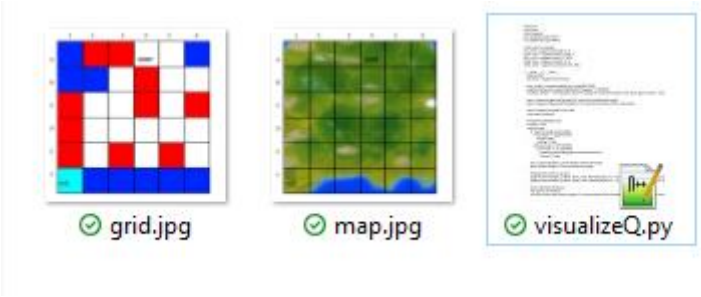
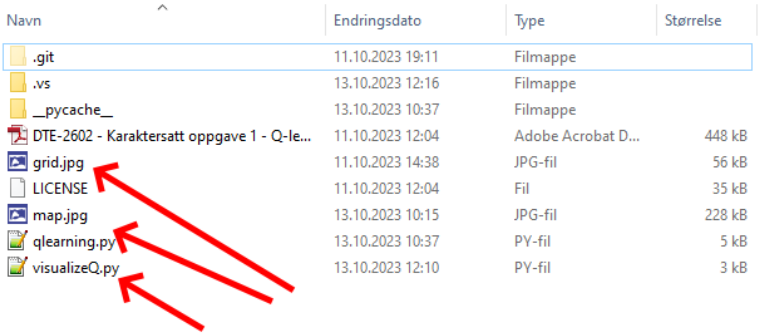
Oppsett av visualiseringskode

I tillegg til dette dokumentet har du fått tre filer: to bakgrunnsbilder og en kodefil:



1. Denne visualiseringen er basert på **pygame**. Installer **pygame** i miljøet til oppgaven. Anaconda: https://anaconda.org/conda-forge/pygame
2. Kopier bildene og visualizeQ.py til oppgavemappen din. Husk å bruke *«git add -A»* slik at de nye filene blir registrert i versjonskontrollen!



3. For at visualiseringen skal fungere må du gjøre noen tilpasninger i Robot-klassen i qlearning.py. Pygame må vite hvor på skjermen den skal tegne agenten, dette løser vi ved å legge til funksjonene get_x() og get_y() som returnerer **kolonnen** og **raden** som roboten står i:

```python
class Robot:

    def __init__(self):
        pass

    def get_x(self):
        pass # Return the current column of the robot (0-5).

    def get_y(self):
        pass # Return the current row of the robot (0-5).
```

Lag en metode som heter «one_step_q_learning()» der du gjør én iterasjon av Q-learning-algoritmen (uten while!):

```python
def one_step_q_learning(self):
    # Get action based on policy.
    # Get the next state based on the action.
    # Get the reward for going to this state.
    # Update the Q-matrix.
    # Go to the next state.
    pass
```

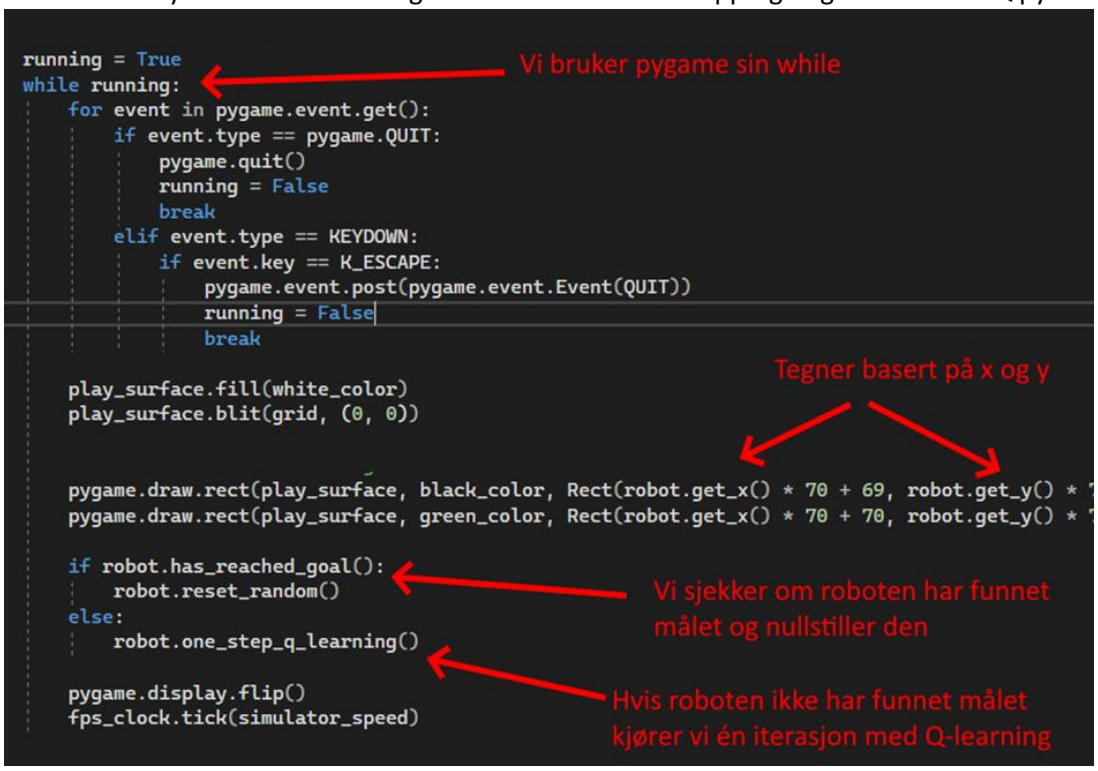
Lag en metode «has_reached_goal()» der du sjekker om roboten står i måltilstanden:

```
def has_reached_goal(self):
    """True if the robot is in the goal state."""
    pass
```

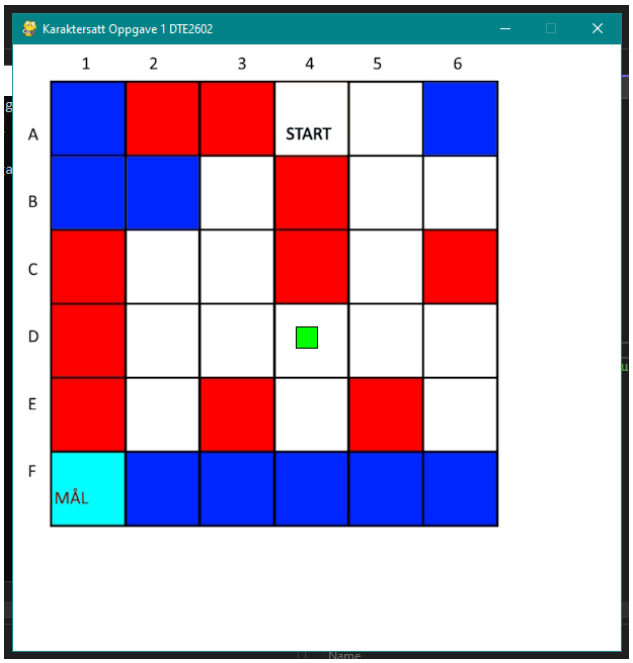
Og til slutt en metode «reset_random()» der du plasserer roboten i en tilfeldig valgt tilstand:

```
def reset_random0(self):
    """Places the robot in a random state."""
    pass
```

Disse metodene er allerede i bruk i visualizeQ.py, og må være på plass før du kan kjøre visualiseringen.

4. Pygame bruker en while True-løkke som kontinuerlig tegner opp vinduet så lenge programmet kjører. Vi kan bruke denne for å velge handling og flytte roboten fra tilstand til tilstand helt til den finner målet, i stedet for while-løkken som brukes i vanlig Q-learning. På denne måten synkroniserer vi bevegelsene til roboten med opptegningen. I visualizeQ.py:
5. 

6. Andre relevante ting er beskrevet med kommentarer i visualizeQ.py. Hvis alt fungerer skal du få opp et vindu når du kjører filen visualizeQ.py der den grønne firkanten er selve roboten:



Til slutt: Visualiseringen er frivillig og ikke en del av oppgaven, og teller derfor ikke mot karakter. Prøv gjerne ut forskjellige ting i visualiseringen og ha det gøy, men ikke gå deg fast og bruk for mye tid på dette!