


Programmering i Python

F1	Grunder i Python
F2	Loopar, moduler
F3	Funktioner, läsa/skriva filer
 F4	Dictionaries, läsa data från internet
F5	...

Idag

- Repetition
- Genomgång uppgift från förra gången
- Dictionaries
- Hämta data från internet

Repetition - Funktioner

Definition funktion i Python:

- Tar in ett eller flera argument (värden)
- Gör något med dessa
- Kan ge tillbaka (returnera) något
- När man använder en funktion så skriver man `funktionsnamn(argument)`

Exempel på inbyggda funktioner i Python:

`sum([1, 2, 3, 4])` : en funktion som tar in en lista med siffror och returnerar summan av dessa

`print("Hej!")` : en funktion som tar in en sträng och skriver ut den till användaren

Gör din egen funktion sum i Python:

```
def my_sum(list):  
    result = 0  
    for number in list:  
        result += number  
    return result
```

Repetition - filer i Python

Öppna filer och printa alla ord

```
f = open("fil.txt", "r")
for line in f.readlines():
    for word in line.split():
        print(word)
f.close()
```

Skapa/Skriv över en fil och skriv innehåll till den

```
f = open("nyttfilnamn.txt", "w")
f.write("Detta skrivs i en fil, och skriver över det
som stod innan")
f.close()
```

Öppna/Skapa en fil och lägg till text i slutet av den

```
f = open("nyttfilnamn.txt", "a")
f.write("Detta lägger till denna text i en fil")
f.close()
```

Genomgång uppgift 2

Gör ett program som ber användaren om namn, efternamn och ålder och skriver det till en ny rad i en fil, filen ska kunna fyllas på och inte nollställas

```
f = open("namnlista.txt", "a")
f_name = input("Förnamn: ")
l_name = input("Efternamn: ")
age = input("Ålder: ")

f.write(f_name + " " + l_name + ": " + age + "\n")
f.close()
```

Genomgång uppgift 6 - rövarspråket

Gör en funktion som tar in ett namn (sträng) och returnerar strängen i rövarspråket (hej => hohejoj, erik => erorikok) och testa den med några strängar

Dictionaries

Vi har lärt om oss listor för att lagra mer än ett värde i.

```
Lista = [1, 2, 3, 4, 5]
```

Värdena lagras efter varandra och har ett varsitt index Lista[0], Lista[1], Lista[2] osv.

Dictionaries - skapa en dictionary

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 error. 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

Vi kanske vill hämta superaktuell data från en hemsida för att vårt pythonprogram ska kunna visa exempelvis dagens väder, priset på aktier osv. Att hämta en hemsida kallas för en request.

Först måste vi installera python-paketet requests:

1. Öppna CMD / Powershell

2. `py -m pip install --user requests`

(MacOS: `python3 -m pip install --user requests`)

Jag har en URL som säger vad vädret är i Stockholm just nu, vi gör ett program som hämtar det och printar ut något om det till användaren.

URL:

<https://api.openweathermap.org/data/2.5/weather?q=Stockholm&appid=831c45d6783bf7371e3237d852d5daf8>

Gå in på hemsidan + idé

<https://api.openweathermap.org/data/2.5/weather?q=Stockholm&appid=831c45d6783bf7371e3237d852d5daf8>

Det vi ser på den hemsidan är massa text och curly brackets. Ser ut som en dictionary.

Vi vill skapa ett program som hämtar den textsträngen och omvandlar den till en dictionary så att vi kan använda den på ett enkelt sätt i Python.

Skapa programmet

1. Skapa en fil
2. Importera paketet requests i vårt program: `import requests`
3. Testa att göra en request till vår URL med `request.get(<URL>)` (<URL> = vår länk som en sträng)
4. Spara returvärdet som en variabel (`response`) och printa `response.text`, för att se om vi lyckades hämta hemsidan
5. För att omvandla från text (i JSON-format) till en dictionary skriver vi `response.json()`
6. Sen hittar vi rätt keys för att printa ut värdet (se demo)

Uppgifter

Uppgift 1: Gör så att användaren kan be om väder i en annan stad än Stockholm (med input)

Uppgift 2: Ge även information om Temperatur i staden (temperaturen är skriven i Kelvin)

Uppgift 3: Hantera error om användaren skriver in något som inte är en stad

Tips1: Kolla vad som ligger i `weather_dict` när användaren anger något som ej är en stad

Tips2: If-sats?

Uppgift 4: Skriv solens upp och nedgång i staden. Solens upp och nedgång är skrivet i ett format som heter UNIX time, googla på hur man kan översätta det till vanlig tid. "Translate UNIX time to regular time python"

Extra-uppgift: Gör ett nytt program där du skickar en request till:

<https://bostad.stockholm.se/Lista/AllaAnnonser> (Innehåller en lista med lediga lägenheter)

Och loopar över alla lägenheter för att räkna antalet ungdomslägenheter

Tips: Kolla på hemsidan, hur vet man om en lägenhet är en ungdomslägenhet?

Python är objektorienterat, vilket ni inte behövt tänka på än så länge.

Det betyder att nästan alla värden i python är så kallade objekt.

Vad är ett objekt i Python?

- Ett objekt har egenskaper, ex: `object.name`, `object.size`
- Ett objekt har metoder (funktioner bundna till ett objekt), ex: `object.capitalize()`

Ett objekt tillhör en klass. Ni kan ta reda på vilken klass ett objekt tillhör genom att kalla på egenskapen

`__class__` hos ett objekt:

- `"sträng".__class__ => type = str`
- `(100).__class__ => type = int`

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

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

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 fysiska)

Objektorienterad programmering går ut på att tänka och formulera alla värden i sin kod som objekt som i sin tur tillhör klasser. Kan verka lite abstrakt till en början.

Varför skulle man vilja skriva objektorienterade program?

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

1. Skapa en egen klass (Bil) som har registreringsnummer, tillverkare och hastighet. Skapa en metod i klassen som heter öka_hastighet och en metod som heter minska_hastighet som ändrar på bilens hastighet. öka_hastighet ökar hastigheten med 1 och minska_hastighet minskar hastigheten med 1 (utan att göra den negativ). Printa även hastigheten i de två metoderna. Skapa sedan ett objekt som är en Bil och som du ökar hastigheten på och sedan minskar den.
2. Gå in på <https://www.practicepython.org/> och gör uppgifter om du blir klar

Fråga gärna om hjälp!

