مطلب فوق را به فارسی برایتان توضیح می‌دهم:

### کلاس GeneticAlgorithm:

#### ویژگی‌ها:

* population، generations، mutation\_rate: پارامترهای الگوریتم ژنتیک.
* winning\_combinations: لیست تاپل‌هایی که ترکیب‌های برنده در صفحه Tic-Tac-Toe را نشان می‌دهد.

#### متدها:

1. \_\_init\_\_(self, population=50, generations=50, mutation\_rate=0.1): الگوریتم ژنتیک را با پارامترهای پیش‌فرض یا تعیین‌شده توسط کاربر مقداردهی اولیه می‌کند.
2. initialize\_population(self) -> List[List[int]]: جمعیتی از کروموزوم‌ها تولید می‌کند، که هر کروموزوم یک لیست 9 عنصری است (که حرکات در صفحه Tic-Tac-Toe را نشان می‌دهد).
3. evaluate\_fitness(self, chromosome: List[int], current\_board: List[int]) -> int: فیتنس یک کروموزوم را با شبیه‌سازی 10 بازی ارزیابی می‌کند و تعداد برد‌ها را شمارش می‌کند.
4. play\_game(self, chromosome: List[int], current\_board: List[int], player: int) -> int: یک بازی با استفاده از یک کروموزوم شبیه‌سازی می‌شود. اگر بازیکن مشخص شده ببرد، 1 وگرنه 0 را برمی‌گرداند.
5. check\_winner(self, board: List[int], player: int) -> bool: بررسی می‌کند که آیا یک بازیکن در بازی برنده شده است یا خیر.
6. evaluate\_move(self, move: int, board: List[int], player: int) -> int: ارزیابی می‌کند که یک حرکت مطلوب است یا خیر. یک امتیاز (2 برای یک حرکت برنده، 1 برای یک حرکت مانع زنی، 0 در غیر اینصورت) برمی‌گرداند.
7. genetic\_algorithm(self, current\_board: List[int]) -> int: الگوریتم ژنتیک را اجرا می‌کند تا بهترین حرکت را براساس صفحه فعلی بازی پیدا کند.
   * یک جمعیت از کروموزوم‌ها مقداردهی اولیه می‌شود.
   * فیتنس هر کروموزوم در جمعیت ارزیابی می‌شود.
   * بهترین کروموزوم انتخاب می‌شود.
   * عملیات ترکیب و جهش برای تولید جمعیت جدید انجام می‌شود.
   * برای تعداد دفعات مشخص شده تکرار می‌شود.
   * ارزیابی مطلوبیت حرکت‌ها در صفحه فعلی انجام می‌شود.
   * بهترین حرکت بر اساس امتیازهای مطلوبیت انتخاب می‌شود.

### کلاس TicTacToeGame:

#### ویژگی‌ها:

* board: نمایش وضعیت فعلی صفحه Tic-Tac-Toe.
* genetic\_algo: نمونه از GeneticAlgorithm برای حرکات حریف.

#### متدها:

1. \_\_init\_\_(self): بازی را با یک صفحه خالی و یک الگوریتم ژنتیک پیش‌فرض مقداردهی اولیه می‌کند.
2. print\_board(self) -> None: وضعیت فعلی صفحه بازی را چاپ می‌کند که نمادهای X، O و فضاهای خالی را نشان می‌دهد.
3. is\_board\_full(self) -> bool: بررسی می‌کند که صفحه بازی پر است یا خیر (هیچ فضای خالی‌ای وجود ندارد).
4. get\_human\_move(self, name: str) -> int: از کاربر درخواست می‌کند تا حرکت خود را وارد کند و شاخص حرکت انتخابی را برمی‌گرداند.
5. play(self, with\_ai: bool = False) -> None: حلقه بازی را شروع می‌کند.
   * بین حرکات انسان و حرکات حریف تعویض می‌کند.
   * بعد از هر حرکت برنده یا مساوی بودن بازی را بررسی می‌کند.
   * پیام‌های مناسب بر اساس نتیجه نمایش داده می‌شوند.

چگونگی انجام حرکات انسان: - متد get\_human\_move از کاربر ورودی می‌گیرد و آن را اعتبارسنجی می‌کند. - سپس حرکت انتخابی به صفحه بازی اعمال می‌شود.

### چگونگی انجام حرکات حریف:

* متد genetic\_algorithm در کلاس GeneticAlgorithm فراخوانی می‌شود تا بهترین حرکت را پیدا کند.
* بهترین حرکت به صفحه بازی اعمال می‌شود.

### جریان بازی:

* بازی ادامه پیدا می‌کند تا زمانی که یک برنده وجود داشته باشد یا صفحه پر شود.
* پس از هر حرکت، پیام‌هایی برای نمایش نتیجه بازی نمایش داده می‌شوند.

### تعامل با الگوریتم ژنتیک:

* کلاس TicTacToeGame با کلاس GeneticAlgorithm تعامل می‌کند تا حرکات حریف را دریافت کند.
* الگوریتم ژنتیک از جمعیتی از کروموزوم‌ها استفاده می‌کند تا تکامل پیدا کند و بهترین حرکت را پیدا کند.

در کل، کلاس GeneticAlgorithm یک حریف مبتنی بر الگوریتم ژنتیک فراهم می‌کند و کلاس TicTacToeGame حلقه بازی را مدیریت می‌کند، حرکات انسان و حریف را انجام می‌دهد و نتیجه بازی را مشخص می‌کند.