Humanistens Digitale Værktøjskasse

Indledning

Denne tekst er et overblik over hvad der gennemgås i workshoppen "Humanistens Digitale Værktøjskasse". Dette dokument forsøger at give et overblik over hvordan de forskellige elementer af workshoppen kan bruges også efter workshoppen er slut. Den er altså tænkt, som et referencepunkt, man kan vende tilbage til, når man er i tvivl om hvordan man anvender materialet fra workshoppen.

Hvad skal man bruge til workshoppen?

Til workshoppen skal vi hente et programmer: Miniconda Miniconda er det program, som vi skal bruge til at køre vores kode i. Det består af følgende tre ting: kodesproget python, en pakke-manager og en miljø-manager. Dette kan lyde forvirrende. Kort fortalt, er det den måde, vi installere kodesproget, som vi bruger under workshoppen på vores computere.

Miniconda hentes her: https://docs.conda.io/en/latest/miniconda.html

Hvordan åbner jeg miniconda?

Miniconda (fremover conda) åbnes forskelligt alt efter om du bruger Windows eller Mac. Fælles for systemerne er, at conda og det bagvedliggende kodesprog python bruges igennnem et Command Line Interface (CLI). Vi skal i workshoppen arbejde med vores computer på en måde, der er fundamentalt anderledes end hvad vi normalt er vandt til at gøre. Vi kommer til at ligge det grafiske interface (vores skrivebord og billeder af mapper) på hylden og interagere med computeren igennem tekst og commands. En af de helt store forskelle fra vores "hverdagsbrug" af computeren og til denne Command Line brug er, at vores mus/trackpad ikke spiller en særligt stor rolle. Det er ikke muligt, at klikke på noget med musen. Istedet skal vi fortælle computeren, hvad vi vil have den til med tekst kommandoer.r

Windows

På Windows skal vi åbne programmet Anaconda powershell promt (miniconda), som ADMINISTRATOR. Dette Command Line Interface (CLI) skal vi arbejde i under workshoppen. Det er som sagt en anden måde at arbejde med sin computer på, som kan kræve lidt tilvænning og oplæring, nu hvor de fleste af os er vokset op med et grafisk skrivebord på computeren.

Mac

På Mac, skal man åbne programmet ”terminal” efter installationen af miniconda. Dette kan gøres ved at bruge genvejen cmd + mellemrum og så søge "terminal". Hvis man ikke vil bruge cmd + mellemrum genvejen kan programmet normalt findes og åbnes i mappen Hjælpeprogrammer under programmer. Terminalen er Mac computeres Command Line Interface (CLI) som vi skal arbejde i under workshoppen. Det er som sagt også en anden måde at arbejde med sin computer på, som kan kræve lidt tilvænning og oplæring

Opsætning af conda

Første gang man bruger conda, skal man lave det miljø, som man vil arbejde i. Det er her man sikre sig, at forskellige python projekter ikke "ødelægger" hinanden. Man kan se på disse virtuelle miljøer som et bur man har sit kæledyr i. Hvis vi skal blive i python sprogets dyreunivers kan man sige at hvert conda miljø man laver sikrer, at ens forskellige kode ikke spiser og ædelægger hinanden, som to forskellige rovdyr, der skal leve sammen i det samme bur. For at lave et nyt miljø i conda skriver man følgende command:

```
conda create --name "toolbox" python=3.9
```

Når dette miljø er lavet, skal man aktivere det. Det gør man ved at skrive:

```
conda activate "toolbox"
```

For at deaktivere et conda miljø skrives:

```
conda deactivate
```

Dette er det grundlæggende conda, som vi bruger i løbet af workshoppen. Hvis man efter at have lavet miljøet "Toolbox" lukker vinduet ned, skal man aktivere sit miljø igen. Nedenfor er en liste af brugbare conda commands:

Command	**Forklaring**
conda create - -name "name" python=3.8	Laver et nyt miljø i conda, som kører python 3.8, der hedder name
conda activate "name"	Aktiverer conda miljøet "name"
conda deactivate	Deaktiverer det aktive conda miljø
conda install "package"	Installerer pakken "package" i conda
conda uninstall "package"	Afinstallere pakken "package"
conda update	Opdaterer conda
conda update "package"	Opdaterer pakken "package"

Sådan navigerer man i sit Command Line Interface

Når man bruger programmerne PowerShell (Windows) & Terminal (Mac) interagerer man som tidligere nævnt med computeren på en helt anden måde. Grunden til at vi skal denne vej ind, er fordi vi denne vej kan få lov til mange andre ting med computeren. Men for at det kan lade sig gøre skal vi vide, hvordan vi bevæger os rundt i systemet, når vi ikke har en mus til at klikke på mapperne. Nedenfor har jeg indsat en tabel, der indeholder nogle af de mest brugbare commands til at finde rundt i Shell og Terminal:

Command	**Forklaring**
cd	Command der skifter placering. Den skal efterfølges af hvor man gerne vil hen i sin computer
cd ./mappe	Skifter placering til mappen "mappe" under den mappe, som vi befinder os i nu
cd ..	Skifter placeirng til mappen ovenover den mappe man er i når commanden køres

Command	Forklaring
cd ~	SKifter til computerens hjemmeplacering
cd c:/users/name/photos	Skifter til mappen photos i mappen name i mappen users på C-drevet
dir	Viser os indholdet af den nuværende placering
ls	Viser os indholdet af den nuværende placering

Installation af python moduler

Når nu vi har skabt og aktiveret vores conda miljø skal vi have installeret de python-moduler vores programmer er afhængige af. I mappen på workshoppens github, som vi har downloaded er der en fil, der hedder *requirements.txt*. Den indeholder informationer om de python moduler vi skal bruge. For at installere disse moduler skriver vi følgende kode i vores CLI efter vi har navigeret til vores mappe:

```
pip install -r requirements.txt
```

Brug et command line program

Fra vores command line kan vi køre programmer, der ikke har grafiske interfaces. De tre programmer som introduceres i denne workshop køres alle fra commannd line. Python programmer køres generelt på følgende måde:

1. Naviger til programmets placering
2. Kald på programmet på følgende måde:

```
python program.py
```

Nogle python programmer har brug for at man også giver dem argumenter igennem command line. Det kan fx være inputs og outputs. Fx kan et progam skulle bruges på følgende måde, hvilket vi skal bruge når vi skal arbejde med workshoppens programmer:

```
python program.py INPUT OUTPUT
```

Et tredje element, som kommer i brug igennem workshoppen er brug af -flags. -flags kan bruges til mange ting, fx kan de bruges til at få mere at vide om programmet og ændre ved nogle indstillinger. De fleste programmer der kører over command line har et -h/--help flag, som vises sådan:

```
python program.py -h
```

Hvis det ikke er et help-flag, kan det muligvis bruges sammen med INPUT og OUTPUT:

```
python program.py -flag INPUT OUTPUT
```

Det første program i workshoppen - selectivecopy.py

Dette program kan kopiere mange filer på en gang. I workshoppen bruges det til at kopiere forskellige pdf-filer fra flere mapper til en samlet mappe. Scriptet virker på alle filer der har samme endelse. Jeg bruger det

fx mod slutningen af et semester, til at samle alle tekster fra et kursus til en samlet mappe. For at bruge programmet gøres følgende:

1. Igennem Command Line navigeres der til programmets placering
2. Skab et overblik over programmets funktion:

```
python selectivecopy.py -h
```

Brug programmet:

```
python selectivecopy.py INPUT OUTPUT FILETYPE
```

Et eksempel kunne være følgende:

```
python selectivecopy.py ~/Pictures ~/Desktop/billeder JPG
```

Eksemplet ovenfor tager alle filer der har endelsen JPG i mappen Pictures og kopierer dem til en ny mappe på skrivebordet der hedder *billeder*. Programmet kigger i mappen *~/Pictures* og alle dens undermapper og kopierer alle filer der ender med JPG til den nye mappe. Man kan også flytte alle PDF-filer. Dette gøres ved at ændre FILETYPE argumentet til PDF istedet for JPG.

Det andet program i workshoppen - rotatepdf.py

Workshoppens næste element er programmet rotatepdf.py, dette program kan som navnet antyder rotere PDF-filer. Ligesom programmet selectivecopy.py skal dette program også bruges igennem vores Command Line Interface. Forestil dig, at du enten har scannet nogle sider fra en bog eller har downloadet tekster fra Brightspace, som vender forkert. Istedet for at skulle vende alle sider om individuelt mens der læses, kan programmet vende hele filer på en gang. Det er også muligt at vende flere hele PDF-filer på en gang. For at bruge programmet gøres følgende i vores CLI:

1. Igennem Command Line navigeres der til programmets placering
2. Skab et overblik over programmets funktion:

```
python rotatepdf.py -h
```

1. Her ses lidt mere information om, hvordan programmet fungerer. Man kan også læse, at programmet har et "*optional argument*", som hedder -b, mere om dette lidt senere.
2. Hvis man vil rotere en enkelt fil skrives følgende:

```
python rotatepdf.py INPUT OUTPUT VALUE
```

- Ovenfor er INPUT stien til filen man vil rotere.
- OUTPUT er, hvor den roterede fil skal gemmes.
- VALUE er hvor mange grader filen skal roteres med uret. Det er her muligt at vælge enten 90, 180 eller 270.

Et eksempel på dette kunne være følgende:

```
python rotatepdf.py ~/Documents/fil_der_vender_på_hovedet.pdf ~/Documents/roteret_fil.pdf 180
```

Det ovenfor nævnte eksempel vender vores vil "*fil_der_vender_på_hovedet.pdf*" 180 grader og gemmer den med navnet "*roteret_fil.pdf*".

Som ovenfor nævnt kan programmet også håndtere flere filer på en gang. Dette gøres med -flaget -b som står for batch. For at bruge programmets batch funktion bruges programmet på følgende måde:

1. Igennem Command Line navigeres der til programmets placering
2. Brug programmet med batch flag:

```
python rotatepdf.py -b INPUT OUTPUT VALUE
```

- I dette tilfælde er INPUT stien til den mappe hvor filerne er.
- OUPUT er stien til den mappe hvor de roterede filer skal gemmes.
- VALUE er den værdi alle filerne skal roteres med uret.

Et eksempel på denne brug kunne være følgende:

```
python rotatepdf.py -b ~/Documents/filer_til_rotering ~/Documents/roterede_filer 90
```

I det ovenfor nævnte eksempel roteres alle pdf-filer i mappen "*filer_til_rotering*" 90 grader med uret og gemmes i mappen "*roterede_filer*".

Det trejde program i workshoppen - pdfannots.py

Workshoppens tredje program hedder pdfannots.py. Scriptet er oprindeligt udviklet til at reviderer akademiske artikler og konferencebidrag. Det kan dog også hjælpe os andre, som læser, overstreger og skriver kommentarer i PDF-filer. Scriptets kernefunktion er, at trække kommentarer og overstregninger ud af PDF-filer. Lad os kigge på det:

Når vi skal bruge scriptet, skal vi gøre følgende i vores CLI:

1. Først naviger hen til scriptet i terminalen
2. Dernæst aktiver scriptet ved at skrive:

```
python pdfannots.py INFILE
```

- INFILE er her stien til den pdf-fil, som man vil trække annoteringer ud af.

Dette vil give alle annoteringer direkte i det vindue man sidder med og man kan så kopiere teksten videre til Word, OneNote, Notion, EverNote eller hvilket som helst program, som man foretrækker at samle sine notater i. En anden mulighed er, at gemme den udtrukne tekst i en tekstfil, Dette kan gøres på følgende måde:

1. Naviger hen til scriptet i terminalen
2. Aktiver scriptet ved at skrive:

```
python pdfannots.py -o output.txt INFILE
```

- Her bruger vi -o flaget som giver os outputtet i en fil, som vi selv kan navngive. I dette tilfælde kommer filen til at hedde output.txt.
- INFILE er stadig stien til den pdf-fil, som man vil trække annoteringer ud af.

Med dette program gælder det samme, som med de andre, hvis man navigerer hen til scriptet og skriver følgende får man vist en hjælpemenu. Med dette program er der lidt flere funktioner at vælge imellem:

```
python pdfannots.py -h
```