Grundläggande Python

# Installation och test run

if chapter==done  
 print(‘Done.’)  
*run*  
Done.

# Introduktion och ett första programexempel

## Hur en dator fungerar

En lärdom att ta med sig är att alla program (inkl. OS) kopieras i någon form över till primärminnet (RAM) för att kunna användas effektivt och sparas sedan till sekundärminnet (SSD). Datorn i sig kan bara hantera binära tal. Ett binärt tal är uppbyggt i bitar (från engelskans binary digit), antingen 0 eller 1. De har varsin spänning:

0 = 0V  
1 = 3,3V

8 bitar = 1 byte. En byte kan alltså representera talen 0-255 (därav den ökända talföljden 256, 512 osv).

A picture containing calendar

Description automatically generated

Ett datorminne består i huvudsak av ett adressfält och ett datafält. Datorn får tillgång till datafältet via adressbussen. Adress- och databussen knyter samman minnena med processorn och övriga komponenter. Adress- och datafältet representeras av ett binärt tal som tolkas i en teckentabell enligt en internationell standard.

Ett datorprogram förklarar för datorn hur den ska utföra en uppgift. Programmet innehåller en serie programsatser (instruktioner) och dessa programsatser bildar en källkod. Du kan skapa källkod i vilken ordbehandlare som helst men vanligast är i en integrerad utvecklingsmiljö (IDE – integrated development environment).

Python är ett högnivåspråk. Det översätts till maskinkod (binära tal) genom antingen en kompilator (översätter hela källkoden och skapar en exekverbar fil) eller en interpretator (översätter instruktion för instruktion löpande). Vid språkfel anges syntax error. Python använder en interpretator. Det är ett indenterande (Andrew: en ”tab” är fyra mellanslag) språk som bidrar till en mera strukturerad och lättläst källkod.

## Hur du skapar ett datorprogram

1. Specificera tydligt uppgiften som datorprogrammet ska lösa.
2. Identifiera vilken information som datorprogrammet ska hantera (input och output).
3. Rita upp ett flödesschema över hur uppgiften ska lösas.
4. Översätt flödesschemat till programkod.
5. Testa programkoden (exekvera programmet).
6. Åtgärda eventuella fel – syntaxfel och logiska fel (debugga programkoden)
7. När programkoden fungerar korrekt är det denna programkod som utgör det slutgiltiga datorprogrammet.

## Läsanvisningar

Läs avsnitt 1.1, 1.2 och 1.6.

Maskinkod är instruktioner i ett datorprogram som skrivs ut i bitar, dvs binära tal.

Ett programspråk är vad som översätter instruktioner till maskinkod. T ex assembleringsspråk eller högnivåspråk (Python).

Ett kompilerat programspråk (C++) översätter instruktioner till maskinkod i förväg. Ett visst programspråk måste alltså ha rätt motsvarande kompilator som översätter till maskinkod, annars går det inte. En kompilator säkerställer först att programspråket följer sina skrivregler (syntax).   
+ snabbt och effektivt  
+ fel upptäcks tidigt  
- kompilering är ett extra steg  
- kompilatorn är låst för just det programspråk det programmerats för.

Ett interpreterande språk (Python) tolkar instruktioner. Utifrån sett ser det ut som att programmet exekveras direkt i interpretatorn. Detta är vanligt i client-server-tillämpningar där klienten använder en webbäsare och där tillämpningen helt/delvis körs på en avlägsen server. Här finns programdelar/småprogram som kallas skript.   
+ slipper kompilera  
+ flyttbara program mellan olika datorer  
- långsammare  
- inte syntaxkontrollerade vilket ökar risken för att det kan bli fel

När interpretatorn används att den kör ett program som en fil kallas det script mode. Motsatsen är att köra en instruktion i taget, interactive mode.

# Datatyper, variabler, operatorer, in- och utmatning

Datatyper är ett centralt begrepp i Python.

Jämförelse-/relationsoperatorer: <, <=, >, >=, ==, !=  
Logiska operatorer: and, or, not

## Läsanvisningar

Kapitel 2 samt 3.3.

Testa att läsa först och sedan gå igenom materialet på webben. Sedan dubbelkolla lärokraven på sammanfattningen i slutet av kapitel 2 för att se så allt fastnat.

## Övningar

Från boken, 2.1-2, 2.4-6.

# Strängar

str()

Konstanta värden kallas literaler. Vad det gäller text kallas de textliteraler (wow!).

Vi använder backslash ”\” för att skriva vilket tecken som helst i en sträng. Kombinationer som börjar med \ kallas escape-sekvenser.

Triple-quoted strings är super! Du anger en string i början och slutet av tre apostrofer, allt däremellan kommer att generera exakt samma utmatning som inmatning. Jag behöver alltså inte ange några escape-sekvenser. Dvs:

|  |
| --- |
| Ex1 = ’’’Min ”mamma” kallas ’Pia’  men jag kallar henne: Mega-Mom’’’ print(Ex1) |

Min ”mamma” kallas ’Pia’  
men jag kallar henne: Mega-Mom

Varje specialtecken har en teckenkod. Tecknet lagras alltså i ett heltal i programmet. T ill exempel har tecknet a teckenkod 97. Nu för tiden finns Unicode som kan lagra över en miljon tecken. Den expanderar på gamla tekniker där de allra vanligaste tecknen har lägre nummer och de ovanligare har högre. Då högre tecken har en högre teckenkod behöver flera bitar/bytes användas vilket gör att det tar mer minne. UTF-8 är en teknik som sparar de vanligare tecknen i en byte medan ovanligare kan lagras i två, tre, fyra bytes.

En indexkonstruktion är att indexering av en sträng ges ett indexnummer. Nedan har bokstaven P ett indexnummer på 4 alternativt -7 beroende på om vi räknar från första tecknet (kallas 0) eller sista tecknet (kallas -1).

Table

Description automatically generated

Om jag indexerar väljer jag ut ett eller flera värden i en sekvens. En sekvens ser ut såhär:

|  |
| --- |
| # s = [forst:sist:steg]  s = bajsmacka  s[3:8] |

smack

Jämförelseoperatorer fungerar även på sekvenser (strängar). Notera att det är teckenkoder som jämförs, alltså inte

# Listor

asd

# Att välja (if-konstruktioner)

asd

# Att upprepa (while- och for-konstruktioner)

asd

# Egendefinierade funktioner

asd

# Moduler

asd

# Filhantering

asd

# Plottning

asd

# Objektorienterad programmering

asd

# Blandade övningsexempel

asd

# Lösningsförslag

asd

# Kompletterande material

asd

# Inlämningsuppgift

## Datum

2022-12-07: Inlämning uppgift (4 av 6 deluppgifter måste vara *påbörjade*)  
2022-12-12: Inlämning kamratrespons (följ instruktion)  
2022-12-19: Inlämning lärarrättning 1 (ger kommentarer)  
2023-01-05: Inlämning lärarrättning 2 (betygsätter)

## Uppgift

Undersök allmän prisutveckling 1980–2022.   
Undersök prisutveckling för varor och tjänster 1980–2022.

Klipp och klistra inte kod, anropa i stället kod som redan skrivits tidigare.   
Stäm av koden mot regler och riktlinjer under *Förutsättningar*.   
Som sista steg: rensa variabler (*Restart & Clear Output*) och testkör programmen en sista gång för säkerhets skull.

### Deluppgift 1

asd

### Deluppgift 2

asd

### Deluppgift 3

asd

### Deluppgift 4

asd

### Deluppgift 5

asd

### Deluppgift 6

asd

# Tentamen

asd