class Celle:

# Konstruktør har tre instansvariablene

    def \_\_init\_\_(self):

        self.\_status = "doed"

        self.\_naboer = []

        self.\_ant\_levende\_naboer = 0

#   setter statusen dode

    def sett\_doed(self):

        self.\_status = "doed"

#   setter statusen levende

    def sett\_levende(self):

        self.\_status = "levende"

#   appender nabo av celle til i listen \_naboer

    def legg\_til\_nabo(self, nabo):

        self.\_naboer.append(nabo)

        return self.\_naboer

#   Dersom status er levende, returnerer True, ellers False

    def er\_levende(self):

        if self.\_status == "levende":

            return True

        else:

            return False

    def hent\_status(self):

        return self.\_status

#   returnerer "0", dersom status er levende, ellers "."

    def hent\_status\_tegn(self):

        if self.\_status == "levende":

            return "O"

        else:

            return "."

#   teller antall levende naboer. Det må være "0" hver gang når det starter

    def tell\_levende\_naboer(self):

        self.\_ant\_levende\_naboer = 0

        for i in self.\_naboer:

            if i.er\_levende() == True:

                self.\_ant\_levende\_naboer +=1

        return self.\_ant\_levende\_naboer

    def oppdater\_status(self):

        if self.\_status == 'doed':

            if self.\_ant\_levende\_naboer ==3:

                self.sett\_levende()

        else:

            if self.\_ant\_levende\_naboer < 2 or self.\_ant\_levende\_naboer > 3:

                self.sett\_doed()

            else:

                self.sett\_levende()

konstruktøren tar to argument som oppgir rader og kolonner

"""

class Rutenett:

    def \_\_init\_\_(self, rader, kolonner):

        self.\_ant\_rader = rader

        self.\_ant\_kolonner = kolonner

        self.\_rutenett = self.\_lag\_tomt\_rutenett()

# oppretter rader, og appender like mange None-verdier som kolonner

    def \_lag\_tom\_rad(self):

        tom\_rad = []

        for i in range(self.\_ant\_kolonner):

            tom\_rad.append(None)

        return tom\_rad

#   oppretter ytre liste, så vi lager selv rutenett

    def \_lag\_tomt\_rutenett(self):

        ytre\_liste = []

        for i in range(self.\_ant\_rader):

            ytre\_liste.append(self.\_lag\_tom\_rad())

        return ytre\_liste

#  et tilfeldig antall celler får status levende i denne metoden

    def fyll\_med\_tilfeldige\_celler(self):

        for rad in range(self.\_ant\_rader):

            for kol in range(self.\_ant\_kolonner):

                self.lag\_celle(rad, kol)

# oppretter celle ved å bruke Celle-objekt

# hver celle har 1/3 sjanse for å være levende ved bruk av randint(0,2)

    def lag\_celle(self, rad, kol):

        celle = Celle()

        tilfeldig = randint(0,2)

        if tilfeldig == 0:

            celle.sett\_levende()

        self.\_rutenett[rad][kol] = celle

#   sjekker om celle som har en bestemt posisjon, ligger i rutenettet

#   hvis det er ikke i rutenett, returnerer None

    def hent\_celle(self, rad, kol):

        if rad not in range(self.\_ant\_rader):

            return None

        elif kol not in range(self.\_ant\_kolonner):

            return None

        else:

            return self.\_rutenett[rad][kol]

#  hvert element i rutenett skrives ut ved hjelp av å kalle metoden hent\_status\_tegn

    def tegn\_rutenett(self):

        for rad in self.\_rutenett:

            for kol in rad:

                print(kol.hent\_status\_tegn(), end="")   # unngår linjeskift med hjelp av end=""

            print("")   # hoppe ny linje

#   henter en celle som har gitt koordinater

#   appender hver nabo av den gitte cellen fordi det trenges å vite statusen ('levende' og 'død') til hver nabo

    def \_sett\_naboer(self, rad, kol):

        celle = self.hent\_celle(rad, kol)

        if self.hent\_celle(rad -1, kol - 1) is not None:

            celle.legg\_til\_nabo(self.hent\_celle(rad -1, kol - 1))

        if self.hent\_celle(rad - 1, kol) is not None:

            celle.legg\_til\_nabo(self.hent\_celle(rad - 1, kol))

        if self.hent\_celle(rad -1, kol +1) is not None:

            celle.legg\_til\_nabo(self.hent\_celle(rad -1, kol +1))

        if self.hent\_celle(rad , kol -1) is not None:

            celle.legg\_til\_nabo(self.hent\_celle(rad , kol -1))

        if self.hent\_celle(rad , kol +1) is not None:

            celle.legg\_til\_nabo(self.hent\_celle(rad , kol +1))

        if self.hent\_celle(rad +1, kol -1) is not None:

            celle.legg\_til\_nabo(self.hent\_celle(rad +1, kol -1))

        if self.hent\_celle(rad +1, kol) is not None:

            celle.legg\_til\_nabo(self.hent\_celle(rad +1, kol))

        if self.hent\_celle(rad +1, kol +1) is not None:

            celle.legg\_til\_nabo(self.hent\_celle(rad +1, kol +1))

#  kaller hver enkelt celle, og sender sett.\_naboer for å koble sammen

    def koble\_celler(self):

        for rad in range(self.\_ant\_rader):

            for kol in range(self.\_ant\_kolonner):

                self.\_sett\_naboer(rad,kol)

#  appender til en enkel liste hver celle fra rutenett

    def hent\_alle\_celler(self):

        celle\_liste = []

        for rad in self.\_rutenett:

            for kol in rad:

                celle\_liste.append(kol)

        return celle\_liste

#     returnerer antall levende celler ved å sjekke om celle er levende

    def antall\_levende(self):

        antall\_levende = 0

        for rad in self.\_rutenett:

            for kol in rad:

                if kol.er\_levende():

                    antall\_levende += 1

        return antall\_levende

class Verden:

    def \_\_init\_\_(self, rader, kolonner):

        self.\_generasjonsnummer = 0

        self.\_rutenett = Rutenett(rader, kolonner)

        self.\_rutenett.fyll\_med\_tilfeldige\_celler()

        self.\_rutenett.koble\_celler()

# tegner rutenett, og skriver ut generasjonsnummeret og antall levende celler

    def tegn(self):

        self.\_rutenett.tegn\_rutenett()

        print("Antall levende celler: ", self.\_rutenett.antall\_levende())

        print("Generasjonsnummer: ",self.\_generasjonsnummer)

#   henter alle celler i rutenett, og teller levende naboer

#   oppdaterer status på hver celle når alle celler  i rutenett kommer på nytt

    def oppdatering(self):

        alle\_celler = self.\_rutenett.hent\_alle\_celler()

        for celle in alle\_celler:

            celle.tell\_levende\_naboer()

            celle.oppdater\_status()

        self.\_generasjonsnummer += 1