- ファイル名のXXXXXXXXのところは学籍番号に変更してください
- プログラム中にstudent_numberに自分の学籍番号を入れてください

```
student_number = XXXXXXXX
```

総合演習: マインスイーパー

Docstringの指示に従って、`initilize_list()`、`open_grid()`、`toggle_flag()`、`check_all_opened()`の関数を完成させよ

`draw_ui()`と`if __name__ == '__main__':`内は変更する必要はない

全ての関数を完成させて本プログラムを実行すると、マインスイーパーの画面と共にユーザーの入力を待機する。以下はその入力例で、(x,y)=(1,2)の座標のマスを開く動作を指示している。コマンドはマスを開く"open"と旗を立てる"flag"の2種類だけ用意されている。

```
> Command ('open' or 'flag'):open
> x:1
> y:2
```

遊び方は以下の通り。この仕様は課題と直接は関係ないが、以下の通りに動かない場合は関数に何らかの問題がある可能性が高い

- ユーザーはコマンドと座標を指定して入力し、それに応じてマインスイーパーの画面は更新され、再びユーザーの入力を待機する
- "open"では指定した座標のマスを開き、'flag'では旗を立てることができる
- ユーザーの入力が想定外の文字列あるいは型だった場合は警告を出して再度ユーザーの入力を待機する
- "open"と"flag"共通の仕様として、フィールドの外側の座標を指定した場合は無視して再びユーザーの入力を待機する
- "open"の仕様として、すでに開いたマスや旗が立っているマスを開こうとしても無視して再びユーザーの入力を待機する
- 地雷を踏んでしまった場合は"Game over"と表示されてプログラムが終了する
- 地雷以外の全てのマスを開いた場合は"You win!"と表示されてプログラムが終了する

採点では実装した4つの関数についてそれぞれテストを行い、マインスイーパー全体が遊べるようになっているかどうかは問わないものとする。また、`draw_ui()`や`if __name__ == '__main__':`内については自由に変更しても構わないが、本プログラムがエラーにならないような状態で提出するように気をつけること

また本プログラムはSpyderでの動作を想定している。もしコマンドラインでの実行を行いたい場合は`if __name__ == '__main__':`内にある`is_ascii_plot`の値を`True`にすることでグラフをプロットするのではなくASCII表示に切り替えることができる

```
import random
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.patches import RegularPolygon

def initialize_list(width, height, mine_num, mine_list, open_list, number_list, fla
  """マインスイーパー用の4つのリストの初期化

  ゲームに使うリストは以下のような仕様となっている
  - mine_list, open_list, number_list、flag_listはwidth * heightの長さのリストである
    - これ以降、特に指定がない限り(x,y)座標の要素は<リスト名>[width * y + x]で示されるものとす
    - ちなみにxは0以上width未満。yは0以上height未満の整数である
  - mine_listは地雷の場所を示す地図であり、bool型を要素とするリストである。地雷があるところがTru
  - open_listはマス目を開いているかどうかを示す地図であり、bool型を要素とするリストである。開いて
  - number_listはマスごとの周囲8マスにある地雷の数を表す地図であり、int型を要素とするリストであ
  - flag_listはプレイヤーが立てる「地雷があるかどうかの旗」を示す地図であり、bool型を要素とするリ

  以下の手順でこれら4つのリストを初期化する
  - mine_list: 全てFalseで埋めたwidth * heightの長さを持つリストを作成。その後、ランダムにmi
  - open_list: まだマスは1つも開いていないので、全てFalseで埋めたwidth * heightの長さを持つリ
  - number_list: mineListを参照しながら、周囲8マスにある地雷の数(int型)でwidth * heightの長
  - flag_list: まだ旗は1つも立っていないので、全てFalseで埋めたwidth * heightの長さを持つリス

  Attributes:
    width: int ゲーム盤の横幅。int以外である場合や0以下である場合は考慮しなくて良い
    height: int ゲーム盤の縦幅。int以外である場合や0以下である場合は考慮しなくて良い
    mine_num: int 地雷の数。int以外である場合や0以下である場合は考慮しなくて良い
    mine_list: list[bool] 地雷の地図。引数には空のリスト(つまり[])が渡される前提とする
    open_list: list[list[bool]] 開いているマスの地図。引数には空のリスト(つまり[])が渡される
    number_list: list[list[number]] 周囲8マスの地雷の数の地図。引数には空のリスト(つまり[])
    flag_list: list[list[bool]] 旗が立っているかマスの地図。引数には空のリスト(つまり[])が渡
  Returns:
    なし
  """

  # ヒント: 0以上n未満のランダムな整数はrandom.randrange(0, n, 1)で取得できる
  pass

def open_grid(x, y, width, height, mine_list, open_list, number_list, flag_list):
  """指定した(x,y)座標のマスを開く

  - xが0以上width未満でない、もしくはyが0以上height未満でないなら何もせずにNoneを返す
  - 指定された(x,y)座標に旗が立っていたら何もせずにNoneを返す
  - 指定された(x,y)座標のマスが開いていたら何もせずにNoneを返す
  - それら以外の場合であれば、指定された(x,y)座標のマスを開く。さらに周囲8マスの地雷の数が0であれ
  - 最後に、(x,y)座標のマスが地雷であればTrue、そうでなければFalseを返す

  Attributes
    x: int 開くマス目のx座標
    y: int 開くマス目のy座標
    width: int ゲーム盤の横幅
    height: int ゲーム盤の縦幅
    mine_list: list[bool] 地雷の地図
    open_list: list[list[bool]] 開いているマスの地図
    number_list: list[list[number]] 周囲8マスの地雷の数の地図
```

```
    flag_list: list[list[bool]] 旗が立っているかマスの地図
  Returns
    bool or None: 開いたマスが地雷であればTrue、地雷でなければFalse、x,yいずれかが無効な座標て
  """
  pass

def toggle_flag(x, y, width, height, open_list, flag_list):
  """指定した(x,y)座標の旗をトグルする

  - xが0以上width未満でない、もしくはyが0以上height未満でないなら何もせずにNoneを返す
  - (x,y)座標がすでに開いているマスであれば何もせずにNoneを返す
  - 指定した(x,y)座標に旗がなかった場合は旗を立て、旗が立っていれば旗を下ろす
  - 旗を立てた場合はTrueを返し、下ろした場合はFalseを返す

  Attributes
    x: int 開くマス目のx座標
    y: int 開くマス目のy座標
    width: int ゲーム盤の横幅
    height: int ゲーム盤の縦幅
    open_list: list[list[bool]] 開いているマスの地図
    flag_list: list[list[bool]] 旗が立っているかマスの地図
  Returns
    bool or None: 旗を立てた場合はTrue、下ろした場合はFalse、x,yいずれかが無効な座標あるいはす
  """
  pass

def check_all_opened(width, height, mine_list, open_list):
  """全ての地雷以外のマスが開かれたかの判定

  Attributes
    width: int ゲーム盤の横幅
    height: int ゲーム盤の縦幅
    mine_list: list[bool] 地雷の地図
    open_list: list[list[bool]] 開いているマスの地図
  Returns
    bool: 地雷以外のマスがすべて開かれていたらTrue、そうでないならFalseを返す
  """
  pass

def draw_ui(width, height, mine_num, mine_list, open_list, number_list, flag_list,
  """マインスイーパーのフィールドを描画

  こちらで用意したもので、特に実装する必要はない。変更しても構わないが文法エラーにならないよう注意す

  Attributes:
    width: int ゲーム盤の横幅
    height: int ゲーム盤の縦幅
    mine_num: int 地雷の数
    mine_list: list[bool] 地雷の地図
    open_list: list[list[bool]] 開いているマスの地図
    number_list: list[list[number]] 周囲8マスの地雷の数の地図
    flag_list: list[list[bool]] 旗が立っているかマスの地図
    is_ascii_plot: bool ASCII文字列でプロットするかどうか
    is_gameover: bool 地雷を踏んでゲームに失敗しているかどうか
  Returns:
    な
```

```
  なし
"""
if is_ascii_plot:
  for y in reversed(range(height)):
    print("  ", end="")
    for x in range(width):
      print("--", end="")
    print("-")
    print(f"{y} ", end="")
    for x in range(width):
      if open_list[y * width + x] == True:
        if mine_list[y * width + x] == True:
          print(f"|*", end="")
        else:
          print(f"|{number_list[y * width + x]}", end="")
      else:
        if not is_gameover:
          if flag_list[y * width + x]== True:
            print(f"|f", end="")
          else:
            print(f"| ", end="")
        else:
          if mine_list[y * width + x] == True:
            print(f"|*", end="")
          elif flag_list[y * width + x]== True:
            print(f"|f", end="")
          else:
            print(f"| ", end="")

    print("|")
  print("  ", end="")
  for x in range(width):
    print("--", end="")
  print('-')
  print("  ", end="")
  for x in range(width):
    print(f" {x}", end="")
  print()
else:
  offset = 0.5
  fig = plt.figure(figsize=((width + 2) / 3.0, (height + 2) / 3.0))
  ax = fig.add_axes((0.05 + offset, 0.05 + offset, 0.9 , 0.9),
                                  aspect='equal', frameon=False,
                                  xlim=(-0.05 - offset, width + 0.05 - offset),
                                  ylim=(-0.05 - offset, height + 0.05 - offset))
  ax.xaxis.set_ticks(range(width))
  ax.yaxis.set_ticks(range(height))
  ax.set_xlabel("x")
  ax.set_ylabel("y")
  covered_color = '#DDDDDD'
  uncovered_color = '#AAAAAA'
  edge_color = '#888888'
  squares = np.array([[RegularPolygon((i + 0.5 - offset, j + 0.5 - offset),
    numVertices=4,
    radius=0.5 * np.sqrt(2),
    orientation=np.pi / 4
```

```
      orientation=np.pi / 4,
      ec=edge_color,
      fc=covered_color)
        for j in range(height)]
          for i in range(width)])
    [ax.add_patch(sq) for sq in squares.flat]

    number_colors = ['none', 'blue', 'green', 'red', 'darkblue', 'darkred', 'darkgr
    for y in range(height):
      for x in range(width):
        if open_list[y * width + x] == True:
          ax.text(x + 0.5 - offset, y + 0.5 - offset, str(number_list[y * width + x
                  ha='center', va='center', fontsize=18, fontweight='bold')
          squares[x, y].set_facecolor('#AAAAAA')
          if mine_list[y * width + x] == True:
              ax.add_patch(plt.Circle((x + 0.5 - offset, y + 0.5 - offset), radius=
              ax.text(x + 0.5 - offset, y + 0.5 - offset, 'X', color='r', fontsize=
        else:
          if is_gameover:
            if mine_list[y * width + x] == True:
              ax.add_patch(plt.Circle((x + 0.5 - offset, y + 0.5 - offset), radius=
              ax.text(x + 0.5 - offset, y + 0.5 - offset, 'X', color='r', fontsize=
        if flag_list[y * width + x] == True:
          flag_vertices = np.array([[0.25, 0.2], [0.25, 0.8], [0.75, 0.65], [0.25,
          flag = plt.Polygon(flag_vertices + [x - offset, y - offset], fc='red', ec
          ax.add_patch(flag)
    plt.show()

# 以降、ゲームの実行手順。変更しても構わないが、文法エラーのまま提出しないように注意すること
if __name__ == '__main__':
  is_ascii_plot = False # Spyder以外の環境で実行する場合はこれをTrueにして実行する。ASCIIで
  width = 4 # フィールドの横幅を変えたい場合はこの数字を変更する
  height = 4 # フィールドの縦幅を変えたい場合はこの数字を変更する
  mine_num = 2 # 地雷の数を変えたい場合はこの数字を変更する
  mine_list = []
  open_list = []
  number_list = []
  flag_list = []

  initialize_list(width, height, mine_num, mine_list, open_list, number_list, flag_
  draw_ui(width, height, mine_num, mine_list, open_list, number_list, flag_list, is
  while True:
    command_type = input("Command ('open' or 'flag'):")
    if command_type != 'open' and command_type != 'flag':
      print("Invalid command type")
      continue

    try:
      x = int(input('x:'))
      y = int(input('y:'))
    except ValueError as e:
      print("Invalid value for (x,y)")
      continue
    if command_type == 'open':
      is_mine = open_grid(x, y, width, height, mine_list, open_list, number_list, f
      is_all_opened = check_all_opened(width, height, mine_list, open_list)
```

```
    is_all_opened = check_all_opened(width, height, mine_list, open_list)
    draw_ui(width, height, mine_num, mine_list, open_list, number_list, flag_list
    if is_mine == True:
      print("Game over")
      break
    if is_all_opened == True:
      print("You win!")
      break
  elif command_type == 'flag':
    toggle_flag(x, y, width, height, open_list, flag_list)
    draw_ui(width, height, mine_num, mine_list, open_list, number_list, flag_list
```