, driver):
        self.reg_nr = reg_nr
        self.driver = driver
        self.color = ["yellow"]
        self.passengers = []
        self.nr_of_passengers = 0
        self.free = True

    def print_info(self):
        print("Taxibil")
        print("=====")
        print("Förare: {}".format(self.driver))
        print("Reg nr: {}".format(self.reg_nr))
        print("Färger: {}, {}".format(", ".join(self.color)))
        print("Antal passagerare: {}".format(self.nr_of_passengers))
        for passenger in self.passengers:
            print("Passagerare: {}".format(passenger))
        if self.free:
            print("Taxin är ledig")
        else:
            print("Taxin är upptagen")

    def add_passengers(self, passengers):
        if self.free:
            self.passengers = passengers
            self.nr_of_passengers = len(passengers)
            self.free = False
            print("Välkomna till taxin")
            for passenger in self.passengers:
                print("Passagerare: {}".format(passenger))
        else:
            print("Taxin är upptagen!")

    def remove_passengers(self):
        print("{} hoppade av taxin.")
        for passenger in self.passengers:
            print("Passagerare: {}".format(passenger))

        self.passengers = []
        self.nr_of_passengers = 0
        self.free = True
```

# Men detta hade vi kunna göra innan ju!

```
def get_people():
    print "Picking up som people..."

def leave_people():
    print "Leaving som people..."

def print_taxis(taxi_car):
    if taxi_car["on_service"]:
        print "{0} is driving in {1} with {2} passangers".format(taxi_car["driver"], taxi_car["location"], taxi_car["nr_of_passengers"])
    else:
        print "{0} the taxi driver isn't working at the moment...".format(taxi_car["driver"])

taxi = {}
taxi["driver"] = "Anton"
taxi["on_service"] = True
taxi["nr_of_passengers"] = 2
taxi["location"] = "Lund" # Eller koordinater, long/lat
taxi["get_people"] = get_people
taxi["leave_people"] = leave_people

# Print taxi info
print_taxis(taxi)
# Picking up people
taxi["get_people"]()
# Ending service
taxi["on_service"] = False
# Print taxi info
print_taxis(taxi)
```

```
>>> ====== RESTART ======
>>>
Anton is driving in Lund with 2 passangers
Picking up som people...
Anton the taxi driver isn't working...
>>> |
```

```
class Taxi:

    def __init__(self, reg_nr, driver):
        self.reg_nr = reg_nr
        self.driver = driver
        self.color = ["yellow"]
        self.passengers = []
        self.nr_of_passengers = 0
        self.free = True

    def print_info(self):
        print("Taxibil")
        print("=====")
        print("Förare: {}".format(self.driver))
        print("Reg nr: {}".format(self.reg_nr))
        print("Färger: {}, {}".format(", ".join(self.color)))
        print("Antal passagerare: {}".format(self.nr_of_passengers))
        for passenger in self.passengers:
            print("Passagerare: {}".format(passenger))
        if self.free:
            print("Taxin är ledig")
        else:
            print("Taxin är upptagen")

    def add_passengers(self, passengers):
        if self.free:
            self.passengers = passengers
            self.nr_of_passengers = len(passengers)
            self.free = False
            print("Välkomna till taxin")
            for passenger in self.passengers:
                print("Passagerare: {}".format(passenger))
        else:
            print("Taxin är upptagen!")

    def remove_passengers(self):
        print("{} hoppade av taxin.")
        for passenger in self.passengers:
            print("Passagerare: {}".format(passenger))

        self.passengers = []
        self.nr_of_passengers = 0
        self.free = True
```

Vill vi bygga fler klasser?

Vad är en film?

Vad är en  
skådespelare?

Hur relaterar en  
skådespelare till en film?

Vad är en  
filmsamling?

