


Programmering i Python

F1	Grunder i Python
F2	Loopar, moduler
F3	Funktioner, läsa data
F4	Dictionaries, läsa data från internet
 F5	Gemensamma projekt, objekt/klasser...

Idag

- Repetition dictionaries
- Testa Trafiklabs API
- Objektorienterad programmering
- Testa grafik med Pygame
- Om tid finns: egna projekt

Dictionaries repetition

Dictionaries är en datastruktur i python där man lagrar “nycklar” och värden. Skillnaden mot en lista är att nycklarna inte behöver vara heltal, utan kan också vara strängar.

Dictionaries skrivs i curly brackets { } och kan se ut så här:

```
land = {  
    "namn": "Sverige",  
    "befolkning": 1000000,  
    "språk": ["svenska", "finska"]  
}  
print(land)
```

Dictionaries - hämta eller ändra värde i en dictionary

Inne i dictionary:n så har vi flera par av nycklar och värden (ex "namn": "Sverige"). För att nå värdena så använder vi oss av nycklarna. Ex:

```
print(land["befolkning"])
```

OBS! Om vi försöker hämta ett värde med en nyckel som inte finns, så blir det problem. Man kan kontrollera att en nyckel existerar med:

```
if "befolkning" in land:  
    # Gör det du vill
```

I listor ändrar vi värdet på en viss index genom att skriva lista[index] = värde

I dictionaries är det likadant, men med nyckeln istället för ett index.

```
land["namn"] = "Norge"  
land["befolkning"] = 2000000  
print(land)
```

Dictionaries - Lägga till/radera en nyckel och värde

Lägga till

```
land["latitud"] = 0.2832938
```

```
land["longitud"] = 0.38403
```

Ta bort

```
del dictionary[key]
```

Ex.

```
del land["longitud"]
```

```
del land["latitud"]
```

Skapa en egen SL-skylt

Trafiklab.se innehåller massa API:er för trafik i Sverige och inkluderar SL reseplanerare-API. Du kan skapa ett eget konto där för att få egna 'API-nycklar' (eller använda mina).

Trafiklab har en API som heter "Realtidsinformation" där man kan få ut samma information som SL:s skyltar i lokaltrafiken har.

Man skickar en request till en URL/länk och får en dictionary/JSON som svar.

De använder inte namnen på hållplatser utan ett ID (en siffra) och varje hållplats har ett unikt sådant

Skapa en egen SL-skylt - Få tag på ett ID

SL Platsuppslag

Alla hållplatser har ett ID. Vi kan få det ID:et genom att skriva detta i din browser:

<https://api.sl.se/api2/typeahead.json?key=<NYCKEL>&searchstring=<SÖKORD>&maxresults=1>

Min nyckel är: 3e2085604f874b2aaf2eb73c6f0e256c, sökord får du skriva in själv eller be om från användaren

Exempel:

<https://api.sl.se/api2/typeahead.json?key=3e2085604f874b2aaf2eb73c6f0e256c&searchstring=slussen&maxresults=1>

=> siteid: 9192

Skapa en egen SL-skylt - Få realtidsinformation

SL Realtidsinformation 4

När du har ett ID så kan du få de nästkommande tågen/bussarna (precis som SLs realtidsinformation) genom att skriva detta i din browser:

<https://api.sl.se/api2/realtimedeparturesV4.json?key=<NYCKEL>&siteid=<siteID>&timewindow=5>

Nyckeln är: 91dd5c34135f47d5afdfd08592ea2987, siteID får du tag på med SL Platsuppslag.

Exempel:

<https://api.sl.se/api2/realtimedeparturesV4.json?key=91dd5c34135f47d5afdfd08592ea2987&siteid=9192&timewindow=5>

=> en dictionary med alla avgångar inom 5 minuter

Skapa en SL-skylt med Trafiklab

Trafiklab.se innehåller massa API:er för trafik i Sverige och inkluderar SL reseplanerare-API. Du kan skapa ett eget konto där för att få egna nycklar eller använda nycklarna som finns på denna sida.

1. Fråga användaren om vilken station de vill ha realtidsinformation om
2. Hitta siteid för den stationen
3. Hämta realtidsinformationen för stationen med den siteid
4. Printa ut alla bussar/tunnelbanor som kommer inom x minuter

Klar med trafikskylten?

Då ska vi prata om något som heter objekt och klasser, eller objektorienterad programmering.

Python är ett objektorienterat språk, och det betyder att element vi jobbar med är objekt som har egenskaper/attribut och funktioner de kan utföra (metoder)

Ex: En sträng (str) har metoder som `str.upper()` och `str.lower()`

En lista har metoder som `list.append()`, `list.insert()` och `list.sort()`

Objekt och klasser

Ett objekt tillhör en klass. Ni kan ta reda på vilken klass ett objekt tillhör genom att kalla på funktionen `type()` med objektet som argument.

- `type("string") => <class 'str'>`
- `type(100) => <class 'int'>`

En klass har inbyggda funktioner (metoder). Ni kan ta reda på vilka metoder en klass har genom att skriva `dir(objektet)` eller `dir(klassen)`, t.ex.:

- `dir(str)`
- `dir(199)`

Klasser

Vad är klasser?

Alla objekt tillhör en klass. Kan tänka som i verkliga världen:

Min Volvo v70 tillhör klassen Bil, jag (Jocke) tillhör klassen Människa

En klass innehåller vissa egenskaper alla objekt som tillhör den klassen har:

T.ex. En bil har en egenskap som är registreringsnummer, tillverkningsår, etc. Bil har funktioner som är att svänga, bromsa, gasa osv.

Objektorienterad programmering går ut på att tänka att allt är objekt som i sin tur tillhör klasser. Man skriver då egna klasser som man sen använder i sin kod.

Varför?

- Tydlig uppdelning av kod
- Man kan lagra många egenskaper i ett objekt/klass istället för att skapa enskilda variabler
- Lätt att veta vad ett objekt kan göra för saker/har för egenskaper om man känner till dess klass

Vår första klass

```
class Land:
    def __init__(self, namn, befolkning, koordinater):
        self.namn = namn
        self.befolkning = befolkning
        self.koordinater = koordinater
```

```
land1 = Land("Sverige", 10000000) # Vi skapar ett objekt av klassen Land
print(land1.befolkning)
```

```
land2 = Land("Tyskland", 89000000) # Vi skapar ett objekt av klassen Land
print(land2.befolkning)
```

Vad är `__init__`? Det som utförs när vi skapar ett nytt objekt som tillhör klassen 'Land'

Vad är `self`? `self` är det specifika Landet som vi skapar. `self=land1` när vi skapar `land1` och `self=land2` när vi skapar `land2`.

Funktioner/metoder i klassen

```
class Land:
    def __init__(self, namn, befolkning):
        self.namn = namn
        self.befolkning = befolkning

    def print_info(self):
        print(self.namn, "är ett land med", self.befolkning, "invånare")

    def öka_befolkning(self):
        self.befolkning = self.befolkning + 1

land1 = Land("Sverige", 10000000)
land1.print_info()
land1.öka_befolkning()
land1.print_info()
```

En andra klass

```
class Player:
    def __init__(self, position):
        self.position = position
        self.points = 0

    def move(self, x, y):
        self.position[0] = self.position[0] + x
        self.position[1] = self.position[1] + y

    def point(self):
        Self.points = self.points + 1

player = Player([0,0]) # Skapa en ny spelare
print(player.position)
player.move(1, 0);
print(player.position)
```

Använda vår klass i praktiken

Nu ska vi göra ett grafiskt (2d) spel där vi använder vår klass “Player”

2d-Grafik i Python

- Finns en modul, **pygame**, som gör det enklare att skapa 2d-grafik i python för exempelvis spel
- Den kan installeras på samma sätt som vi installerade moduler tidigare

2d-Grafik i Python

Finns exempelkod på github-sidan som implementerar väldigt enkel grafik.

Vi utgår från det när vi gör vårt spel!

Steg att gå vidare

- **TKInter** - gör grafiska interfaces istället för bara textbaserade program
 - `import tkinter as tk`
 - IDLE är gjort med TKInter
- **Pygame**
 - Gör 2d-Spel
- Skaffa en mer kraftfull IDE än **IDLE**
 - Ex. **Pycharm**, **Spyder** eller **Visual Studio Code**
- Kolla upp roliga API:er som man kan göra praktiska/roliga saker med
- Leta upp praktiska moduler du kan installera med pip

TKInter

TKInter är standarden för grafiska program i python (Dvs program som inte bara är textbaserade). Vi kan skapa knappar, textrutor och fönster precis som vi är vana med.

Vi testar några grunder i TKInter

Uppgifter

1. Utveckla programmet med SL-skylden, så att man ser bussar också. Hämta gärna också mer information om varje avgång, kolla på denna länk vad som kan vara relevant: <https://www.trafiklab.se/node/15754/documentation>
2. Skapa ett program som räknar antalet ord och antalet bokstäver i en textfil och printar detta till användaren. Användaren ska kunna specificera vilken textfil som ska analyseras (som input).
Extra-uppgift: Räkna ut vilka de vanligaste orden är i textfilen (tips: dictionary { "ord1": 10, "ord2": 2 })
3. Skapa en lösenords-generator. När du kör programmet så ska användaren kunna säga hur långt lösenordet ska vara. Programmet ska sen printa ut ett lösenord som består av slumpvist valda bokstäver och siffror och specialtecken.
4. Skapa ett pygame-program där man ritar med en 'pensel'
5. Kom på ett eget projekt eller fråga mig! Exempel: Twitterbot, tre i rad spel. Tkinter. Moviepy.

Testa Pycharm

Ladda ned här: <https://www.jetbrains.com/pycharm/download/>

Öppna Pycharm och Create new Project (med lämpligt namn)

Inom ditt projekt kan du skapa filer (File -> New -> Python file)

För att köra en viss fil: Högerklicka på filen och tryck Run