# Tekst-adventure – del 1

ms

November 2021

I dette projekt skal I bygge jeres eget tekstbaserede adventure-spil, i stil med f.eks:

- Dreamhold
- The Hitchhiker's Guide To The Galaxy-spillet
- Zork

Fokus for projektet er kontrolflow – I skal altså øve jer på at bruge Python's if-sætninger og while-løkker i et større projekt, og i at dokumentere jeres programs kontrolflow ved hjælp af flow-diagrammer.

## 1 Hvad skal afleveres?

I skal arbejde sammen i grupper på 2-3 personer. Hver gruppe skal aflevere en zip-fil der indeholder:

- Jeres program.
- Et flow-diagram, der viser det overordnede kontrolflow i programmet.
- En kort guide til brugeren, der forklarer hvordan man kommer igang med at spille jeres spil (kan evt. inkluderes i programmet som en `help`-kommando).

## 2 Jeres program

Med udgangspunkt i det udleverede program, `adventure-template.py`, skal I lave et spil hvor spilleren, ved hjælp af tekst-kommandoer, kan bevæge sig fra rum til rum og undersøge sine omgivelser. Figur 1 på side 3 viser et flow-diagram over den udleverede kode.

Senere, i del 2 af projektet, skal I også lave et inventory-system til spillet, men det behøver I ikke tænke på nu.

### 2.1 Spillets omfang

Når man udvikler spil, så tager det typisk rigtig meget tid at finde på selve indholdet til spillet, og at producere det nødvendige lyd, grafik, og tekst. I dette tilfælde skal I selvfølgelig ikke bruge hverken lyd eller grafik, men tekst skal der til.

Da dette er et programmeringsprojekt, og ikke et fiktions-skrivningsprojekt, så skal I til at starte med begrænse omfanget af jeres spil. I skal altså nøjes med at lave 3-4 forskellige rum, spilleren kan bevæge sig mellem. Hvis I synes sådan noget er sjovt at nørkle med, kan I altid udvide spillet i jeres fritid, efter I har afleveret.

## 3 Opgaver

I dette afsnit finder I en række opgaver, der skal hjælpe jer i gang med at bygge jeres spil.

### Opgave 1: Kort over jeres spil

Lav et kort over de rum der skal være i jeres spil – husk at der kun skal være 3-4 rum.

I kan også diskutere hvad der skal være i rummene, men lad være med at bruge for meget tid på det nu. I kan altid ændre på teksterne i spillet senere.

### Opgave 2: Det udleverede program

Undersøg det udleverede program, `adventure-template.py`.

- Kør koden. Hvad sker der når man indtaster kommandoer? Sammenlign med flow-diagrammet fra figur 1.
- Lige nu indeholder spillet kun et enkelt, meget kedeligt rum. Prøv at lave om på hvilken tekst der bliver vist for dette rum.
- Prøv at sætte `current_room`-variablen til 1 i stedet for 0. Hvad sker der når i så kører programmet? Hvorfor?

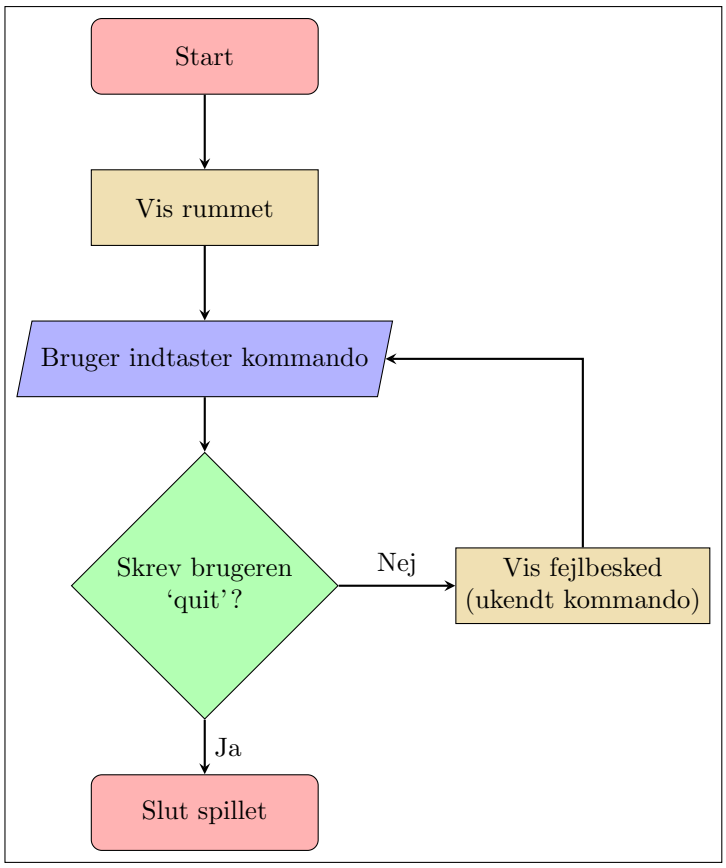
  (Husk at sætte den tilbage til 0 bagefter)

### Opgave 3: At kigge sig omkring

Stort set alle tekst-adventures har en kommando der lader spilleren kigge sig omkring i det rum de står i (ofte `look` eller `l`). Sådan en skal jeres spil også have. Den skal tilføjes til if-sætningen i `game_loop`-funktionen.

### Opgave 4: Flere rum

Tilføj en funktion for hvert rum på jeres kort (undtagen det rum der skal være rum 0). Tilføj flere forgreninger til `show_room`-funktionen, og test at programmet kan vise de nye rum (ved at ændre på `current_room`-variablen i `game_loop`-funktionen).




Figur 1: Flow-diagram over den udleverede kode

### Opgave 5: Forbindelser mellem rum

Spilleren skal også kunne bevæge sig rundt mellem rum. Det kræver et par forskellige udvidelser, før det virker.

- Start med at finde ud af hvordan kommandoerne for at bevæge sig skal se ud. Hvis spilleren vil tage den nordlige udgang, skal de så indtaste `north`, `n`, `move north`, eller noget helt fjerde?
- Tilføj de relevante kommander til `game_loop`-funktionen. Til at starte med behøver de ikke gøre noget – de kan bare printe en besked om at kommandoen ikke er lavet færdig endnu.
- Tilføj kode til de nye kommandoer der – baseret på hvilket rum spilleren er i, og hvilken retning de gerne vil gå – tjekker om det er muligt, og i så fald ændrer `current_room` til at være nummeret på det nye rum.

  Man kan med fordel lave en struktur der lidt ligner den for `show_room`, hvor man har funktioner for hvert rum spilleren kan bevæge sig væk fra: `move_from_room_0`, `move_from_room_1`, og så videre. Hver af disse funktioner tager retningen som input, og bruger en if-sætning til at tjekke hvilket rum man ender i. F.eks. kan det være at hvis man går nordpå fra rum 0, så ender man i rum 1. Dette kan i koden se sådan ud:

```
def move_from_room_0(direction):
  """Room 0 only has a single exit, which leads
  north to room 1."""
  if direction == "north":
    return 1
  else:
    return 0
```

  Hvis man forsøger at bruge en udgang der ikke findes – altså alt andet end nord – så forbliver man bare i samme rum.

## 4 Valgfri opgave

Skulle I gå hen og blive færdige med projektet med god tid til overs, så er her et forslag til en udvidelse.

### Opgave 6 (valgfri): NPC-dialog

Tilføj en *non-player character*, eller NPC, til jeres spil. Der skal være en kommando der lader spilleren snakke med NPC'en, f.eks. `talk [npc'ens navn]`. Når man snakker med NPC'en, kommer man ind i en ny løkke, hvor NPC'en siger noget, og man derefter får lov at vælge hvad man vil svare. Det kunne f.eks. se sådan her ud fra spillerens synspunkt:

```
You are in an empty room with white walls, and a
single door leading north.
By the door stands a man dressed in a blue uniform.
A name tag on his chest reads 'Jeff'.
```

```
> talk jeff
'Hello,' says Jeff. 'How may I help you?'
1. Where am I?
2. Who are you?
3. What's on the other side of the door?
4. [Leave conversation]
> 2
'I am an example NPC,' says Jeff. 'I am here to answer
your questions.' His expression does not change at
all as he speaks.
1. Where am I?
2. Who are you?
3. What's on the other side of the door?
4. [Leave conversation]
> 5
There is no option numbered 5.
> 4
You say your goodbyes to Jeff.
```

Det er selvfølgelig op til jer at bestemme hvad det skal være for en NPC, og hvilke muligheder spilleren skal have i samtalen.