تمرین سری پنجم: یادگیری تقویتی

لطفاً به نکات زیر توجه کنید:

- مهلت ارسال این تمرین برای هر دو گروه **۷ دی ماه** است.
- در صورتی که به اطلاعات بیشتری نیاز دارید می‌توانید به صفحه‌ی تمرین در وب‌سایت درس مراجعه کنید.

- ما همواره هم‌فکری و هم‌کاری را برای حلِ تمرین‌ها به دانشجویان توصیه می‌کنیم. اما هر فرد باید تمامی سوالات را به تنهایی تمام کند و پاسخ ارسالی حتماً باید توسط خود دانش‌جو نوشته‌شده باشد. لطفاً اگر با کسی هم‌فکری کردید نام او را ذکر کنید.
- لطفاً برای ارسال پاسخ‌های خود از راهنمای موجود در صفحه‌ی تمرین استفاده کنید.
- هر سؤالی درباره‌ی این تمرین را می‌توانید در گروه درس مطرح کنید و یا از دستیاران حلِ تمرین بپرسید.

- آدرس صفحه‌ی تمرین: https://iust-courses.github.io/ai97/assignments/05_reinforcement_learning
- آدرس گروه درس: https://groups.google.com/forum/#!forum/ai97

موفق باشید

سؤال‌ها

در این سری از تمرین‌ها از ماژول PLE استفاده می‌کنیم. برای نصب این ماژول به سایت آن مراجعه کنید. همچنین این ماژول بر روی پایتون ورژن ۲.۷ کار می‌کند.

روش استفاده از ماژول PLE:

```
from ple import PLE
from ple.games.flappybird import FlappyBird

# Making an agent from game env
agent = FlappyBird(width=256, height=256)

# Creating a game environment
env = PLE(agent, fps=15, force_fps=False, display_screen=True)
env.init()
```

گرفتن لیست تمام اکشن‌های ممکن برای بازی:

```
actions = env.getActionSet()
```

انجام دادن یک اکشن خاص توسط عامل و خروجی آن که مقدار پاداشی است که عامل توسط انجام دادن آن عمل گرفته است:

```
env.act(action)
```

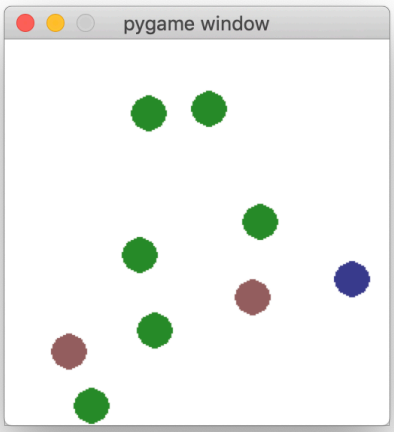
ریست کردن بازی و فهمیدن زمان تمام شدن یک بازی (بردن یا باختن):

```
if env.game_over():
        env.reset_game()
```

گرفتن موقعیت از بازی:

```
agent.getGameState()
```

سوال یک (۵۰ نمره)

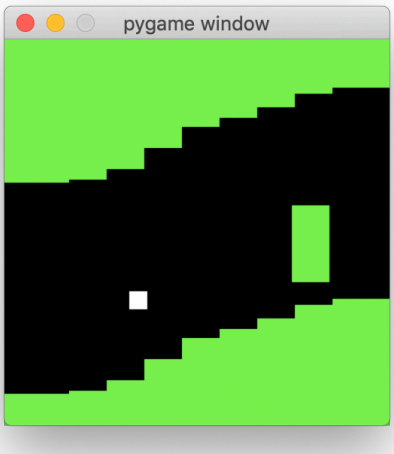


بازی waterworld:

- حالت‌ها: یک دیکشنری با کلیدهای زیر
  - `player_velocity_y`: یک float که نشان‌دهنده سرعت در راستای y است
  - `player_velocity_x`: یک float که نشان‌دهنده سرعت در راستای x است
  - `creep_dist`
    - `BAD`: لیستی از فاصله‌ی عامل تا دایره‌های قرمز
    - `GOOD`: لیستی از فاصله‌ی عامل تا دایره‌های سبز
  - `creep_pos`
    - `BAD`: لیستی از موقعیت دایره‌های قرمز
    - `GOOD`: لیستی از موقعیت دایره‌های قرمز
  - `player_x`: موقعیت عامل در راستای x
  - `player_y`: موقعیت عامل در راستای y

- اکشن‌ها: در این بازی ۵ نوع اکشن وجود دارد:
  - `115`: عمل پایین رفتن
  - `100`: عمل راست رفتن
  - `119`: عمل بالا رفتن
  - `97`: عمل چپ رفتن
  - `None`: عمل سکون

سوال دو (۵۰ نمره)



بازی pixelcopter:

- حالت‌ها: یک دیکشنری با کلیدهای زیر
  - `player_vel`: یک float که نشان دهنده سرعت عامل است
  - `player_dist_to_ceil`: یک float که نشان دهنده فاصله تا سقف است
  - `next_gate_block_top`: موقعیت بلوک بالایی بعدی
  - `next_gate_block_bottom`: موقعیت بلوک پایینی بعدی
  - `next_gate_dist_to_player`: فاصله‌ی بلوک بعدی تا عامل
  - `player_dist_to_floor`: یک float که نشان دهنده فاصله تا زمین است
  - `player_y`: یک float که نشان دهنده موقعیت عامل است

- اکشن‌ها: در این بازی ۲ نوع اکشن وجود دارد:
  - 119: عمل بالا رفتن
  - `None`: هیچ حرکتی انجام نمی‌دهد در نتیجه به خاطر جاذبه پایین می‌رود.

نحوه‌ی پاسخگویی به تمرین

همراه با صورت سوال‌ها، ۲ فایل پایتون به نام‌های WaterWorld-Agent.py و Pixelcopter.py وجود دارد که یک عامل random در آن‌ها پیاده‌سازی شده است. شما می‌بایست جواب عامل خود را در این ۲ فایل پیاده‌سازی کنید و در یک فایل فشرده‌شده به نام asg05_[Student ID] ارسال نمایید. در ابتدا شما میبایست یک تابع برای محاسبه پاداش‌ها برای هر بازی پیاده‌سازی کنید، که در ورودی یک state و یک action می‌گیرد و در خروجی میزان پاداش حاصل از آن action را می‌دهد. سپس با استفاده از الگوریتم Q-Learning یک عامل هوشمند برای هر کدام از بازی‌ها بنویسید.