


Programmering i Python

27 oktober	Grunder i Python
10 november	Slingor, moduler
17 november	Funktioner, läsa data
 1 december	Läsa data forts., dict + objekt + klasser
8 december	? - Ge gärna förslag

Idag

- Repetition
- Genomgång uppgift från förra gången
- Dictionaries
- Objekt
- Klasser

Repetition - Funktioner

Definition funktion i Python:

- Tar in ett eller flera argument (värden)
- Gör något med dessa
- Kan ge tillbaka (returnera) något
- När man kallar på en funktion 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()
```

Frågor från tidigare

from random import randint?

Enklare sätt att skriva på än `import random` och
`random.randint`

Låta användaren bestämma när programmet ska avslutas? Bra fråga, kan återkomma.

Kontroll av indata? Är det verkligen en int som användaren gav t.ex.?

```
while not indata.isdigit():  
    Indata = input("Inte en siffra.. ")
```

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 3

Gör en funktion som tar in en sträng och returnerar strängen baklänges. Du får inte använda metoden `reverse()`.

```
def my_reverse(meddelande):  
    resultat = "" # Tom sträng  
    antal_bokstaver = len(meddelande)  
  
    for index in range( antal_bokstaver ):  
        # Gå igenom varje index för varje bokstav i meddelande  
        resultat += meddelande[antal_bokstaver - 1 - index]  
  
    return resultat  
  
print(my_reverse("hejsan"))
```

Genomgång uppgift 4

Skapa en funktion som tar in en lista med siffror och returnerar den största siffran

```
def my_max(lista):  
    max = -float("inf") # det minsta talet som finns  
    for heltal in lista:  
        if heltal > max:  
            max = heltal  
    return max  
  
print(my_max([100000, 2, 3000000, 4, 5, -1000]))
```


Dictionaries - skapa en dictionary

I en lista lagrar vi värden på en viss position (index). För att nå värdet på index 3 i lista så skriver vi lista[3]. En lista kopplar med andra ord ett index (heltal) till ett värde.

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)
```

Hämta data från internet (om ni vill)

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

Objekt

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)`

Klasser

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)

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

Vår första klass

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

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

Men detta är ju i princip en dictionary fast med annan syntax? Varför skulle vi använda en klass?

Vår första metod i klassen

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

    def printName(self):
        print(self.name)

sverige = Land()
sverige.printName()
```

Vad är self? Self är det objektet som kallar på metoden. Det blir kanske tydligare snart, när vi gör om klassen så att den funkar för vilket land som helst och inte bara Sverige.

Vår första riktiga klass

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

    def printBefolkning(self):
        print(self.befolkning)
    def printNamn(self):
        print(self.namn)
```

```
sverige = Land("Sverige", 10000000)
```

```
norge = Land("Norge", 3000000)
```

```
sverige.printNamn()
```

```
norge.printBefolkning()
```

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

←

←

Uppgifter

1. Skapa en dictionary som innehåller några engelska ord som nyckel och deras svenska översättningar som värden, skapa sedan en funktion som tar in ett engelskt ord och returnerar det svenska ordet. Om översättningen inte finns så returnera *“Sorry, not found”*
2. Ändra programmet där vi fick väderinformation så att användaren får välja vilken stad den får väderinformation om
3. Skapa en klass Bil som innehåller parametrarna reg_plate, owner, color, year_built och som innehåller metoden __init__ och metoden print_car_info som printar ut all information om bilen på ett snyggt sätt
 - a. Skapa ett objekt av klassen Bil och kalla på metoden print_car_info()
4. Skapa en funktion som översätter en sträng till strängen i rövarspråket (<https://sv.wikipedia.org/wiki/R%C3%B6varspr%C3%A5ket>)