

# TDP003 Projekt: Egna datormiljön

## Installationsmanual för Portföljsystem

Jimmie Roos, [jimro697@student.liu.se](mailto:jimro697@student.liu.se)  
Sebastian Grunditz, [sebgr273@student.liu.se](mailto:sebgr273@student.liu.se)

## Innehåll

<b>1</b>	<b>Revisionshistorik</b>	<b>2</b>
<b>2</b>	<b>Grundläggande information för installation</b>	<b>2</b>
2.1	Programversioner . . . . .	2
<b>3</b>	<b>Python3</b>	<b>2</b>
3.1	Installation av Python3 . . . . .	2
3.2	Installation av pip och virtualenv . . . . .	3
3.3	Köra en python3 fil . . . . .	3
<b>4</b>	<b>Atom</b>	<b>3</b>
4.1	Installation . . . . .	3
<b>5</b>	<b>Git</b>	<b>4</b>
5.1	Installation . . . . .	4
5.2	Komma igång . . . . .	4
<b>6</b>	<b>Flask och Jinja2</b>	<b>4</b>
6.1	Installation . . . . .	4
6.2	Användning . . . . .	5
<b>7</b>	<b>Felsökning och underhåll</b>	<b>5</b>
7.1	Felsökning Flask . . . . .	5
7.2	Underhåll . . . . .	5
7.2.1	Git . . . . .	5
7.3	Projektmodifiering . . . . .	6
7.3.1	Lägga till projekt . . . . .	6
7.3.2	modifiera ett existerande projekt . . . . .	6

## 1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Dokument skapat	20/9

## 2 Grundläggande information för installation

Allt kommer att installeras med hjälp av terminalen i Linux. Den öppnas med:

CTRL-ALT-t

För att följa manualen behöver man ha antingen Linux Mint eller Ubuntu installerat.

### 2.1 Programversioner

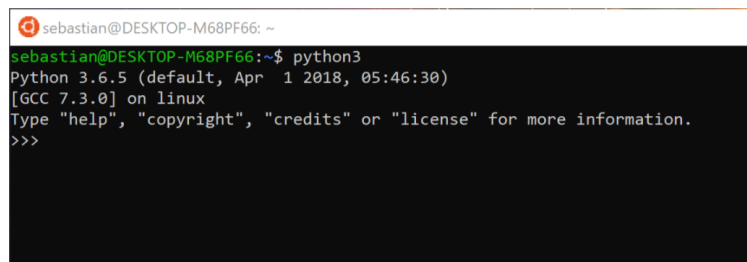
Nedan är en lista med versionerna på de program som finns med i den här manualen.

- python 3.6
- Jinja 2.0
- Flask 1.0.2
- Ubuntu 18.04
- Atom 1.30.0
- git 2.17.1
- HTML5
- CSS3

## 3 Python3

### 3.1 Installation av Python3

Prova skriv in python3 i terminalen och klicka på ENTER för att se om det är installerat. Om det ser ut som bilden så är det installerat.



```
sebastian@DESKTOP-M68PF66: ~  
sebastian@DESKTOP-M68PF66:~$ python3  
Python 3.6.5 (default, Apr 1 2018, 05:46:30)  
[GCC 7.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

Figur 1: En vild pythonterminal

Annars så måste vi installera det. Det gör vi genom att skriva:

```
$ sudo apt install python3
```

## 3.2 Installation av pip och virtualenv

Installation av pip är lite krångligare än python3, men inte jättesvår. Man får även med python3 virtualenv när man installerar.

```
$ sudo apt-get install python3-pip python3-dev build-essential python3-venv
```

```
$ sudo pip install --upgrade pip
```

```
$ sudo pip install --upgrade virtualenv
```

## 3.3 Köra en python3 fil

För att köra en python3 fil i terminalen behöver man ha:

```
#!/usr/bin/env python3
```

i toppen på .py filen. Första gången man ska köra den kommer det se ut ungefär som:

```
$ chmod u+x <filnamn>.py
```

```
$ python3 ./<filnamn>.py
```

“chmod” kommandot behöver man bara köra en gång per fil, då den gör så att man får rättigheter att köra den.

# 4 Atom

## 4.1 Installation

En editor som är lätt att använda är atom. Den installerar vi genom att skriva dessa kommandon:

```
$ curl -sL https://packagecloud.io/AtomEditor/atom/gpgkey | sudo apt-key add -
```

```
$ sudo sh -C 'echo "deb [arch=amd64] https://packagecloud.io/AtomEditor/atom/any any main" > /etc/apt/sources.list.d/atom.list'
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install atom
```

## 5 Git

### 5.1 Installation

De flesta linuxinstallationer kommer med git installerat men det kan vara bra att kolla så det är uppdaterat eller om det saknas. Skriv in dessa kommandon i ordning och följ instruktionerna på skärmen.

```
$ sudo apt-get update  
  
$ sudo apt-get upgrade  
  
$ sudo apt-get install git
```

### 5.2 Komma igång

När man installerat git, vill man få igång ett repository, en mapp som länkas mot github eller gitlab, beroende på vad du använder. Först behöver man en ssh-nyckel, som används för att identifiera datorn. Den skapar man med:

```
$ ssh-keygen -t rsa -C <Valfritt namn> -b 4096
```

De prompter som kommer upp kan man bara trycka förbi med ENTER. När den är skapad ska man kopiera publika nyckeln, och lägga till i SSH-nyckel kategorien på hemsidan. Detta kan man göra genom att öppna `id_rsa.pub` med hjälp av till exempel Atom. Kopiera allt med CTRL+A och CTRL+C.

Om du har skapat ett eget repository på motsvarande hemsida, står det hur man gör för att komma igång med Git lokalt. Om du har tillgång till ett repository som redan har filer i sig ska du göra följande:

```
$ cd <mapp>  
  
$ git init  
  
$ git remote add origin <git-url>  
  
$ git pull origin master
```

## 6 Flask och Jinja2

### 6.1 Installation

Det sista vi behöver är Flask och Jinja2 för att kunna göra en hemsida med både databas, och snygg framsida. Vi behöver även skapa en virtuell miljö, med hjälp av `virtualenv` som vi installerade tillsammans med PIP. Det gör man genom att skriva:

```
$ mkdir <projektnamn>  
  
$ cd <projectnamn>  
  
$ python3 -m venv venv
```

```
$ . venv/bin/activate
```

```
$ sudo pip3 install flask
```

Vi använder `. venv/bin/activate` kommandot för att komma in i vår virtualenv. För att komma ut ur virtualenv så skriver vi `deactivate`. Jinja2 är en del av flask och installeras tillsammans med flaskinstallationen.

## 6.2 Användning

För att göra en fil körbar i Flask så gå vi till mappen där filen finns och skriver:

```
$ export FLASK_APP=<filnamn>.py
```

```
$ flask run
```

Där `<filnamn>` är huvudfilen i projektet.

## 7 Felsökning och underhåll

### 7.1 Felsökning Flask

Flask sparar ingen log per automatik, utan skriver ut felmeddelanden i den terminal appen körs i. För att spara loggen, kan man till exempel skriva till `> log.txt` till sitt run kommando, så det ser ut som exemplet:

```
$ flask run > log.txt
```

Om du bara får

```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

så har du gjort rätt, och all annan info som Flask skickar när den kör, kommer hamna i `log.txt` För mer information om felhantering i Flask kan man gå in på [Flask Logging](#).

### 7.2 Underhåll

För att se till att allt är uppdaterat kan vi använda oss av kommadot:

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

#### 7.2.1 Git

Gör det som vana att alltid göra en “pull” innan man ska börja koda. Detta gör man med:

```
$ git pull
```

Detta gör att man inte får några konflikter med den “branchen” man arbetar i just nu. När man är färdigkodad, så behöver man “pusha” det man gjort. Detta gör man med:

```
$ git add .  
  
$ git commit -m '<Ett beskrivande meddelande om dagens arbete>'  
  
$ git push
```

Mer info om “branches” och annat om Git hittar du [här](#).

## 7.3 Projektmodifiering

Det finns i samma mapp som alla html-filer och så, en .json-fil, som agerar databas.

### 7.3.1 Lägga till projekt

För att lägga till ett färdigt projekt, börja med att kopiera html-mallen som finns och döpa om den till <projektnamn>.html med:

```
$ cp mall.html <projektnamn>.html
```

Sedan öppnar man databasen i Atom för att lägga till all information. Titta på hur strukturen är på de tidigare projekten, kopiera ett av dem, och byt ut informationen mot det som ska beskriva det projekt du vill lägga till.

### 7.3.2 modifiera ett existerande projekt

För att modifiera ett existerande projekt, öppna databasen i Atom. Leta sedan rätt på det projekt du vill modifiera och ändra på den aktuella datan.