

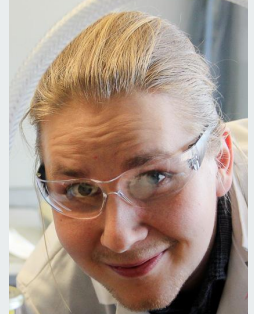


# Databearbetning

Steget innan datavetenskap

Lektion 3 - Listor, Numpy, Matplotlib, Uppg 1

Dennis Biström  
bistromd@arcada.fi



# Upplägg - 8 lektioner, 4 inlämningsuppg



<b>Lektion 1</b> - Kursinfo, verktyg & resurser, Intro till Databearbetning. My first python app	Läxa 1 ut
<b>Lektion 2</b> - Python Moduler och Klasser, My second and third app. Läxa 1 hjälp?	Förhör 1 ut, <del>Läxa 2 ut</del>
<b>Lektion 3</b> - Python Datastrukturer, Numpy & Matplotlib, Uppg 1 start	Förhör 1 in, Uppg 1 ut
<b>Lektion 4</b> - Pandas, Webscraping, Uppg 1 forts	Läxa 3 ut, Förhör 2 ut
<b>Lektion 5</b> - Visualisering, BeautifulSoup, Pandas, Matplotlib Övning, Uppg 2 start	Uppg 1 in, Uppg 2 ut
<b>Lektion 6</b> - Förhör 2 (Numpy, MatPlotLib & Pandas), Ljud och Bilder som data	Förhör 2 in, Uppg 3 ut
<b>Lektion 7</b> - Inlämningsuppg 3 fortsättning, Övning med Bilder och Signaler	Uppg 2 in, Uppg 3 in, Uppg 4 ut
<b>Lektion 8</b> - Inlämningsuppg 4. Kodande & Feedback, Julglögg?	Uppg 4 in

# Förhöret - Feedback?



## Förhöret

1. Problem med itslearning? Rättningslogik? "Fråga av fel typ?"
2. Hur kändes nivån? Nån fråga mycket svårare än andra?
3. Hur kändes innehållet? Var det relevanta frågor om Python?

Hur många använde vilka resurser?

- |    |   |   |
|----|---|---|
| 1. | <a href="#">Socratica</a> tutorialen                        | 5 |
| 2. | Derek Banas " <a href="#">Learn Python in one video</a> "   | 1 |
| 3. | <a href="#">Intro to python for data science</a> - DataCamp | 1 |
| 4. | <a href="#">CH1: Writing your own functions</a> - DataCamp  | 0 |
| 5. | Gör sedan excersizes.ipynb på IL                            | 2 |

# Python in one slide? - Kom ihåg shift+tab

**Python quirks** - Indentation styr koden, försiktigt med mellanslag! Kolontecken efter if och else, *and or not*

**Python** - GPP, bygga webbsidor, analysera data, koda verktyg    # Kommentar, även """ Multiline comment """

**Variabler** - behöver endast ett namn, tolken känner igen typen "Sträng" + str(int) + "."

**Strings** - "Text" eller 'strängar'    **Escape chars** med \ för att skriva t.ex citattecken bland strängar.

**Numror** - Decimaltecken . | j för komplexa tal | int -> float -> complex. Tolk konv till bredare innan aritmetik.

**Aritmetik** - % modulo returnerar resten, // returnerar kvoten, \*\* fungerar som exponent.

**Booleans** - = för tilldelning (assignment), == för utvärdering. Efter str(True) går variabeln inte att använda i logik!

**If elif else** - raw\_input("Mata in en sträng"). (Error handling) med try: except Error: + if else för input validation

**Interaktiv hjälp:**

**dir()** - Se vilka moduler, objekt, klasser och metoder ni har.

**help(someObject.someMethod())** - Få tilläggsinformation om objekt, metod eller funktion

**someObject.someMethod?** - Visa docstring

**jupyter-notebook** - tryck tab 1-4 gånger för att utöka information om det ni håller på att skriva just nu

# Funktioner - Valfria parametrar och programloop



**Funktioner** inkapslad funktionalitet. Definieras med *def*. Tar ofta emot **parametrar**. **Returnerar** värden.

`def funcname(params1[, param2])` Parametrar inom parenteserna. `[]` i docstring betyder valfri parameter

`def g(y, x=0)` Obligatoriska parametrar först, valfria måste nämnas vid namn

`g(3, x=5)` Kalla på funktionen genom att ge obligatoriska värden först, valfria parametrar med namn!

**While loopen** - "Programloop", alltid om ni tänker copypastea funktionalitet, tänk, kan jag loopa?

```
while 1 == 1:                # Alltid sant
    if input == "exit":      # Undantaget
        break               # Stoppar en while loop
    elif input == "restart":  # undantaget som skippar funktionalitet
        continue           # börja en ny iteration
    else:                   # I alla andra fall
        #programlogik      # Gör "som vanligt"
```

# OOP - Metoder, parametrar och returvärden

**Objekt** har ofta metoder()

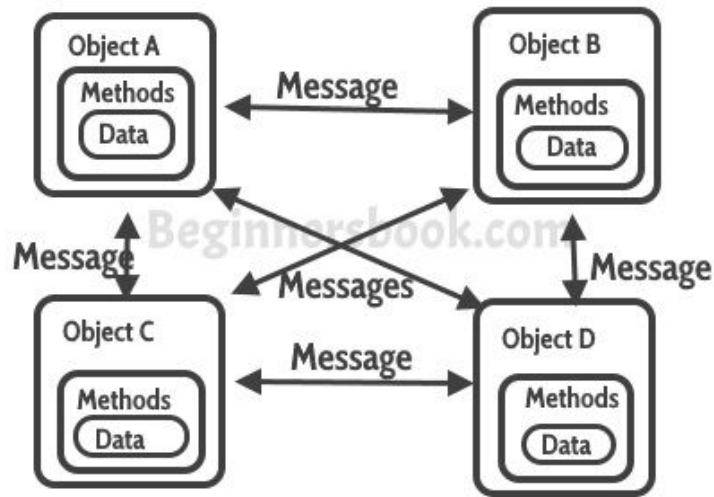
Istället för att skriva all kod efter varann som i en uppsats, så lär vi oss att dela upp koden i olika metoder/funktioner.

Metoder används för att kapsla in beteende. När den är inkapslad kan vi enkelt återanvända samma betende.

Att dela upp funktionalitet i metoder.      *Encapsulation*

Metoder kommunicerar med varandra genom parametrar och returvärden.

```
def multiply(a,b):  
    return (a*b)
```



# Moduler & Klasser - Encapsulation & Message Passing

**Moduler** - En modul innehåller python **Objekt**. Exempel på en moduler `__builtins__` eller `math`  
Objekt i moduler kan innehålla **Klasser**. Många fungerar även som funktioner ex: `datetime.time(6,30)`  
Objekt kan innehålla funktioner, som ofta tar emot **parametrar** ex: `math.cos(90)`

**Metod** = funktion, men vi kallar ofta funktioner inuti objekt för metoder ex: `myTimeVariable.isoformat()`  
Metoder används för att  kapsla in  beteende. När den är inkapslad kan vi enkelt återanvända samma beteende.  
Metoder kommunicerar med varandra genom parametrar och returvärden. Det här kallas Message passing

Klasser innehåller instruktioner om hur man skapar objekt (även funktioner och data). Data i objekt sparas i **fält**

## Exempel:

<code>gamla_bettan</code> är en instans av klassen <code>bil</code>	<code>#klassen bil innehåller instruktioner över hur man skapar bilar</code>
<code>gamla_bettan.color</code>	<code>#Klassen bil innehåller även data som färg och märke</code>
<code>gamla_bettan.accel(10)</code>	<code>#Klassen bil innehåller även metoder, som tar emot parametrar</code>
<code>~/bistromd   \$</code> 82 km/h	<code>#Returvärde för metoden accel() kunde vara hastigheten</code>

# OOP - Klass, Objekt, Instans

En **klass** innehåller instruktioner över hur man skapar ett **objekt**

```
class Car:
    ''' Instruktioner hur man använder klassen '''
    def __init__(self, color, prod_year):
        self.year = prod_year
        self.color = color

    def age(self):
        ''' Räkna ut åldern på bilen '''
        age = (2018-self.year)
        return(age)
```

```
gamla_bettan = Car("red", "1956")
print( gamla_bettan.age() )
```

# Data inuti objekt sparas i **fält**

# **Docstring!**

# **init metod** aka konstruktor

# Vi sparar värden i objektet

# Märk input values vs field name

# Vi lägger till funktionalitet

# Skapa instans av bil

# Använd klassens funktionalitet



# Datastrukturer i python



**Sets()** - Inspirerad av mattan, vi kommer int röra de här ordning

[Socratica Sets](#)

Inga dubletter

Ingen

[Pythonspot Set tutorial](#)

**Tuples()** - Immutable list

[Socratica Tuples](#)

Har ordning

[inLearning](#)

Mindre och snabbare än lists

[Pythonspot Tuple Tutorial](#)

**Listor[]** - Mutable list

[Socratica Lists](#)

Har ordning

[inLearning](#)

Kan ha dubletter

[Pythonspot List Tutorial](#)

**Dictionary{}** - Key:value pair

[Socratica Dictionaries](#)

Bekant från JS objekt?

[inLearning](#)

Ingen ordning

[Pythonspot Dictionaries](#)

# Listor - Klurigheter och tilläggsmoduler



**Listor** - Från andra språk kanske bekant som arrays, i python kallas det här en lista: [1, 2.3, "hej"]

**list[1][3:]** - returnerar all värden efter det fjärde värde i den andra sublistan av list

**Lägg till/modifiera eller ta bort värden:**

list + ["new", 2.3]    del(list[0])    list.append("hej")

Listor har metoder, liksom strängar. **Allting är objekt** men ha koll på ifall du gör string.index eller list.index

Vissa metoder ändrar på instansen list.reverse, andra skapar nya objekt med ändringarna gjorda list[5:6]

## Moduler i form av bibliotek

Numpy för att jobba med **arrays** bl.a. Aritmetik över listor

Matplotlib för att **visualisera** data - Line, Bar, Pie, Histogram etc.

Pandas för att introducera **Data Frames** och därmed bredda listfunktionaliteten i Python

# Idag - Bibliotek, Upppg 1 och Läxa 1 forts.



**Python** recap från senast och fortsättning med datastrukturer, speciellt listor

**Exercises.ipynb** ifall inte redan gjort - Öppna [t.expythonspot.com](https://t.expythonspot.com)

Om du redan gjort, läs Lists-Functions på [Tutorialspoint Python 3 tut](#) (obs Datetime)

Gör övningar på [Datacamp](#)

- Chapter 1 - Python basics
- Chapter 2 - Python lists
- Chapter 3 - Function and packages

# Numpy - flerdimensionella tabeller (arrays)

**Installera numpy med pip** - pip3 install numpy

**import numpy** - för att få access till **numpy.array(list)** ofta `import numpy as np` för att minska syntax

## **Python list to Numpy array**

```
np.array([1,2,3,4,5])
```

## **Numpy Arrays kan sparas**

```
np.save('data.npy',a)
```

## **Matematik över arrays**

```
sin(np.array)
```

[inLearning Numpy Math](#) - Michele Vallisneri

[inLearning Numpy Slicing and Boolean Masks](#) - Charles Kelly

## **Numpy 2D array**

```
np.array([1,2,3,4] , [6,7,8,9])
```

## **och läsas in till/från .npy filer**

```
np.load('data.npy')
```

## **Aritmetik över arrays**

```
y1 = sinx * cosx          y2 = cosx**2 - sinx**2
```

# Matplotlib - Visualisering made easy

**Matplotlib** - `import matplotlib.pyplot as plt` # Använder pyplot paketet från matplotlib

**Line och pie** `plt.plot(x,y)` #`kind='bar'` `kind='barh'` `kind='pie'`

**Färger - colormaps** - sekventiell, divergent, kvalitativ `plt.plot(colormap='Pastell1')`

**Scatter** `plt.scatter(x,y)`

**Skalor** `plt.scale('log')`

**Histogram** - `plt.hist(data, bins=10)` Visualisera distribution av data # standard Python optional variable!

**Anpassa graf** - `plt.xlabel("X-axel")` `.title` `.yticks` #använd aritmetik för att förbättra det visuella meddelandet

Läs om flera alternativ för [pyplot.plot](#) och [pyplot.scatter](#)

[%matplotlib inline](#) - För att få matplotlib o funka inline i jupyter:

# Numpy - betydligt färre for loops

## Matematiska operationer över listor

```
numpy_bmi_array = list_of_weights / list_of_heights ** 2    #Bara en data type i array!
```

Märk också skillnad mellan      `pylist + pylist` #konkatenering      `np_array + np_array` #aritmetik

## Array of booleans:

```
numpy_bmi_array > 20 returnerar en list av booleans:      [False,False]
```

```
numpy_bmi_array[numpy_bmi_array > 20] returnerar:      [ 24.20, 21,24 ] # Praktiskt!
```

**2D numpy arrays:** En förbättrad version av list of lists `array[0,10] * array[2,:]`

`array[row][column]` eller `array[row,col]` t.ex `array[2,3:5]` # Fjärde och femte kolumnen på tredje raden.

**Numpy simple data analytics** `np.mean()`, `np.median()`, `np.std()`, `np.corrcoef()`, `np.column_stack()`

Om du delar upp datan i två np.arrays, nycklar och värden, kan du hänvisa till index med endast nyckelvärden

```
positions = ['GK', 'M', 'A', 'D', ...]      heights = [191, 184, 185, 180, ...]
```

```
gk_heights = heights[positions=='GK']      # Superhändigt!
```

# Idag - Bibliotek, Upppg 1 och Läxa 1 forts.



**Python** recap från senast och fortsättning med datastrukturer, speciellt listor

**Exercises.ipynb** ifall inte redan gjort - Öppna t.ex [pythonspot.com](https://pythonspot.com)

Om du redan gjort, läs Lists-Functions på [Tutorialspoint Python 3 tut](#) (obs Datetime)

Gör övningar på [Datacamp](#)

- Chapter 1 - Python basics
- Chapter 2 - Python lists
- Chapter 3 - Function and packages
- Chapter 4 - Numpy

Klar redan? Kör Data Analysis and Visualization pathen på [DataQuest](#)

- Step 1 är Python Introduction (Lätt?)
- Step 2 är Data Analysis and Visualization dvs Course 1 - Numpy, Course 2 - Matplotlib

# Läxa till imorgon! - Lite numpy



Python o Numpy: [Numpy - Charles K](#) Videorna 2.1 till 3.1

Recap av idag: [Numpy - Michail V](#) Videorna 4.2 till 5.4

[Bra Numpy resurser](#)

[Tutorialspoint Matplotlib](#)




# Inlämningsuppg 1



## Inlämningsuppg 1 - 100k filmratings och ratearnas demografi

1. Läs in användardata, ratingdata och filmdata
2. Filterövning: Visa endast användare som är bibliotekarier och män
3. Kombinationsövning: Medelrating per film, vilka filmer har högst rating?
4. Kombinationsövning2: Vilka filmer har högst rating enligt kön/arbete?

# Lynda och resurser - Kolla även itslearning!

- 
- Cheat sheet: [Anaconda Cheat Sheet - Getting Started - PDF](#)  
[Pandas Cheat Sheet - PDF](#)
- Manual/Docs [Conda package manger - Docs](#)  
[Pandas - QuickStart & Cookbook](#)
- Tutorials (text) [Anaconda Getting Started - User Guide](#)  
[Python - Intro till avancerat - Övningar och förklaringar](#)  
[Pandas tutorial - PythonSpot](#)  
[Intro to data science Numpy, MatPlot & Panda](#)  
[\(Pandas - How do pivot tables work - ExcelCampus\)](#)
- Tutorials (video) [Socratica python tutorial - Youtube](#)  
[Derek Banas - "Learn Python in one video"](#)

# Lynda och resurser2 - Kolla även itslearning!



Interaktiva:

[Intro to \*\*python\*\* for data science - Gratiskurs - DataCamp](#)

[Intro to python for data science - Ch4 - \*\*Numpy\*\* \(DataCamp\)](#)

[Intermediate python for data science - Ch1 - \*\*MatPlotLib\*\* \(DataCamp\)](#)

Lynda:

[6h nybörjarkurs \*\*Python\*\* för datavetenskap med Lillian Pierson - Lynda](#)

[2h intermediate - \*\*Numpy\*\* Data Science Essentials - Charles Kelly](#)

[Intermediate - Ch3: \*\*Numpy\*\*, Ch4: \*\*Pandas\*\*, Ch 9: \*\*matplotlib\*\* - Miki Tebaka](#)

[2h intermediate kurs i \*\*Pandas\*\* med Jonathan Fernandes - Lynda](#)

[2h intermediate \*\*Pandas\*\* för Datavetenskap med Charles Kelly - Lynda](#)

[Big Data Analysis in python using \*\*Numpy\*\* and \*\*Pandas\*\* - Michele Vallisneri](#)

# Lynda och resurser3 - Kolla även itslearning!



Interaktiva roligheter

[The Python Challenge](#)

[Roliga övningar i logisk ordning - Practice Python](#)

[How to think like a Computer Scientist](#)

[CodeSignal - Interaktiva utmaningar, badges, points etc.](#)

[Reddit daily programmer challenges](#)

Vill du vinna 1 miljon \$

[7h gratis tutorial på Kaggle - Känner ni till kaggle?](#)

Python 2 vs Python 3 trubbel, kolla [här](#)

# Jag har redan fallit av kälken! - Brush up ur skills1



1. Kolla [Socratica tutorialen](#) videorna 1-17
2. [Chapter 1-3 - Python for data science \(DataCamp\)](#)
3. [Chapter 1 - Writing python functions \(DataCamp\)](#)
4. [Chapter 4 - Numpy \(DataCamp\)](#)
5. [Chapter 1 - Matplotlib \(DataCamp\)](#)
6. [Chapter 1-4 - Pandas for data science \(Lynda: Kelly\)](#)

Sök hjälp bland resurserna om du kör fast.