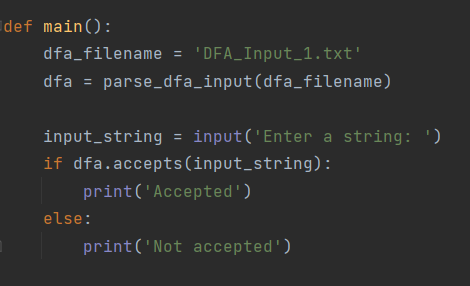
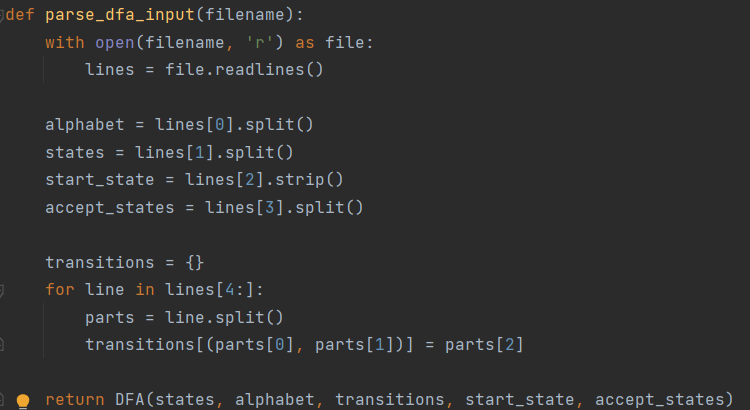
گزارش پروژه اول نظریه زبان ها و ماشین ها

بخش 1)

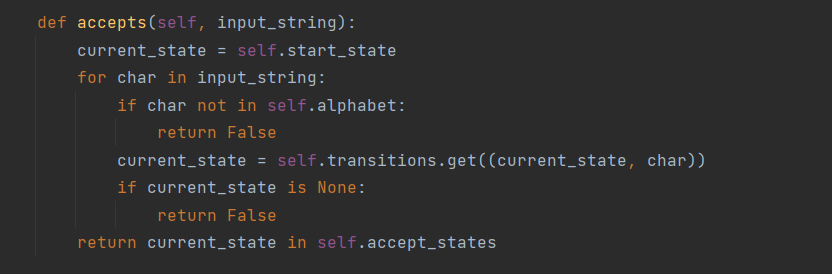
فایل ورودی را میخوانیم:

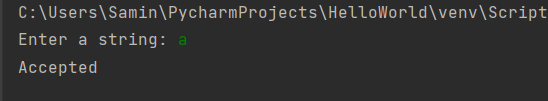


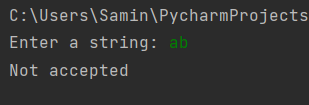


با استفاده از این فایل اطلاعات وارد میشود.

از کاربر ورودی میگیریم و با تابع زیر چک میکنیم که قابل قبول است یا خیر:

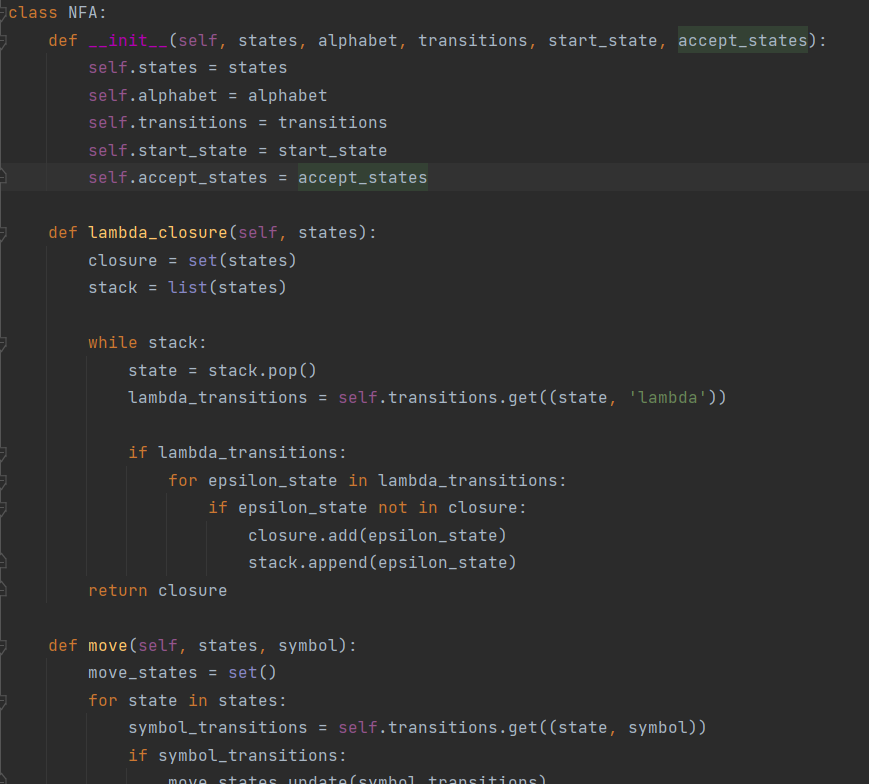




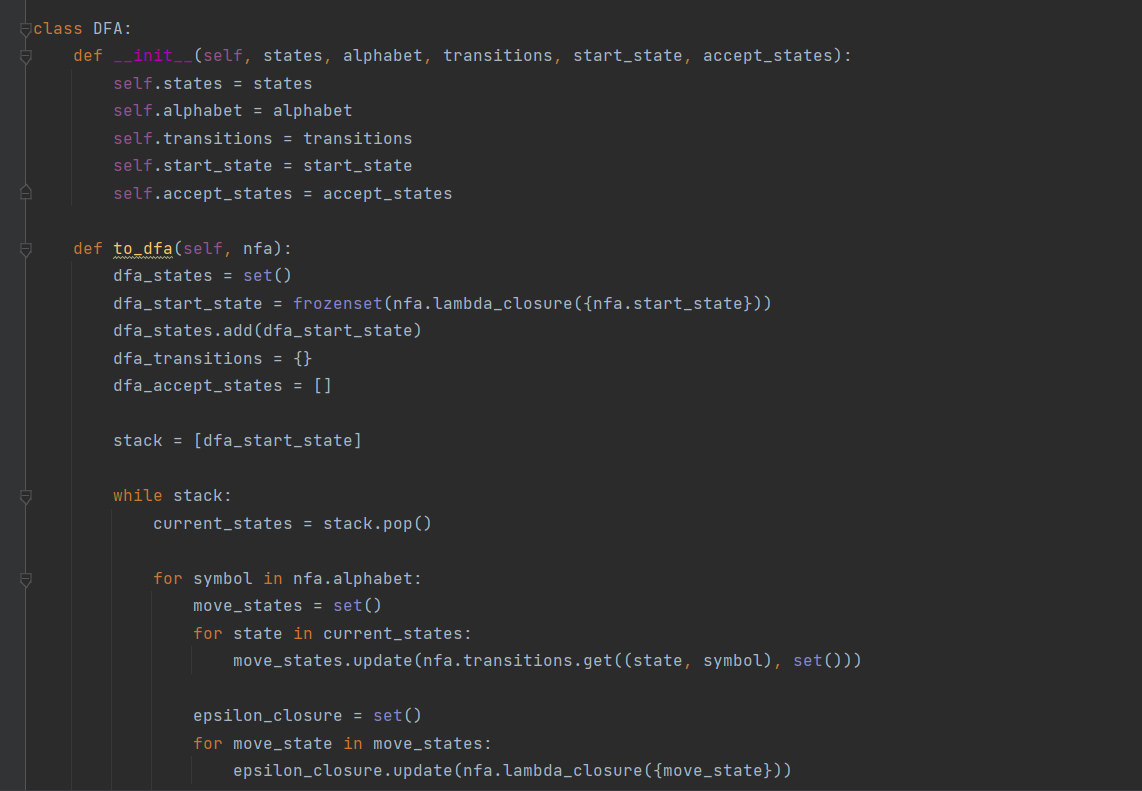


بخش2)

کلاس `NFA` مدلی از یک ماشین پذیرنده متناهی غیرقطعی (NFA) را نشان می‌دهد. این کلاس دارای ویژگی‌های مهمی مانند وضعیت‌ها (`states`)، الفبا (`alphabet`)، گذارها (`transitions`)، وضعیت شروع (`start\_state`) و وضعیت‌های قبولی (`accept\_states`) است. همچنین این کلاس توابعی مانند `lambda\_closure` و `move`و convert\_lambda\_nfa\_to\_nfa ‘ را برای محاسبه نقطه بسته فاصله‌ای (lambda closure) و گذارها دریافتی (move) و تبدیل لامبدا nfa به nfa ارائه می‌دهد.



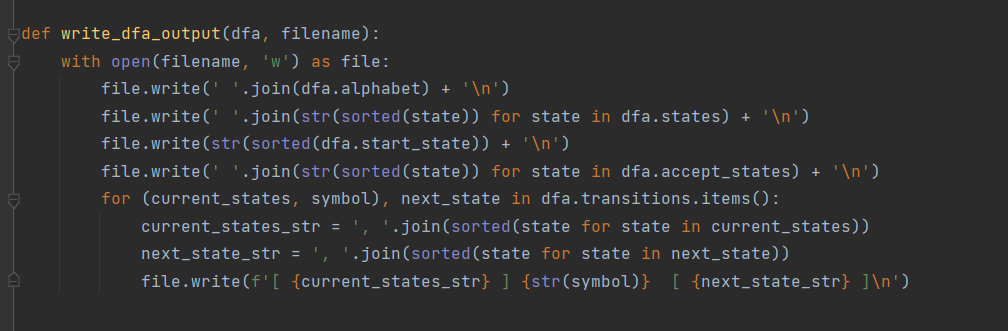
کلاس `DFA` مدلی از یک ماشین پذیرنده متناهی قطعی (DFA) را نشان می‌دهد. این کلاس نیز دارای ویژگی‌های مشابه با کلاس `NFA` است. همچنین تابع `to\_dfa` در این کلاس، یک NFA را به یک DFA تبدیل می‌کند.



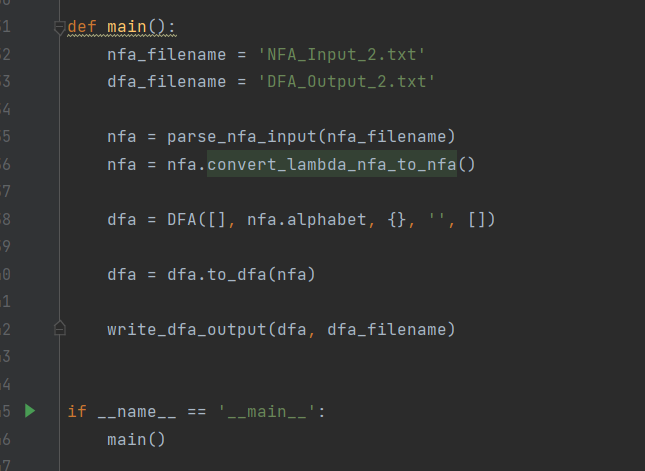
تابع `parse\_nfa\_input` وظیفه‌ای دارد که ورودی NFA را از یک فایل مشخص خوانده و اطلاعات مربوط به آن را استخراج کند و یک شیء NFA ایجاد کند.



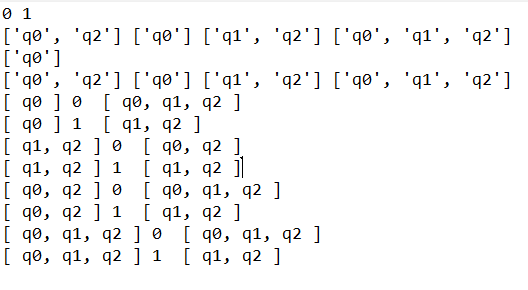
تابع `write\_dfa\_output` یک DFA را در یک فایل خروجی ذخیره می‌کند. این تابع مشخصات مربوط به DFA را به همراه گذاره‌ها در فایل خروجی ذخیره می‌کند.



تابع `main`، تابع اصلی برنامه است که وظیفه مدیریت فرآیند اجرای برنامه را بر عهده دارد. در این تابع، ابتدا فایل ورودی NFA خوانده می‌شود و سپس با استفاده از توابع `parse\_nfa\_input` و `to\_dfa`، NFA به DFA تبدیل می‌شود. در نهایت، خروجی DFA در یک فایل خروجی ذخیره می‌شود.



با اجرای این برنامه، فایلNFA\_Input2\_ خوانده می‌شود و نتیجه به صورت یک فایل خروجی با نام DFA\_Output\_2.txt ذخیره می‌شود.

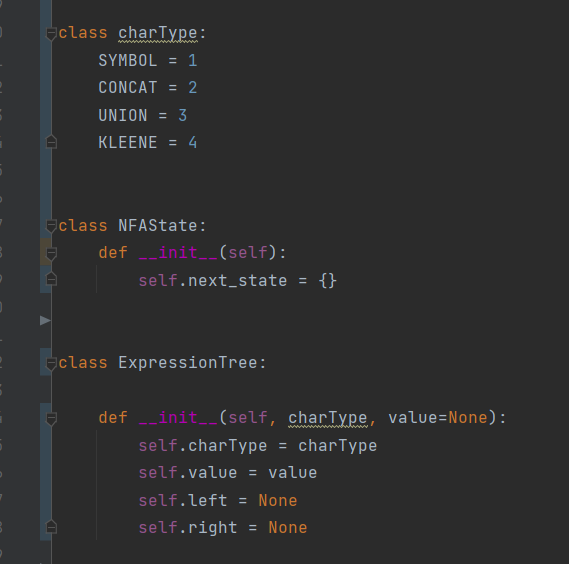


بخش3)

- `charType`: یک شاخص برای نوع کاراکترهای عبارت منظم. مقادیر ممکن برای این شاخص شامل: `SYMBOL`، `CONCAT`، `UNION` و `KLEENE` هستند که به ترتیب به ترتیب معادل با کاراکترهای عملگر، عملگر اتصال، عملگر اجتماع و عملگر تکرار صفر یا بیشتر هستند.

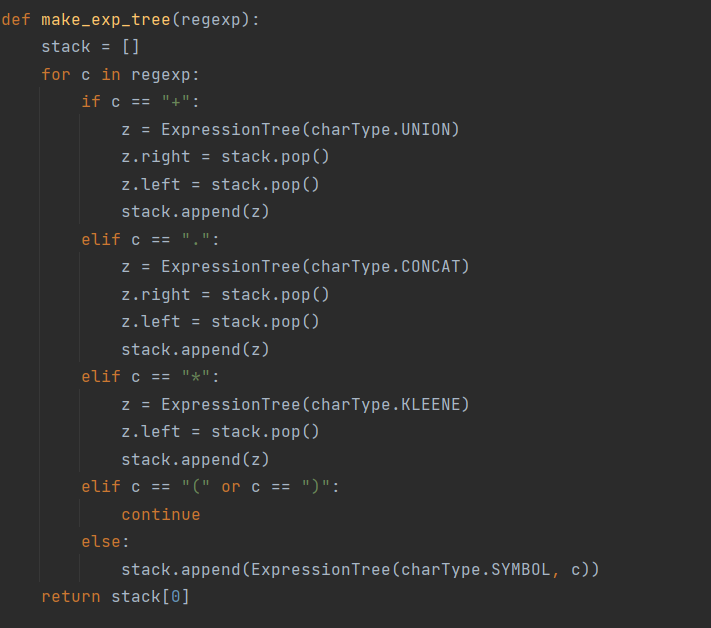
- `NFAState`: یک کلاس که وضعیت‌ها (استیت‌ها) را در اتومات نمایش می‌دهد. هر وضعیت دارای یک وضعیت بعدی است که به کلمه ورودی مشخصی وابسته است.

- `ExpressionTree`: یک کلاس که یک درخت بیان عبارت منظم را نمایش می‌دهد. هر گره از درخت دارای نوع کاراکتر و مقدار مربوطه است.

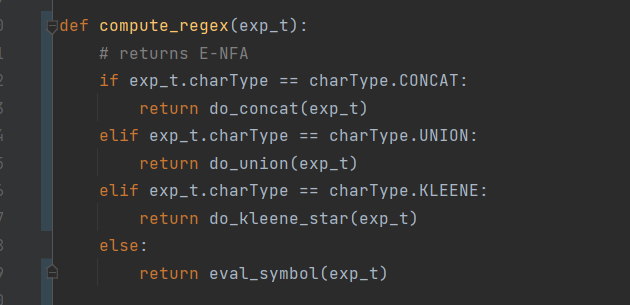


توابع مهم کد عبارتند از:

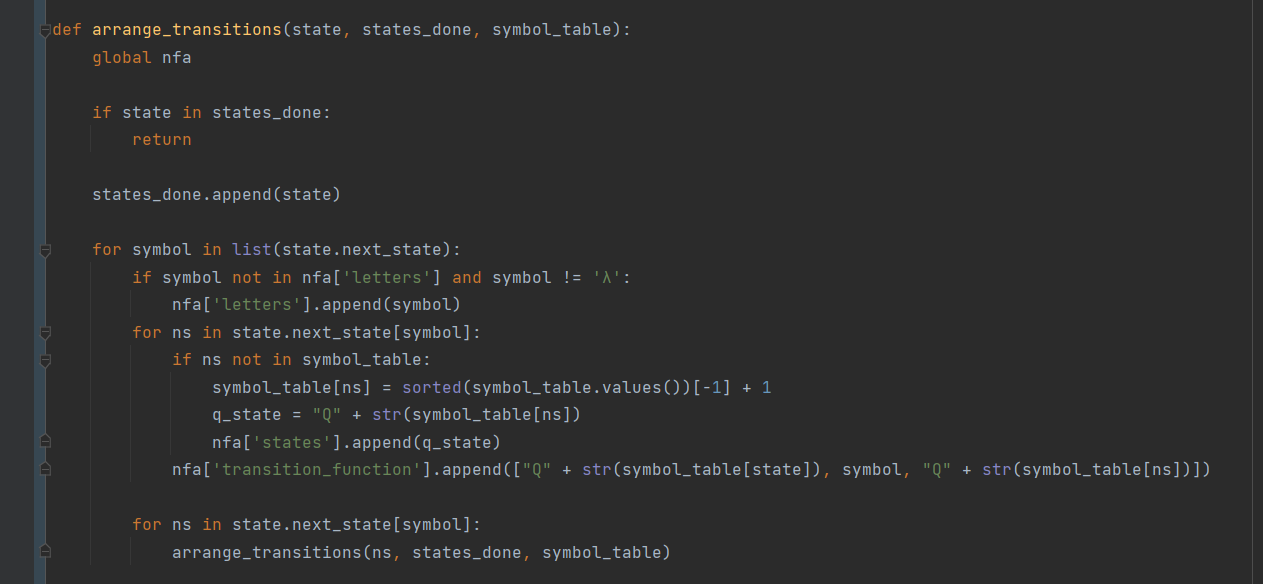
- `make\_exp\_tree`: با گرفتن عبارت منظم ورودی، درخت بیان مربوطه را برمی‌گرداند.



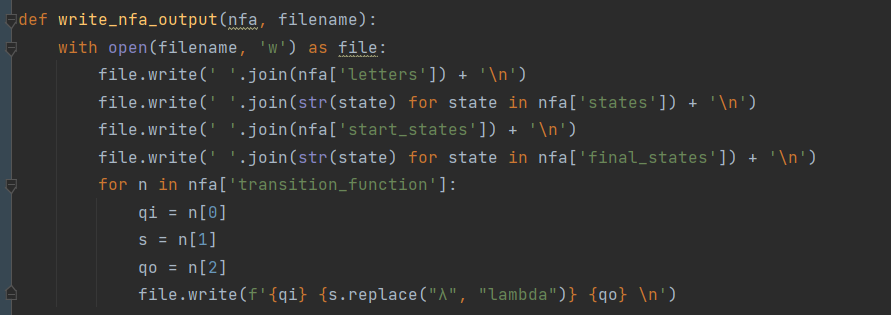
- `compute\_regex`: با گرفتن درخت بیان عبارت منظم، معادل آن NFA را محاسبه می‌کند.



- `arrange\_transitions`: تابع بازگشتی است که ترتیب گذار از وضعیت‌ها و ترانزیشن‌ها را در اتومات تنظیم می‌کند.



- `write\_nfa\_output`: این تابع NFA تولید شده را به یک فایل متنی ذخیره می‌کند.



در بخش اصلی برنامه، عبارت منظم ورودی از یک فایل خوانده می‌شود، سپس عبارت منظم ورودی به یک عبارت منظم با فرم پولیش شده تبدیل می‌شود. سپس از روی این عبارت منظم پس از ساختن درخت بیان و محاسبه NFA، اتومات تولید شده به فایل خروجی نوشته می‌شود.

