

# TDP003 Projekt: Egna datormiljön

## Installationsmanual för Portföljsystem

IP1 2018

## Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
0.1	Template med grunduppgifter skapad	18-09-12
0.2	Första ifyllda version	18-09-27
0.3	Korrigerad slutversion	18-10-04

## Innehåll

<b>1</b>	<b>Introduktion</b>	<b>2</b>
<b>2</b>	<b>Allmän information</b>	<b>2</b>
2.1	Linuxversion . . . . .	2
2.2	Terminalen . . . . .	2
2.3	Enkla terminalkommandon . . . . .	2
<b>3</b>	<b>Installation av program</b>	<b>3</b>
3.1	Python3 . . . . .	3
<b>4</b>	<b>Pip3</b>	<b>4</b>
4.1	Felsökning och alternativa situationer . . . . .	4
4.2	Virtuell miljö . . . . .	4
4.2.1	Virtuell miljö via Python3 venv . . . . .	5
4.2.2	Virtuell miljö via Virtualenv . . . . .	5
4.3	Installation av Flask . . . . .	5
4.4	Flaskpaket . . . . .	6
4.4.1	Din första app med Flask . . . . .	6
4.5	Installation av Jinja2 . . . . .	7
4.6	Log från Flask . . . . .	7
4.7	Installation av Git och hämtning av portfolio . . . . .	8
4.8	Avstämning av installerad mjukvara . . . . .	9
4.8.1	Python3 & Pip . . . . .	9
4.8.2	Flask och Jinja2 . . . . .	9
<b>5</b>	<b>Texthanteraren Atom</b>	<b>10</b>
5.1	Felsökning och alternativa situationer . . . . .	10
<b>6</b>	<b>Felsökning</b>	<b>10</b>
<b>7</b>	<b>Underhåll</b>	<b>11</b>
7.1	Källkod . . . . .	11
7.2	Projektmodifiering . . . . .	11
7.2.1	Lägga till projekt . . . . .	11
7.2.2	Modifiering av existerande projekt . . . . .	12

# 1 Introduktion

Vänligen läs igenom hela manualen innan installation. Använd manualen under installation samt vid användning och underhåll av portfolion. Manualen innehåller även lösningar till vanliga fel som kan uppstå vid installation samt vid användningen av portfolion.

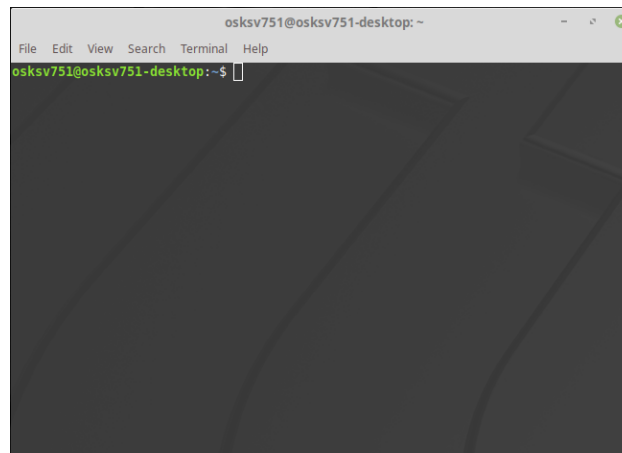
## 2 Allmän information

### 2.1 Linuxversion

Denna manual är skriven med terminalkommandon för debianbaserade Linuxdistributioner, som till exempel Ubuntu och Linux Mint.

### 2.2 Terminalen

Innan du börjar att installera verktygen kan det vara bra att veta några saker om Linux. I de flesta stegen kommer vi att använda den inbyggda terminalen. Den används för att ladda ner, installation av program och verktyg samt för att navigera till olika mappar i datorn. Terminalen kan även användas till att öppna, titta och ändra på filer. Terminalen tolererar inte stavfel. Vid stavfel kommer den inte att förstå vad du vill göra och du kommer att få ett felmeddelande. Var noga med stavningen. För att öppna terminalen så kan du trycka på **Ctrl - Alt - t**. Ett terminalfönster bör nu ha öppnats som liknar det i figur 1, men med ditt användarnamn istället.



Figur 1: Terminalen

### 2.3 Enkla terminalkommandon

I terminalen så kan man skriva kommandon och trycka **Enter**. Då tolkar datorn kommandot och utför det kommandot du angett. Ett bra kommando att känna till är **ls**, som listar alla mappar och filer i den mappen du står i. Ett annat användbart kommando är **cd <mappnamn>** som används för att förflytta dig till en angiven målapp. **cd ..** gör att du flyttar dig en mapp upp. Om vi säger att du ska ställa dig i en viss

mapp får du använda `cd` tills det står den angivna mappens namn i terminalfönstret. `cd ~` kan du skriva för att ta dig tillbaka till hemmappen.

### 3 Installation av program

För att kunna använda din portfolio behöver du installera en samling program och Python-paket. Nedan finner du de program du behöver installera:

- Python 3
- Pip
- Flask
- Jinja2
- Git

Installationsanvisningar för varje program finns under respektive sektion. Terminalkommandon markeras med `user@computer:~$` före kommandot. Du finner även anvisningar till hur du laddar ner de filer som behövs för att använda din portfolio.

Det rekommenderas starkt att uppdatera alla repositories vilket du gör med följande kommando:

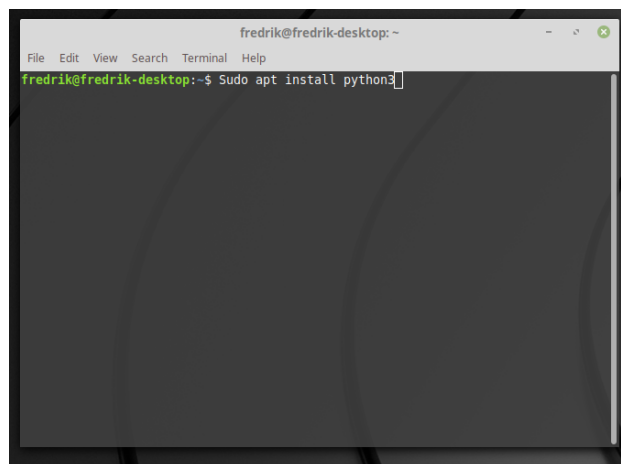
```
user@computer:~$ sudo apt-get update
```

#### 3.1 Python3

För att installera Python3 till din dator behöver du öppna terminalen och skriva följande:

```
user@computer:~$ sudo apt-get install python3
```

När det står att du ska skriva något tryck alltid enter efter det som står. Det kan vara att Python redan är installerat och då kommer det stå *python3 is already the newest version* i terminalen beroende på vilket operativsystem du har. Ingenting kommer att hända.



Figur 2: Installera Python

**Sudo** skriver du innan nästkommande kommando för att använda filer med administratörsrättigheter, och detta kräver att du skriver in ditt lösenord som bekräftelse. `apt-get install` betyder helt enkelt att du

ska visa att du ska installera något. Du kan använda Python3 genom att skriva `python3` i terminalen. Du märker ifall `python3` är startat ifall du får ett litet välkomstmeddelande som b.l.a. innehåller din version av python. Raden där du skriver in kommando kommer även börja med `>` ifall python är igång.

För att se om installationen gått som den ska kan du skriva följande när du står i hemmappen (för att komma till hem-mappen i terminalen kan du alltid skriva `cd ~`). Skriv `/usr/bin/python3 --version`, då borde terminalen skriva vilken version du har. Flask behöver version 3.4 eller nyare. **Om något går fel!** Du kan alltid gå in på <https://realpython.com/installing-python/> och läsa hur du ska installera Python3 beroende på vilket operativsystem du har.

## 4 Pip3

Pip är nästa verktyg som kommer behövas för att kunna hantera projektet. Första steget är att uppdatera systemets repository. För att uppdatera repository:n kan du använda kommandot:

```
user@computer:~$ sudo apt-get update
```

Nästa steg är just installationen av pip. Tänk på att du vill installera rätt version av pip för rätt version av python. Den versionen av pip du vill installera är pip3, eftersom den versionen fungerar för python3. För att installera pip3 så använder du kommandot:

```
user@computer:~$ sudo apt-get install python3-pip
```

Du kan sedan kontrollera pip3-versionen med hjälp av kommandot `pip3 -V`. För närvarande är pip3 9.0.1-2 den senaste versionen. Om du vill göra en dubbelkontroll av att du har den senaste versionen av pip3 kan du skriva kommandot:

```
user@computer:~$ sudo apt-get install python3-pip
```

Den kommer då ladda ner den nyaste versionen av pip, eller om du redan har den senaste versionen kommer det att stå vilken version som är den senaste. Om den senaste versionen av pip3 inte finns installerad kan man även för säkerhets skull köra kommandot:

```
user@computer:~$ pip3 install --upgrade pip
```

för att få den senaste versionen. Användbara kommandon för pip3 är:

```
user@computer:~$ pip3 install <package_name>
```

för att installera paket,

```
user@computer:~$ pip3 help
```

för att få flera kommandon och

```
user@computer:~$ pip3 search <sökord>
```

för att söka bland python-paket.

### 4.1 Felsökning och alternativa situationer

Terminalen kan be dig om lösenord eller bekräftelse när du vill installera någonting. Om den behöver bekräftelse ger du det med hjälp av `y` (vilket står för yes) eller `j` ifall du använder svenska som standardspråk. Det kan mot all förmodan även vara något verktyg som behövs som inte redan är installerat av någon anledning. Terminalen är oftast bra på att skriva ut vad det är som saknas och ibland vad du behöver skriva för att installera det.

### 4.2 Virtuellt miljö

För att inte få kompatibilitetsproblem med ditt system, projektets bibliotek samt pythonversion är det bra att du skapar en virtuell miljö att hantera de i.

#### 4.2.1 Virtuellt miljö via Python3 venv

Detta kan ske med hjälp av Python3 venv. Ställ dig i den mapp där du vill att projektets modulmiljö ska sparas. Detta kan göras med hjälp av terminalkommandot `cd <mappnamn>`. För att skapa en ny mapp använd terminalkommandot `mkdir <mappnamn>`.

```
user@computer:~$ mkdir TDP003
```

```
uRser@computer:~$ cd TDP003
```

Alternativt kan du via programmet `files` navigera/skapa en mapp. Om terminalen inte redan står i den skapade målmappen kan du högerklicka och välja alternativet 'Open in Terminal' för att få en terminal som står i mappen.

Med kommandot:

```
user@computer:~$ python3 -m venv <venv>
```

Då skapas det en virtuellt miljö där du nu kan gå vidare med installation av kommande moduler. Parametern `<venv>` bestämmer namnet på den virtuella miljön och kan således bytas ut om det önskas.

#### 4.2.2 Virtuellt miljö via Virtualenv

Om du använder dig av en tidigare version av Python eller av andra anledningar vill använda en extern virtuellt miljö kan du installera Virtualenv. Detta gör du via terminalkommandot:

```
user@computer:~$ python3 -m pip install --user virtualenv
```

Aktivera miljön i rätt mapp:

Navigera i terminalen (`cd <path>`) till din portfoliomapp. Från mappen i terminaler skriver du kommandot:

```
user@computer:~$ python3 -m virtualenv env
```

När detta gjorts skapas en mapp i portfolion som heter "env". (OBS! Om du använder GitLab eller liknande versionshanteringssystem bör du exkludera denna mapp från repository:t.)

När du ska arbeta med projektet genom din virtuella miljö måste du aktivera virtualenv varje gång för att kunna använda de paket du kommer installera i denna virtuella pythonversion. Detta gör du genom terminalkommandot (från projektmappen):

```
user@computer:~$ source env/bin/activate
```

I terminalen bör det nu stå (venv) längst till vänster på kommandoraden. Detta indikerar att din virtuella miljö är aktiv. Exempel:

```
( venv ) user@computer:~$
```

För att gå ut ur din virtuella miljö (när du är klar med arbetet med portfolion) skriver du:

```
( venv ) user@computer:~$ deactivate
```

### 4.3 Installation av Flask

1. Installera Flask med följande kommandon:

```
user@computer:~$ pip install Flask
```

2. Kontrollera om Flask är installerad med följande kommandot:

```
user@computer:~$ pip freeze
```

Listan med installerade paket ska också innehålla Jinja2.

## 4.4 Flaskpaket

Vid installation av Flask och även Jinja installeras flera paket som du kan komma att använda dig av i framtiden när du arbetar med olika projekt. De flesta av dessa paket installeras oftast utan problem, men ibland fungerar installationen inte på det vis du förväntat dig. Vid installationen av Flask kan det komma upp en röd text där det står:



Failed building wheel for MarkupSafe

Figur 3: Fel felmeddelande av installation av FLASK

Det är även vanligt att det står samma sak för itsdangerous paketet. Det är inte garanterat att dessa paket kommer att användas men skulle dessa paket inte fungera när de bör användas blir det problem. Du kan åtgärda detta problem genom att följa dessa steg:

1. Installera wheel med följande kommando:  

```
user@computer:~$ pip install Flask
```
2. Installera paketet genom att använda paketets namn (exempel, its markupsafe och itsdangerous):  

```
user@computer:~$ pip install markupsafe  
user@computer:~$ pip install itsdangerous
```

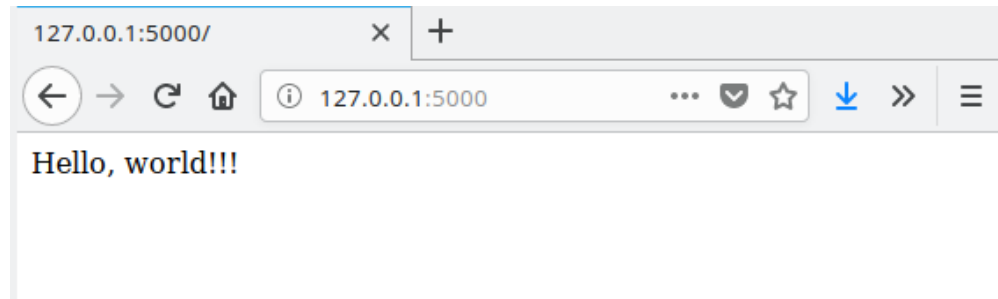
### 4.4.1 Din första app med Flask

1. Öppna Emacs eller annan texthanterare och skriv:  

```
from flask import Flask  
  
app = Flask(__name__) #create a Python object app  
  
@app.route('/') #to handle web request to the '/'  
def hello():  
    return 'Hello ,_World!!! '  
  
if __name__ == '__main__':  
    app.run()
```
2. Spara filen som flaskimport.py (Filen finns också i katalogen **examples** i samma repository som denna installationsmanual)
3. Öppna terminalen och kör applikationen med:  

```
user@computer:~$ python3 flaskimport.py
```
4. Flask kör på den lokala adressen  
  

```
http://127.0.0.1:5000/
```
5. Öppna adressen i en webbläsare eller klicka på länken som visas i terminalen. Resultatet visas i figur 4.



Figur 4: Min första app

## 4.5 Installation av Jinja2

Jinja2 är ett mallspråk till Python som används för att förenkla skapandet av webbsidor. Detta är med i listan eftersom det räknas som en separat modul, men om du har följt föregående instruktion (4.3) behövs ingen annan installation utföras här eftersom installationen av Flask även installerar Jinja2.

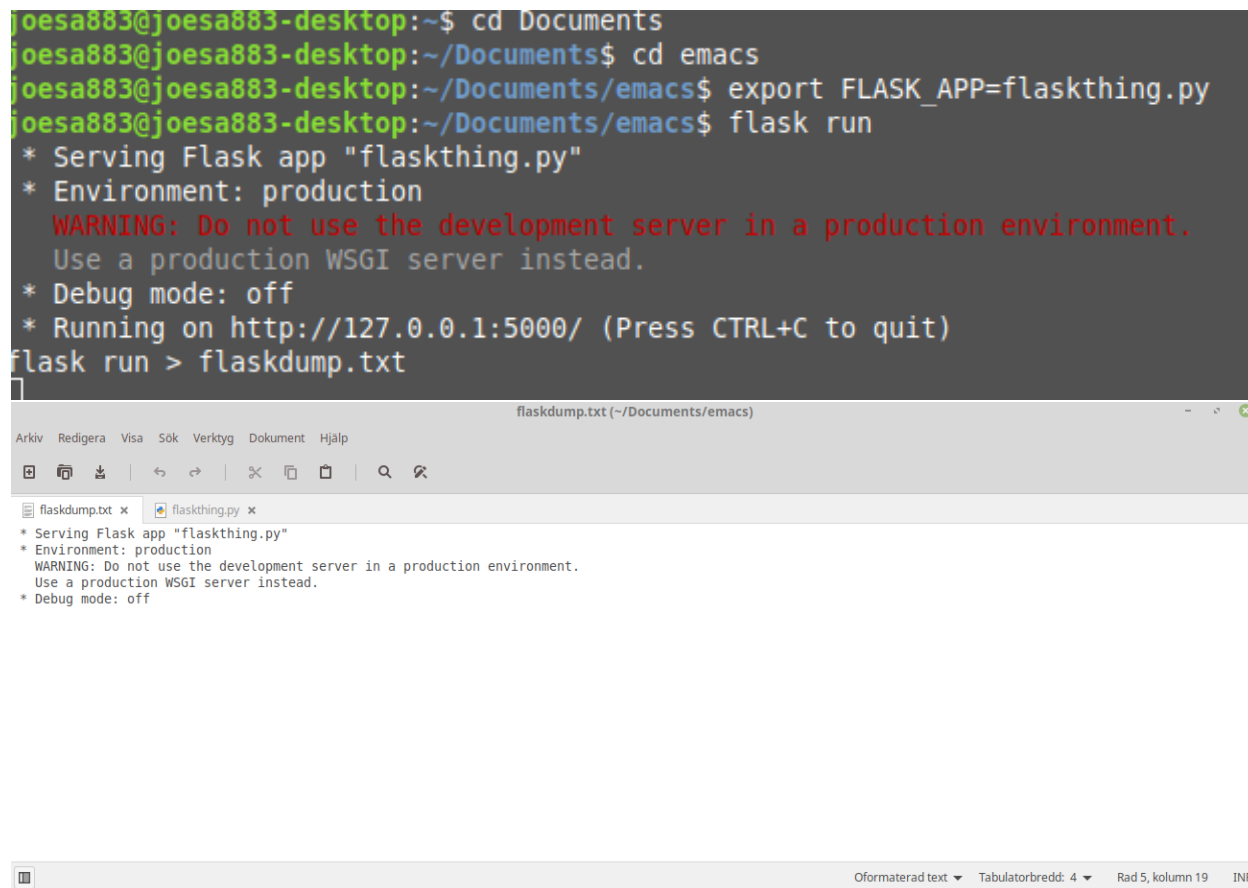
## 4.6 Log från Flask

Du sparar log från Flask genom att skriva följande i terminalen:

```
user@computer:~$ export FLASK_APP=flaskthing.py
user@computer:~$ flask run
user@computer:~$ flask run > flaskdump.txt
```

Flaskloggen sparas då i den specificerade filen, i detta fall flaskdump.txt (skapar också den specificerade filen om den inte redan existerar). Se figur 5 för ett bildexempel.





The screenshot shows a terminal window at the top with the following commands and output:

```
joesa883@joesa883-desktop:~$ cd Documents
joesa883@joesa883-desktop:~/Documents$ cd emacs
joesa883@joesa883-desktop:~/Documents/emacs$ export FLASK_APP=flaskthing.py
joesa883@joesa883-desktop:~/Documents/emacs$ flask run
* Serving Flask app "flaskthing.py"
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
flask run > flaskdump.txt
```

Below the terminal is an Emacs editor window titled "flaskdump.txt (~/.Documents/emacs)". The editor shows the same log output as the terminal. The Emacs interface includes a menu bar (Arkiv, Redigera, Visa, Sök, Verktyg, Dokument, Hjälp), a toolbar, and a tab bar with two tabs: "flaskdump.txt" and "flaskthing.py".

Figur 5: Flask loggning

## 4.7 Installation av Git och hämtning av portfolio

Git är det största verktyget för versionshantering, men i detta fall kan det vara bra för att just ladda ner portfolion.

Installera Git genom att öppna terminal och skriva:

```
user@computer:~$ sudo apt-get install git
```

När git är installerat är det sedan möjligt att hämta hem själva portfolioprojektet där all källkod finns. Om du redan har satt upp en virtuell miljö i en specifik mapp enligt tidigare instruktioner kan det vara lämpligt att kлона projektet in i samma mapp. Detta görs genom att först kлона det Git-repo som vi vill ha med kommandot:

```
user@computer:~$ git clone git@gitlab.ida.liu.se:user/repo.git målmapp
```

Där git@gitlab.ida.liu.se är den generella adressen till den instans av git som används, i detta fall LiUs egna. User är den användarens repository som vi vill kлона och repo.git är själva repositoryet i sig. All denna information går att hitta på projektets git-sida. Målmapp är helt enkelt den mapp som vi vill kлона repot in i.

Det är bra att känna till hur du laddar ner den senaste versionen av portfolioprojektet, vilket enkelt görs genom att köra kommandot:

```
user@computer:~$ git pull
```

Observera att du måste stå i samma mapp som du klonade repositoret till när du kör kommandot.

I den nya mappen bör det finnas en `data_example.json` fil med lite testdata som visar hur projekt i portfolion är strukturerade. Portfolion använder sig dock av `data.json` för att läsa in de projekt som finns, och därför behöver `data_example.json` kopieras till `data.json` med kommandot:

```
user@computer:~$ mv data_example.json data.json
```

Det bör nu finnas en användbar version av projektet på din dator där du kan lägga till projekt i `json.data`. Mer information om detta finns under sektionen om underhåll.

## 4.8 Avstämning av installerad mjukvara

För att se till att samtliga installationer lyckats bör du kontrollera att du har all mjukvara tillgängligt på systemet.

### 4.8.1 Python3 & Pip

För att kontrollera om Python3 kör du följande kommando:

```
user@computer:~$ python3 --version
```

Utskriften bör bli *Python 3.6.5* eller högre

Pip kan kontrolleras på samma sätt.

```
user@computer:~$ pip3 --version
```

Terminalen bör visa *pip 9.0.1* och sökvägen där pip installerar Pythonpaketen. Versionsnumret ska ej vara lägre än detta.

### 4.8.2 Flask och Jinja2

Flask kan du kontrollera om det finns installerat på två sätt: det tidigare förslaget, men även med hjälp av Pip. Jinja2 däremot kontrolleras endast med hjälp av Pip.

För att se om Flask finns skriver du följande i terminalen: `user@computer:~$ Flask --version`

Terminalen bör visa något i stil med:

```
Flask 0.12.2
```

```
Python 3.6.5 (default, Apr 1 2018, 05:46:30)
```

```
[GCC 7.3.0]
```

Det andra sättet som visar både Flask och Jinja2 hittar du genom att skriva följande i terminalen:

```
user@computer:~$ pip3 list
```

Om Flask och Jinja2 visas i listan har installationen lyckats.

Problem	Förslag
Ladda ner fel version av pip för fel Python version	Se till att kommandot <code>sudo apt-get install python-pip3</code> skrevs in rätt med 3 efter pip för att pip versionen som passar med Python3 är vald

## 5 Texthanteraren Atom

För nya användare av texthanterare kan Emacs vara krångligt att använda. Vi rekommenderar texthanteraren Atom som substitut då den är användarvänlig och lätt att lära sig. Atom har även inbyggd versionhantering i Git. Atom finns inte med i datorns standardrepository och du måste lägga till ett repository för att installera det. Detta gör du enklast genom att skriva eller kopiera in följande kommandon:

```
user@computer:~$ curl -sL https://packagecloud.io/AtomEditor/atom/gpgkey | sudo apt-key add -
user@computer:~$ sudo sh -c 'echo deb [arch=amd64] https://packagecloud.io/AtomEditor/atom/any/any main"> /etc/apt/sources.list.d/atom.list'
```

När du har gjort detta ska du köra kommandot:

```
user@computer:~$ sudo apt-get update
```

för att datorn ska söka igenom alla tillagda repositories efter tillgängliga paket. Skriv sedan följande kommando för att installera Atom:

```
user@computer:~$ sudo apt-get install atom
```

För att utnyttja Atoms väldigt användbara git-funktioner kan du öppna mappen där du har ditt git repository. Ändrar du filer i den här mappen kan du enkelt pusha dem med hjälp av den uppekande pilen längst ner till höger i programmet. För att hämta hem filer från projektet som tillhör repository:n så trycker du istället på den nerpekande pilen.

### 5.1 Felsökning och alternativa situationer

Skulle det vara att pilarna inte visas längst ner i programmet beror det på att du inte befinner dig i en mapp där det finns ett repository. Mer om hur du skapar repositories och hanterar dem finns i delen som handlar om Git.4.7

## 6 Felsökning

### Det går inte att installera Virtuellt Miljö:

Ett vanligt fel är att Python misslyckas med att skapa en virtuell miljö med kommandot `python3 -m venv venv`. I sådana fall behöver du installera Python-setuptools med följande kommando:

```
user@computer:~$ sudo apt-get install python-setuptools
```

### Det går inte att använda pip för att installera Flask

Pip är en modulhanterare för Python. För att installera pip följ kommandot nedan:

```
user@computer:~$ sudo apt-get install python3-pip
```

### ModuleNotFoundError: No module named 'flask'

Kontrollera att din virtuella miljö(virtual environment) är aktiverad. (venv) visas innan prompten. Annars aktivera miljön med:

```
>>> source venv/bin/activate
```

## Problem med Flask

Ibland kan du få problem vid uppstart av Flask där till exempel en port är blockerad eller att du kör en annan serverinstans på samma port som Flask vill köra på. Det kan leda till följande felmeddelande:

```
socket.error: [Errno 13] Permission denied
```

Det går att lösa genom att lägga till en extra flagga till kommandot `flask run` som gör att du kan köra `flask run --port=9000`. Det kommer köra Flask på porten 9000 som med stor sannolikhet inte kommer vara upptagen.

Om du fortfarande får problemet kan du testa ett annat nummer på till exempel 9001.

Det kan också vara bra att lista alla upptagna portar med hjälp av följande kommando:

```
user@computer:~$ netstat -lnput
```

som visar en tabell av nätverkinformation. Under titeln *Local Address* visas rader av ip-adress och Port separerade med kolon.

## 7 Underhåll

### 7.1 Källkod

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

```
user@computer:~$ git pull
```

Detta gör att du inte får några konflikter med den “branchen” du arbetar i just nu. Det ser också till att allt är uppdaterat och att du inte jobbar med gammal kod.

Skulle något bli konstigt eller om du får ett felmeddelande finns det mycket info på GitHubs hemsida som du hittar här.

### 7.2 Projektmodifiering

#### 7.2.1 Lägga till projekt

För att lägga till ett färdigt projekt räcker det med att modifiera `data.json` filen, som finns i rotkatalogen på projektet. Detta kan du göra med förslagsvis Atom eller annan godtycklig texthanterare. Se hur strukturen för ett redan existerande projekt ser ut och skriv en ny projektstruktur. Minimikraven för att ett projekt ska läggas till är projektnamn och projektid. Övriga fält kan lämnas tomma. Exempel:

```
{
  "start_date": "2018-09-20",
  "short_description": "Hell Yeah",
  "course_name": "TDP003",
  "long_description": "PANICCCCCCCCC",
  "group_size": 2,
  "academic_credits": "-100HP",
  "Aneurysms_had": "infinite",
  "external_link": "www.call911.com",
  "small_image": "brainscan.jpg",
  "techniques_used": [
```

```
"python"  
"CSS"  
"C++"  
],  
"project_name": "create your own portfolio, with google as tutor",  
"course_id": "TDP003",  
"end_date": "2021-12-24",  
"project_id": 1337,  
"big_image": "chickengoespanic.gif"  
},
```

### 7.2.2 Modifiering av existerande projekt

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