Programmering i Python

30 januari	Grunder i Python
6 februari	Loopar, moduler
13 februari	Funktioner, läsa data
20 februari	Dictionaries, läsa data från internet +objekt
5 mars	Rep, uppgifter, egna projekt

Idag

- Repetition
- Genomgång uppgift från förra gången
- Klasser i olika filer
- Uppgifter eller eget “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"
}
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”]
```

Hämta data från internet

Först måste vi installera python-paketet requests:

1. Öppna CMD
2. `python -m pip install --user requests` (MacOS: python3 -m pip install requests)

Python-scriptet finns i förra veckans mapp

Klasser repetition

```
class Land:
    name = "Sverige"
    befolkning=10000000

sverige = Land() # Vi skapar ett objekt av klassen Land
print(sverige.befolkning)
```

Vad är klasser?

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

Min bil tillhör klassen Bil, jag tillhör klassen Människa

En klass innehåller vissa parametrar/egenskaper och metoder.

T.ex. En bil har en egenskap som är registreringsnummer, ålder, etc.

En människa har egenskaper som namn, ålder, längd etc.

Man kan också se klassen som en ritning för objekt, medan objektet är implementationen (det konkreta)

Klasser - repetition

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

    def increaseBefolkning(self, inc):
        Self.befolkning += inc
    def nationalAnthem(self):
        print(self.namn * 10)

sverige = Land("Sverige", 10000000)
norge = Land("Norge", 3000000)

sverige.nationalAnthem()
norge.increaseBefolkning(100)
```

__init__ är en method som automatiskt kallas på när vi skapar ett nytt objekt av den klassen.

← Skapar ett objekt med namn Sverige
← Skapar ett objekt med namn Norge

Analysera en csv fil

Vi har en csv-fil (comma separated values). Det är ett vanligt sätt att lagra data på, och kan visas i Excel på ett smidigt sätt.

CSV-filen innehåller alla städer som har haft corona-fall, och antalet corona-fall i varje stad och vilket land staden ligger i.

Vi vill analysera hur många corona-fall varje land har och låta användaren be om antal coronafall i ett specifikt land.

Vi vill sen skapa en lista med länder och antal corona fall. Den ska sorteras så att alla länder står i ordning efter flest antal corona-fall, och sedan printa ut denna listan.

1. Ladda ned csv-filen från github (tillfalle5/corona)
2. Vi skriver ett python-program som öppnar csv-filen
3. Vi gör saker med innehållet i csv-filen

Steg att gå vidare

- **TKInter** - gör grafiska interfaces istället för bara textbaserade program
  - import tkinter as tk
- **Pygame**
  - Gör 2d-Spel
- Skaffa en mer kraftfull IDE bättre än **IDLE**
  - Ex. **Pycharm**
- Skriv dina Python-program i en textredigerare och kör programmen i **Terminalen** (**CMD** eller **PowerShell** på Windows)
- Kolla upp roliga API:er som man kan ha kul med, eller göra praktiska saker med
- Kolla upp praktiska paket, tex för att skapa csv-filer eller liknande

Uppgifter

1. Skapa en dictionary som innehåller några engelska ord som nyckel och deras svenska översättningar som värden, skapa sedan en funktion, translate() som tar in ett engelskt ord och returnerar det svenska ordet. Om översättningen inte finns så returnera *“Sorry, word not found”*

2. Utveckla programmet som beskrev vädret i en specifik stad till att skriva ut mer än bara en beskrivning av vädret (t.ex. temperatur, solens upp- och nedgång) om du inte redan gjort det

3. Skapa ett program som analyserar hur många ord och bokstäver en textfil har. Användaren ska kunna specificera vilken textfil som ska analyseras.
   Extra-uppgift: Räkna ut hur många gånger varje ord förekommer (tips: dictionary { “ord1”: 10, “ord2”: 2 })

4. 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 är helt slumpmässigt och består av bokstäver och siffror.

5. Skapa ett program som ber användaren om en hållplats inom SL och printar ut realtidsinformation.
   Använd trafiklab.se:s API:er SL Platsuppslag och SL Realtidsinformation 4.
   > Se nästa sida i slidesen för information om hur man kan göra detta.

6. Kom på ett eget projekt eller fråga mig! Exempel: Twitterbot, tre i rad spel.

Skapa en SL-skylt med Trafiklab

Trafiklab 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.

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
Nyckeln är: 3e2085604f874b2aaf2eb73c6f0e256c, sökord får du skriva in själv
Exempel:
https://api.sl.se/api2/typeahead.json?key=3e2085604f874b2aaf2eb73c6f0e256c&searchstring=slussen&maxresults=1
=> siteid: 9192

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=<ID>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 innehåller massa API:er för trafik i Sverige och inkluderar SL reseplanerare-API

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 tunnelbanor och bussar som kommer inom 10 minuter

Säg till om ni behöver hjälp!