به نام خدا


استاد: دکتر حسین کارشناس

دستیاران آموزشی:

مهدی مهدیه  یونس ایوبی‌راد

دانیال شفیعی  پویا اسفندانی


**مبانی هوش مصنوعی و کاربردها**

**نیم‌سال اول ۱۴۰۴-۱۴۰۵**

**ددلاین: ۱۲ دی**


دانشکده مهندسی کامپیوتر

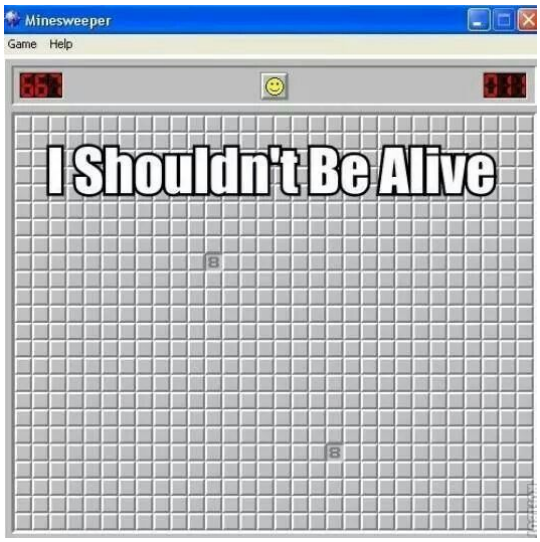

---

**فاز چهارم پروژه - استنتاج به کمک منطق درجه اول**

---

مرحله چهارم پروژه درس از یک بخش تشکیل شده است. برای پروژه خود بعد از پیاده‌سازی گزارش تهیه کرده و در آن پیاده‌سازی خود را توضیح داده و نتایج را نمایش دهید. فایل های محیط در گیت‌هاب در اختیار شما قرار داده خواهد شد.

# نوستالژی مین‌ها و پرچم‌ها




**مین‌روب** یا **MineSweeper** ، یک بازی قدیمی ویندوزی که بعضی‌ها را همزمان عصبانی و معتاد کرد! شاید شما هم خاطره‌ای از لرزیدن دستتان هنگام کلیک روی مربع‌های خاکستری را به یاد داشته باشید. اما اینبار نه با حدس و خطا، بلکه با منطق و استدلال قرار است این بازی را حل کنیم. هدف این پروژه این است که نشان دهیم چگونه می‌توان قوانین ساده یک بازی را به صورت حقایق و قواعد منطقی بیان کرد و سپس با یک موتور استنتاج، راه‌حل را پیدا نمود.

## انجام بازی مین‌روب

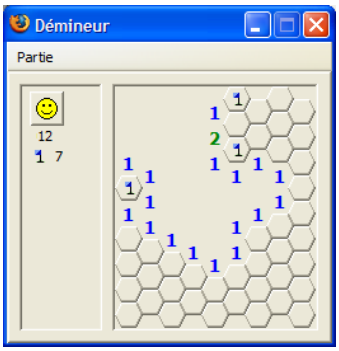
در بازی مین‌روب دو کنش وجود دارد:

- Reveal: با کلیک چپ روی هر مربع آن خانه فاش می‌شود. اگر مین (Mine) باشد شما می‌بازید! در غیر این صورت اگر خانه امن باشد، یک سرنخ (Clue) افشا می‌شود. هر سر نخ شامل یک عدد بین ۰ تا ۸ است و بیانگر این است که در همسایگی این خانه چند مین وجود دارد.

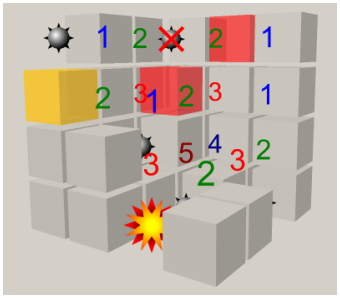
- Flag: با کلیک راست روی هر مربع آن خانه به عنوان مین شناخته می‌شود. می‌توان خانه‌ای که مین نیست را پرچم‌گذاری کرد اما این باعث تصمیم‌گیری اشتباه در آینده می‌شود که ممکن است برایتان گران تمام شود.

شرط برد وضعیتی است که تمام خانه‌های مین پرچم‌گذاری شده و تمام خانه‌های امن فاش شوند. می‌توانید در این لینک روند حل یک نمونه از این بازی را مشاهده کنید:

https://en.wikipedia.org/wiki/Minesweeper_(video_game)#Objective_and_strategy




(آ) مین‌روب شش‌ضلعی



(ب) مین‌روب سه‌بعدی

(ج) مین‌روب با چند مین

شکل ۱: نسخه‌های دیگری از مین‌روب

# ۱ منطق درجه اول

منطق درجه اول یا First-Order Logic (FOL) زبانی است که به ما اجازه می‌دهد جهان را نه با متغیرهای ساده، بلکه با «اشیاء» و «روابط» بین آن‌ها توصیف کنیم. در پایتون، کتابخانه PyDatalog این امکان را فراهم می‌کند که مستقیماً از پارادایم برنامه‌نویسی منطقی (شبیه به Prolog) استفاده کنیم.

## ۱.۱ نصب و راه‌اندازی

برای شروع، ابتدا باید کتابخانه را نصب کنید:

```
pip install pyDatalog
```

## ۲.۱ مفاهیم پایه: ترم، فکت، قانون و کوئری

برای استفاده از PyDatalog، باید چهار مفهوم کلیدی را درک کنید:

- **ترم** (Term): متغیرها یا اشیاء (مثل $X$، $Y$ یا اعداد یا مسندها).
- **فکت** (Fact): حقایقی که در مورد جهان می‌دانیم (داده‌های اولیه).
- **قانون** (Rule): روابط منطقی که دانش جدید تولید می‌کنند (اگر ... آنگاه ...).
- **کوئری** (Query): پرسش از سیستم برای یافتن جواب.

### ۱.۲.۱ یک مثال ساده

فرض کنید می‌خواهیم رابطه پدری و پدربزرگی را تعریف کنیم:

```python
from pyDatalog import pyDatalog

# 1. Define Terms (Variables)
pyDatalog.create_terms('X, Y, Z, Father, GrandFather')

# 2. Add Facts (Father(Parent, Child))
+Father('Ali', 'Reza')    # Ali is father of Reza
```

```
+Father('Reza', 'Sara')   # Reza is father of Sara

# 3. Define Rules (Logic)
# Logic: X is GrandFather of Y IF X is Father of Z AND Z is Father of Y
GrandFather(X, Y) <= Father(X, Z) & Father(Z, Y)

# 4. Query
print(list(GrandFather(X, 'Sara')))
# Output: [('Ali',)]
```

در مثال بالا، عملگر <= به معنی «اگر» و عملگر & به معنی «و» منطقی است. این قانون معادل عبارت زیر است:

$$\forall X \forall Y \, (\exists Z \, (\text{Father}(X, Z) \wedge \text{Father}(Z, Y)) \rightarrow \text{GrandFather}(X, Y))$$

### ۲.۲.۱ یادگیری سینتکس

کتابخانه‌ی PyDatalog ساده‌ترین سینتکس را بین موتورهای استنتاج FOL دارد.

- آموزش خلاصه‌ی این کتابخانه به همراه مثال:

https://sites.google.com/site/pydatalog/Online-datalog-tutorial

- در لینک زیر یک مثال کاربردی از این کتابخانه آورده شده است:

https://blog.vishok.me/blog/app/pydatalog

- برای یادگیری کامل سینتکس و قابلیت‌های پیشرفته، آموزش موجود در لینک زیر را مطالعه کنید:

https://sites.google.com/site/pydatalog/home

- حل چند مثال دیگر با این کتابخانه:

https://github.com/pcarbonn/pyDatalog/blob/master/pyDatalog/examples

## ۳.۱ موتورهای استنتاج دیگر

موتورهای دیگری هم وجود دارند که می‌توانند حقایق را دریافت کنند و بر اساس منطق مرتبه اول مسائل را حل کنند. یکی از مشهورترین این موتورها SWI-Prolog است. می‌توانید کد خود را به این زبان بنویسید.

```
% Define facts
father(ali, reza).
father(reza, sara).

% Define rule: X is grandfather of Y if X is father of Z and Z is father of Y
grandfather(X, Y) :- father(X, Z), father(Z, Y).

% Query examples
?- grandfather(X, sara).
% X = ali

```

سپس به کمک کتابخانه‌ی pyswip که یک wrapper برای این موتور استنتاج است از قواعد نوشته شده استفاده کنید و کوئری بزنید.

- می‌توانید موتور استنتاج را از اینجا دانلود کنید:

https://www.swi-prolog.org

- می‌توانید آموزش کار با کتابخانه pyswip را از اینجا ببینید.

https://pyswip.org/

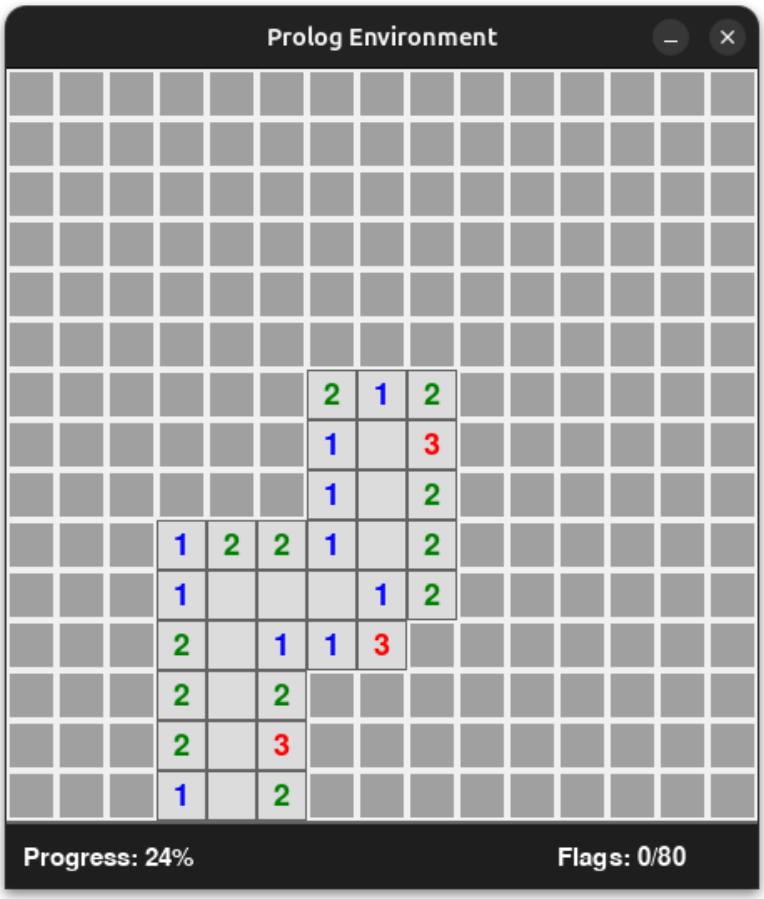
# ۲ آشنایی با محیط و توابع رابط

فایل محیط شامل کلاس MineSweeper در اختیار شما قرار گرفته است. شما نباید کدهای داخلی این کلاس را تغییر دهید، بلکه باید از توابع تعبیه شده (رابط‌ها) برای دریافت اطلاعات و ارسال دستورات استفاده کنید.

## ۱.۲ محیط

برای ساخت محیط یک نمونه از کلاس `MineSweeper` می‌سازید. این کلاس شامل ویژگی‌های زیر است:

- ابعاد نقشه شامل تعداد سطر و ستون
- تعداد مین در نقشه
- وضعیت تصادفی بازی برای بازتولید نقشه




شکل ۲: مثالی از صفحه بازی $15 \times 15$ با `mines=80` و `seed=5`

**راهنمایی:** می‌توانید با استفاده از کد `manual.py` بازی را بازی کنید. کلیک چپ معادل فاش کردن و کلیک راست معادل نشانه‌گذاری/عدم نشانه‌گذاری است.

## ۲.۲ دریافت حقایق (Facts)

در ابتدای بازی شما می‌توانید با صدا زدن `get_start_pos()` به مختصات خانه شروع برسید. تضمین می‌شود خانه‌ی شروع همیشه دارای سرنخ ۰ است (یعنی هیچ مینی اطراف آن نیست).

**نکته:** *درمورد سایر حقایق همچون همسایه بودن یا نبودن، نشانه‌گذاری شده بودن یا نبودن و فاش شده بودن یا نبودن و سرنخ خانه‌های فاش شده خودتان باید قانون بنویسید.*

## ۳.۲ کنش‌ها (Actions)

پس از اینکه موتور استنتاج شما تصمیم گرفت کدام خانه امن است یا کدام خانه مین است، باید با توابع زیر محیط را آپدیت کنید:

- `reveal(r, c)`: خانه مشخص شده را فاش می‌کند. اگر مین باشد، بازی تمام می‌شود.

  **خروجی:** اگر خروجی بمب باشد مقدار ۱- و در غیر این صورت عددی بین ۰ تا ۸ خروجی می‌دهد که نشان‌دهنده‌ی سرنخ آن خانه است. شما باید این حقیقت را به حقایق خود اضافه کنید.

- `flag(r, c)`: روی خانه مشخص شده پرچم می‌گذارد (نشانه‌گذاری مین).

- `unflag(r, c)`: از روی خانه‌ی مشخص شده پرچم را حذف می‌کند.

# ۳ مراحل پیاده‌سازی پروژه

برای انجام موفق این پروژه، پیشنهاد می‌شود روند زیر را در اسکریپت اصلی خود (main.py) طی کنید:

## ۱.۳ آماده‌سازی اولیه

کلاس محیط را Import کنید و یک نمونه از بازی بسازید. ترجیحاً از یک Seed ثابت استفاده کنید تا در حین دیباگ کردن، نقشه تغییر نکند.

```
ms = MineSweeper(rows=10, cols=10, mines=10, seed=42)
```

## ۲.۳ نوشتن قوانین و حقایق اولیه

شما باید حقایق اولیه بازی را اضافه کنید این حقایق می‌تواند تعریف سلول‌ها و وضعیت خانه شروع باشد. همچنین قوانینی چون همسایگی و امن بودن یا مین بودن را تعریف کنید.

- **قانون امنیت:** اگر یک خانه باز شده دارای عدد $N$ باشد و تعداد همسایه‌های پرچم‌گذاری شده‌ی آن برابر با $N$ باشد، تمام همسایه‌های بازنشده‌ی دیگر **امن** هستند.
- **قانون خطر:** اگر یک خانه باز شده دارای عدد $N$ باشد و تعداد کل همسایه‌های بازنشده (به علاوه پرچم‌ها) برابر با $N$ باشد، تمام همسایه‌های بازنشده **مین** هستند.

قوانین پیچیده‌تری هم وجود دارند که استخراج آن‌ها به شما واگذار شده است.

## ۳.۳ حلقه اصلی بازی (Game Loop)

شما نیاز به یک حلقه while دارید که تا زمانی که `ms.game_over` فالس است، ادامه یابد. در هر تکرار حلقه، مراحل زیر باید انجام شود:

۱. **بدست آوردن حقایق مرتبط با این دور (Query):** مثلا بدست آوردن تعداد خانه‌های Hidden یا Flagged که در اطراف خانه‌ی (R, C) هستند.

۲. **پرس‌وجو (Query):** از موتور بپرسید: «کدام خانه‌ها امن هستند؟» (Safe(R, C)) و «کدام خانه‌ها مین هستند؟» (Mine(R, C)).

۳. **اعمال تغییرات:** نتایج کوئری را با استفاده از `reveal(r, c)` و `flag(r, c)` روی محیط اعمال کنید. اگر در یک دور هیچ حرکتی پیدا نشد، عامل شما می‌تواند یک خانه را تصادفی انتخاب کند (ریسک) یا بازی را متوقف کند.

۴. **نمایش:** تابع `ms.render()` را صدا بزنید تا وضعیت را ببینید.

## ۴. نحوه ارزیابی

عامل هوشمند شما باید بتواند در ابعاد و سختی‌های مختلف بازی را حل کند. برای تحویل پروژه، کد شما روی سناریوهای دیگری تست خواهد شد. برای ارائه سعی کنید کدتان را برای این موارد بهینه کنید:

| **سطح** | **ابعاد** (Rows × Cols) | **تعداد مین** | Seed | **انتظار** |
|---|---|---|---|---|
| ساده (تست منطق) | $9 \times 9$ | 9 | 99 | حل کامل بدون باخت |
| استاندارد | $15 \times 15$ | 35 | 42 | حل کامل بدون باخت |
| چالشی | $20 \times 20$ | 75 | 123 | حداقل ٪۸۰ پیشرفت |
| بزرگ (تست سرعت) | $45 \times 80$ | 200 | $-1$ | عدم کرش کردن برنامه |

جدول ۱: سناریوهای ارزیابی پروژه

**نکته مهم:** در برخی نقشه‌ها ممکن است موقعیت‌هایی پیش بیاید که هیچ استنتاج منطقی قطعی وجود نداشته باشد (شانس ۵۰-۵۰). در این موارد، باختن عامل ایرادی ندارد، اما عامل باید تا حد ممکن با استفاده از منطق پیشروی کرده باشد.

## نکات ارائه

- لطفاً نتایج خود را در قالب یک فایل pdf به همراه کدها تحویل دهید. برای قالب گزارش می‌توانید از قالب دانشکده یا TEX استفاده کنید.
- گزارش به زبان فارسی و **خلاصه‌ای** از شرح کارهای انجام شده و نتایج حاصل شده باشد.
- در صورت پیاده‌سازی هر یک از این موارد اضافی، **تمام اعضای تیم** باید دلیل استفاده از آن و منطق استفاده از آن را بدانند. این موضوع باید هم در گزارش قید شود و هم در زمان تحویل پروژه کاملاً بر آن مسلط باشند.
- ارسال گزارش و کد در کوئرا توسط یک نفر از اعضای گروه کفایت می‌کند.

موفق باشید.

⚠ هشدار: لطفا انجام پروژه را به روز پایانی ددلاین موکول نکنید!