Oscar Antbring

PROGRAMMERING 1

Våra böcker och tillhörande produkter är noggrant kontrollerade, men det är ändå möjligt att fel kan förekomma. Vi tar gärna emot förbättringsförslag.

Tryckt av Elanders
Utgiven 2024

ISBN: 978-91-7531-178-4
Artikelnummer: 3120

Författare: Oscar Antbring

Omslag: Docendo AB
Bild på omslaget: © Adobe Stock

Innehållsförteckning

2 Programutveckling i Python

I detta kapitel ska du få läsa om **Python** som är ett av världens mest kända och populära programmeringsspråk. Det är ett mångsidigt språk och används inom flera områden såsom för att bygga mobila applikationer, hantera databaser eller skriva skrivsbordsprogram till vanliga operativsystem som MacOS och Microsoft Windows.

Python är också kanske det allra mest populära programmeringsspråket idag för att utveckla AI (artificiell intelligens). Några andra vanliga ändamål är att det används för forskning, analys av data samt utveckling av webbapplikationer.

En viktig anledning till att Python är populärt inom så vitt skilda områden är för dess omfattande utbud av *bibliotek*. Ett bibliokek är en uppsättning färdiga koder som är skriven för ett särskilt ändamål och som finns fritt tillgängliga för alla utvecklare att använda.

I detta kapitel så kommer du att få läsa om flera programmeringsbegrepp och idéer som du bör känna till. I detta skede så handlar det om att göra dig som läsare medveten om dessa och vad de innebär men den verkliga förståelsen kommer nog inte infinna sig ännu.

Avslutningsvis så kommer du att kunna läsa om hur du installerar Python och en utvecklingsmiljö i syfte att skriva koder och utveckla program.

Vad utmärker Python?

Om du skulle läsa på nätet eller i böcker om vilka för- och nackdelar som Python medför som utvecklingsspråk så omnämns ofta många. I detta avsnitt ska du få läsa om utvalda sådana och som vi tycker är särskilt intressanta och/eller viktiga.

Fördelar med Python

Följande fördelar omnämns ofta i samband med att man läser om Python som programmeringsspråk:

Enkel syntax

En fördel som uppskattas av många är Pythons syntax som är väldigt enkel och förhållandevis lättläst.

Kodsyntax är de regler som är kopplade till ett programmeringsspråk och som styr vilka tecken du behöver skriva och med vilken struktur för att det ska kunna åstadkomma något specifikt.

I kodexemplet (1) kan du se kod som skapar en samling (lista) i Python med flera tal och att dessa sedan skrivs ut.

```
1.  #Exempel 1: Skapa en lista med tre frukter i Python
2.
3.  my_fruits = []                          #Skapa en lista
4.  my_fruits.append(”äpplen”)              #Lägg till en ”frukt” i listan
5.  my_fruits.append(”apelsiner”)           #Lägg till en ”frukt” i listan
6.  my_fruits.append(”bananer”)             #Lägg till en ”frukt” i listan
7.
8.  print(my_fruits)                        #Skriver ut ”frukterna”
```

Motsvarande kod i Java ser ut som föjer:

```
1.  #Exempel 2: Skapa en lista med tre frukter i Java
2.
3.  import java.util.ArrayList                      //Lägger till en klass för samlingar
4.
5.  public class Program{                           //Man måste inte lägga all kod i klasser i Python
6.      public static void main(String[] args){    //Man måste inte lägga all kod i metoder i Python
7.          ArrayList myFruits = new ArrayList(); //Skapa en lista
8.          myFruits.add(”äpplen”);                //Lägg till en ”frukt” i listan
9.          myFruits.add(”apelsiner”);             //Lägg till en ”frukt” i listan
10.         myFruits.add(”bananer”);               //Lägg till en ”frukt” i listan
11.
12.         System.out.println(myFruits);          //Skriver ut ”frukterna”
13.     }
14. }
```

Nu visar dock exemplet en ogynnsam situation för Java och skillnaden är vanligtvis inte så här påtaglig. En viktig anledning till att koden i Python kan skrivas så mycket enklare i detta fall är för att Python tillåter att du skriver kod rakt upp och ner, utan att koden behöver ”paketeras in” i så kallade *klasser* och *metoder* (som vi snart kommer skriva mer om)

I andra kända programmeringsspråk som Java och C# så måste all kod paketeras in på detta sätt.

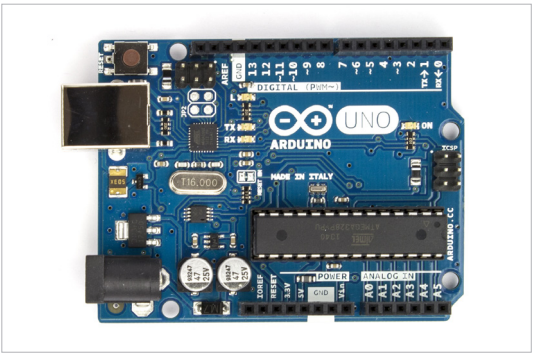
Det är vanligt att nybörjare i programmering väljer Python som språk just för att koden i sig upplevs enklare och mindre ”skrämmande”. Och det ska inte underskattas så klart.

Men Pythons enkla syntax är inte bara fördelaktigt för nybörjare som vill lära sig programmering. Även erfarna progammerare uppskattar den enkla syntax och inom vissa områden såsom i att programmera smart utrustning i våra hem (IoT) så kommer särskilt den enkla och rena syntax till sin rätt.

Plattformsoberoende

En av de viktigaste fördelarna med Python är att det är *plattformsoberoende*.

För att förstå vad det innebär så behöver vi utreda begreppet *plattform*. En sådan handlar i princip om vilken datormiljö du syftar till och då en specifik *hårdvara* och ett *operativsystem*.



När det handlar om hårdvara så grupperar man dessa utifrån vilken processor som finns i datorn och den vanligaste sådan är **x86** (som i tur inkluderar flera olika varianter). Eller som du troligtvis känner igen plattformen som: **PC** (även om kopplingen mellan begreppet PC och x86 inte är helt okomlicerad)**.**

Andra exempel på hårdvaruplattformar är ARM (som används i exempelvis chromebook, mobiltelefoner och Arduino) och Playstation.

I bilden kan du en Arduino som är en så kallad enkorts-dator och som bland annat består av en processor med ARM-arkitekturen.

> En plattform är den datormiljö som användaren sitter vid och det syftar till hårdvara (det du kan ”ta på”) samt operativsystemet som är det program som styr hårdvaran.

Även en dator med samma hårdvara (som en PC) kan innebära olika plattformar och det beror på att en dator kan köra olika operativsystem. Och ett operativstem är den viktigaste mjukvaran på en dator och som hanterar datorns hårdvara samt tillåter en miljö för andra program att köra i. Det betyder att program som webbläsare, ordbehandlingsprogram och datorspel behöver ett operativsystem för att kunna köra.

Exempel på operativsystem som finns till PC-kompatibla arkitekturer är OS/2 (”mac”), Linux och Microsoft Windows.

Alla dessa kombinationer av hårdvara och mjukvara bildar en möjlig plattform. Och att Python är plattformsoberoende innebär alltså att den kod som du skriver kan köras oberoende av vilken typ av dator eller annan utrustning som programmet ämnar användas för.

Vi kommer att återkomma lite mer till detta när du kommer läsa om hur kod som du skriver i Python omvandlas till ett körbart program.

Python används för IoT

Python används ofta för att styra mindre enheter som är uppkopplade mot nätet och som går under samlingsbegreppet **Internet Of Things**. Enheter som kan betecknas ingå i detta begrepp är exempelvis sensorer, lampor och kameror.

För ändamålet är Python mångsidigt och har ett stort utbud av färdiga relevanta bibliotek och moduler. Även programmeringsspråkets enkla och läsbara syntax är en aspekt som ofta omnämns när man läser om varför de flesta som utvecklar mjukvara för denna typ av utrustning väljer att använda Python.

Python har ett enormt ”community”

”*Community*” är inte bara en väldigt rolig tv-serie utan också begreppet för en grupp människor med gemensamma intressen och som verkar för att tillsammans utveckla det som de är intressadera av. I detta fall programmeringsspråket Python.

Och de personer som är en del av denna gemenskap har en betydande roll i hur Python utvecklar sig som programmeringsspråk. Detta skapar ett genuint engagemang där pengar inte nödvändigtvis den högsta motivationsfaktorn.

Det handlar om att utveckla ramverk, bibliotek och andra resurser som bidrar till ökad(e) produktivitet och användningsområden.

Python är ett objekt-orienterat programmeringsspråk...

De flesta moderna programmeringsspråk bygger på en tankeidé hur moderna program ska konstrueras - ett *paradigm* som kallas *objektorientering*.

> Ett *programmeringsparadigm* är ett synsätt på hur program ska konstrueras och fungera. Ett objektorienterat paradigm används i språk som **Java**, **C#** och **Python**.

Vi kommer snart beskriva begreppet lite mer ingående, mest för att du ska kunna förstå andra koncept kopplat till Python som programmeringsspråk och PyCharm som utvecklingsmiljö.

Och i det sista kapitlet i denna lärobok så kommer du att uttryckligen skriva kod enligt ett objektorienterat synsätt.

... men också ett scriptspråk..

Det är dock viktigt att påpeka att Python inte är ett renodlat objektorienterat språk, utan att det också är ett *scriptspråk* som kännetecknas av mindre koder som tolkas rad för rad. Sedan är det också delvis fel att sätta ett motsatsförhållande mellan scriptspråk och objektorienterad programmering.

3 Hello World

Du ska nu få skriva dina allra första körbara kod. Sedan kommer du bara att utveckla dina kunskaper genom kommande kapitel och dina program kommer bli alltmer omfattande och användbara. Och roliga att skriva koden för.

Inledningsvis så kommer du säkerligen bli varse hur petigt det är med programmering. Ett felaktigt tecken gör koden okörbar. Lyckligtvis så jobbar du i en utvecklingsmiljö som **PyCharm** och som hjälper dig att skriva en korrekt kod. Detta skrev vi om i föregående kapitel.

Din första kod

Du ska skriva ett program som skriver ut en klassisk text ”*Hello World*” och som brukar utgöra det allra första programmet som en programmerare skriver. Om du skulle söka på uttrycket så skulle kunna inse att det har gedigna historiska anor och omnämns i väldigt gamlga programmeringsböcker. Enligt vissa källor så kan man hitta dess ursprung så långt bakåt som till slutet av 60-talet och språket **BCPL** (*Basic Combined Programming Language*).

För att skriva ut texten så behövs oavsett enbart följande kod:

```
1.  #Exempel 1: Utskrift av ”Hello world” som script
2.
3.  print(”Hello world”)
```

Och det är det som krävs. Du använder en funktion som heter **print()** och innanför parentesen skriver du texten som ska skrivas ut, funktionens *argument*.

Hello world med färdig kod

Vi ska dock utgå ifrån att du valt att skapa ett projekt med alternativet för *välkomst script*. Och då kommer det innebära mer kod.

I kodexempel (2) så har vi skrivit en programsats som skriver ut texten ”*Hello World*” på skärmen.

```
1.  #Exempel 2: ”Hello world” med välkomstscript
2.
3.  def print_text(word):
4.          #Kodblock som tillhör funktionen main() börjar : Du skriver all din kod här.
5.          print(word)
6.
7.  if __name__ == ”__main__”:    #Om koden (modulen) körs som ett script (toppnivåinstruktion)
8.          print_text(”Hello world”)  #... så körs metoden main()
```

Detta är den kod som alltså visas om du valt att markera alternativet för att skapa ett script i samband med att du väljer att skapa ett nytt projekt.

I **bilagan** kan du läsa en mer ingående beskrivning av detta välkomstscript och där vi beskriver syftet med de olika kodraderna. Vi har valt att lägga det som en bilaga då du som läsare själv kan bedöma när det känns rimligt att ta sig an innehållet.

Mer om funktioner

Nedanstående avsnitt inkluderar kodning som vi går genom från och med nästa kapitel. Syftet är därmed inte i första hand att du ska förstå koderna utan att du ska få en inledande idé om hur du arbetar med funktioner samt hur den kontextuella hjälpen som finns inbyggd i utvecklingsmiljön kan underlätta arbetet med koden.

Prova gärna att skriva koderna och se om du kan få funktionerna att fungera samt att du kan se den inbyggda hjälpen i utvecklingsmiljön som visas såsom du kan se i bilderna.

I Python jobbar du inledningsvis med *funktioner* och dessa bygger på precis samma princip och har samma syfte som en *metod* men med skillnaden att en funktion är fristående. Den tillhör ingen klass. Såsom funktionen **print_text()** i kodexemplet tidigare.

Dessa skapas (ofta) direkt på toppnivån i koden, alltså utan någon indentering (indrag från vänster).

> Tänk tills vidare att funktioner och metoder är exakt samma sak. Det är bara att de är placerade lite annorlunda i koden. Där metoder tillhör klasser medan funktioner ligger på den högsta nivån i koden utan indentering.

4 Variabler

Med all respekt för programmet som kan skriva ut ”*Hello World*” men det händer inte så där väldigt mycket. För att vi ska kunna ta nästa steg och involvera användaren i programmet så behöver vi lära oss att använda variabler.

En fundamental komponent i alla programmeringsspråk är *variabler* som är ett sätt för oss att hantera värden. Dessa värden kan vara av olika typer som text, heltal och decimaltal. Vi kan ge en variabel ett värde direkt i koden, men ännu mer spännande blir det när vi ber användaren att mata in värdet. Då öppnas äntligen dörrarna för att vi ska kunna skapa *interaktiva* program. Alltså att användaren får en aktiv roll i programmet, att det sker ett samspel mellan användaren och datorn.

I detta kapitel så kommer du att få lära dig att hantera variabler och olika typer av värden. Detta kommer du behöva tampas med ett tag. Sedan kommer du också att lära dig hantera inmatningar från användaren, som då alltså är en förutsättning för att skapa en interaktion.

Även om du skrev din första kod i förra kapitlet så är det i och med detta kapitel som du kan börja skriva koder som skapar någon form av användbara program. Att programmet kan interagera med användaren är inte bara trevligt, det är ju helt avgörande för de flesta typer av program.

Det finns många kodexempel i detta kapitel. Jobba med dessa och hitta gärna på egna övningar. Du kommer säkert vilja läsa detta kapitel ett par gånger, men kom ihåg att det är först när du sätter dig ner för att skriva kod som den verkliga utvecklingen sker. Du kan inte läsa dig till att bli duktig i programmering, det måste praktiseras också.

Att skapa en variabel

En *variabel* är oavsett en plats i ett program där vi kan lagra ett värde, och för att åstadkomma detta så behöver du tänka på tre moment.

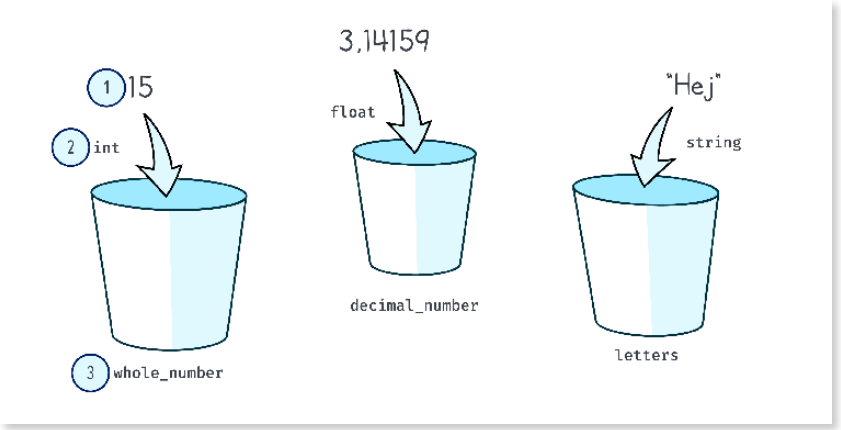
Nedanstående beskrivning utgår ifrån scenariot att du skapar tre olika variabler i koden och som alla kan lagra ett specifikt typ av värde.

```
1.  #Exempel 1: Tre variabler med olika datatyper
2.
3.  whole_number: int = 15                    #type hint (int)
4.  decimal_number : float = 3.14159          #type hint (float)
5.  letters : str = ”hej”                     #type hint (str)
```

I kodexemplet (1) kan du se koden som skapar de tre variabler som vi berättar om nu:

Namnge variabeln

Du behöver bestämma namnet på variabeln (1) och detta kommer du att märka är inte så lätt som det låter. I alla fall när det kommer till att välja *bra* namn för variabeler. Tills vidare så ska du dock inte ligga vaken om nätterna på grund av detta.



Vi kommer att återkomma till namngivning av variabler titt som tätt genom boken. I bilden så har vi exempelvis gett en variabel namnet ”*whole_numer*” och för detta exempel får vi lov att säga att det är ett tydligt och lämpligt namn. Men i kommande program så är det troligtvis inga vidare bra namn, eftersom syftet med variabeln troligtvis är något annat än att demonstrera att det lagrar just ett heltal.

Fastställa en datatyp

I Python kan du inte fastställa en datatyp själv, i alla fall inte när variabeln skapas. Däremot kan du i ett senare skede tvinga koden att tolka en variabel som om den innehar en specifik datatyp (mer om det snart).

Alla variabler *måste* tilldelas en datatyp (2) men till skillnad från många andra programmeringsspråk där du som utvecklare styr detta så sker alltså valet i datatyp av Python.

PROGRAMMERING 1 MED PYTHON

Boken är skriven för dig som vill lära dig programmering med Python från grunden. Programmering är utvecklande på flera sätt, du kommer att utveckla din problemlösningsförmåga och din logiska tankeförmåga. Likaså kommer du att få träna upp ditt tålamod och din förmåga att uppmärksamma detaljer och vår utgångspunkt när vi skrev boken är att programmering är lika mycket ett kreativt område som ett tekniskt.

I denna lärobok så får du lära dig det som krävs för att du ska få en stabil grund att stå på inför ytterligare studier och med mer avancerad programmering. Du kommer att få lära dig om variabler och vektorer som är två olika sätt att lagra värden på. Vidare kommer du att få lära dig att använda kontrollstrukturer som används för att styra program i olika riktningar. Som vi ofta vill ska styras av användaren till programmering. Vidare så kommer du att få lära dig att använda funktioner som används för att gruppera koder som fyller ett särskilt syfte och att vi sedan kan använda den uppsättning kod när vi behöver den. På samma sätt som att vi har olika verktyg i en verktygslåda och som alla har en specifik uppgift, en funktionalitet.

Du kommer få läsa om vad som anses vara en bra kod. Hur du kommenterar och dokumenterar kod, hur du strukturerar koden och hur du ska tänka för att eftersträva enkel kod i förmån för onödigt svår kod. Likaså kommer du att få lära dig om vad som utmärker Python som programmeringsspråk och vad som krävs för att du ska kunna skriva dina allra första program. Boken är grundad i ett paradigm eller tankesätt som kallas för objektorienterad programmering. Det innebär att du lär dig att skriva klasser som representerar saker eller så kallade entiteter i vår omvärld, och att du sedan kan skapa förekomster av dessa som kallas för objekt.

Boken är skriven utifrån gymnasieskolans läroplan och de riktlinjer som ligger till grund för kursen Programmering 1.

ISBN 978-91-7531-178-4

9 789175 311784 >

DOCENDO