به موارد زیر توجه کنید:

* پروژه تحویل حضوری خواهد داشت.
* پروژه را در قالب گروه‌های ۲ یا ۳ نفره انجام دهید.
* مهلت تحویل امتیازی پروژه ۲۰ دی ماه و مهلت تحویل عادی آن ۸ بهمن ماه ساعت ۲۳:۵۹ می‌باشد.
* گروه‌هایی که تا قبل از مهلت امتیازی پروژه را تحویل دهند علاوه بر نمره اصلی، ۱۰ درصد از نمره دریافتی را به عنوان نمره امتیازی دریافت می‌کنند (نمره آنها شامل ضریب ۱.۱ می‌شود).
* مهلت تحویل امتیازی و نهایی به هیچ عنوان **قابل تمدید نیستند.**
* در نهایت تمام فایل‌های خود را در یک فایل زیپ با نام $P1\_StudentIDs$ قرار داده و در کوئرا آپلود کنید. آپلود یکی از اعضای گروه کافیست.
* دقت کنید که در صورت عدم پیاده‌سازی print باید یک ابزار نظارتی برای بررسی صحت اجرای برنامه‌ها داشته باشید.
* لطفا پروژه را از یکدیگر کپی نکنید. در صورت وقوع چنین مواردی مطابق با سیاست درس رفتار می‌شود.

۱ تعریف گرامر

در این بخش گرامر زبانی که قصد طراحی آن را داریم را مشاهده می‌کنید.

1. $Program \rightarrow Statements\ EOF$

2. $Statements \rightarrow Statement\ \text{';'} \mid Statements\ Statement\ \text{';'}$

3. $Statement \rightarrow Compound\_stmt \mid Simple\_stmt$

4. $Simple\_stmt \rightarrow Assignment \mid Global\_stmt \mid Return\_stmt$

$\text{'pass'} \mid \text{'break'} \mid \text{'continue'} \mid \text{'print'}\ \text{'()'} \mid \text{'print'}\ \text{'('}\ Arguments\ \text{')'}$

5. $Compound\_stmt \rightarrow Function\_def \mid If\_stmt \mid For\_stmt$

6. $Assignment \rightarrow ID\ \text{'='}\ Expression$

7. $Return\_stmt \rightarrow \text{'return'} \mid \text{'return'}\ Expression$

8. $Global\_stmt \rightarrow \text{'global'}\ ID$

9. $Function\_def \rightarrow \text{'def'}\ ID\ \text{'('}\ Params\ \text{')'}\ \text{':'}\ Statements$

$\mid \text{'def'}\ ID\ \text{'()'}\ \text{':'}\ Statements$

10. $Params \rightarrow Param\_with\_default \mid Params\ \text{','}\ Param\_with\_default$

11. $Param\_with\_default \rightarrow ID\ \text{'='}\ Expression$

12. $If\_stmt \rightarrow \text{'if'}\ Expression\ \text{':'}\ Statements\ Else\_block$

13. $Else\_block \rightarrow \text{'else'}\ \text{':'}\ Statements$

14. $For\_stmt \rightarrow \text{'for'}\ ID\ \text{'in'}\ Expression\ \text{':'}\ Statements$

15. $Expression \rightarrow Disjunction$

16. $Disjunction \rightarrow Conjunction \mid Disjunction\ \text{'or'}\ Conjunction$

17. $Conjunction \rightarrow Inversion \mid Conjunction\ \text{'and'}\ Inversion$

18. $Inversion \rightarrow \text{'not'}\ Inversion \mid Comparison$

19. $Comparison \rightarrow Eq\_Sum | Lt\_Sum \mid Gt\_Sum \mid Sum$

20. $Eq\_Sum \rightarrow sum \text{ ‘} == \text{ ‘} Sum$

21. $Lt\_Sum \rightarrow sum \text{ ‘} < \text{ ‘} Sum$

22. $Gt\_Sum \rightarrow sum \text{ ‘} > \text{ ‘} Sum$

23. $Sum \rightarrow Sum \text{ ‘} + \text{ ‘} Term \mid Sum \text{ ‘} - \text{ ‘} Term \mid Term$

24. $Term \rightarrow Term \text{ ‘} * \text{ ‘} Factor \mid Term \text{ ‘/‘} Factor \mid Factor$

25. $Factor \rightarrow \text{ ‘} + \text{ ‘} Power \mid \text{ ‘} - \text{ ‘} Power \mid Power$

26. $Power \rightarrow Atom \text{ ‘} * *\text{‘} Factor \mid Primary$

27. $Primary \rightarrow Atom \mid Primary \text{ ‘[‘} Expression \text{ ‘]‘} \mid Primary \text{ ‘()‘}$

$\mid Primary \text{ ‘(‘} Arguments \text{ ‘)‘}$

28. $Arguments \rightarrow Expression \mid Arguments \text{ ‘,‘} Expression$

29. ~~$Atom \rightarrow ID \mid \text{‘True‘} \mid \text{‘False‘} \mid \text{‘None‘} \mid NUMBER \mid List$~~

30. ~~$List \rightarrow \text{‘[‘} Expressions \text{ ‘]‘} \mid \text{‘[]‘}$~~

31. ~~$Expressions \rightarrow Expressions \text{ ‘,‘} Expression \mid Expression$~~

نکات گرامر:

- این گرامر (۱) LALR است بنابراین برای پیاده‌سازی‌های بعدی نیاز به هیچ‌گونه تغییری در آن نیست.
- توجه کنید که NUMBER اعداد مثبت صحیح و یا اعشاری است.
- موارد درون" terminal هایی هستند که به همین شکل در زبان ظاهر خواهند شد. مواردی که همه حروف آن‌ها بزرگ است مانند $NUMBER$ نیز termi-nal هستند با این تفاوت که مقدار مورد نظر برنامه‌نویس به جای آن‌ها می‌آید. کلماتی هم که با حروف بزرگ شروع می‌شوند nonterminal هستند.

۲ آشنایی با اسکنر و پارسر

برای راحتی کار، کد این دو قسمت در اختیارتان قرار داده شده است که می‌توانید در لینک زیر مشاهده کنید:

https://github.com/PL-Fall-۲۰۲۳/Project-Main-Parser.git

همچنین اگر علاقه‌مند بودید تا با نحوه پیاده‌سازی آن‌ها آشنا شوید، می‌توانید از لینک زیر استفاده کنید:

https://docs.racket-lang.org/parser-tools/index.html

۳ پیاده‌سازی مفسر (۷۰ نمره)

در این بخش شما باید به کمک آموخته‌های خود در درس، یک مفسر ساده برای این زبان پیاده‌سازی کنید.
دقت کنید که در تست‌های نهایی، برنامه‌ی با خطا داده نخواهد شد. بنابراین نیازی به پیاده‌سازی Error Handler نیست.
همانطور که در گرامر این زبان مشخص است برنامه‌های این زبان شامل تعدادی statement هستند که با ; از هم جدا شده اند. این گرامر ساده‌شده‌ی گرامر زبان python است و نحوه‌ی کلی برنامه مشابه پایتون خواهد بود. یکی از تفاوت‌های موجود آن است که بین هر دو statement یک ; خواهد آمد و در مقابل نیازی به رعایت INDENT نیست. حال به توضیح خط‌های مورد نیاز گرامر می‌پردازیم:

- $pass$ دستوری است که هیچ کاری انجام نمی‌دهد.
- خط ۹: در هنگام تعریف تابع باید به تمام متغیرهای آن مقدار اولیه بدهید.
- خط ۱۴: جلوی عبارت in تنها یک لیست می‌تواند بیاید. (این مورد بر خلاف پایتون است)
- عبارات شرطی مورد نیاز if و for تنها باید از نوع boolean باشند. (این مورد بر خلاف پایتون است.)
- خط ۲۳ و ۲۴ و ۲۶ گرامر: عملگرهای ریاضی تنها روی داده‌هایی از یک نوع قابل اجرا هستند.
- خط ۲۳ و ۲۴ و ۲۶ گرامر: برای نوع داده‌ی list تنها عملگر $+$ به معنای $concat$ (به هم چسباندن دو لیست) قابل اعمال است.
- خط ۲۵ گرامر: این دو عملگر $+$ و $-$ تنها بر روی اعداد قابل استفاده هستند.
- خط ۲۷: فراخوانی تابع به شکل call by value است.
- در نهایت در هر قسمتی از پیاده‌سازی نیازی به فرض خاصی داشتید، آن را در یک داک نوشته و به همراه پروژه ارسال کنید. (نیاز به تهیه داک در حالت کلی نیست.)

۴ خواندن کد از فایل (۵ نمره)

در این بخش باید یک تابع evaluate بنویسید که به عنوان ورودی آدرس کد موردنظر را گرفته و آن را اجرا کرده و خروجی را نمایش دهد. مثال:

```
evaluate("a.txt")
```

۵ تابع print (۵ نمره)

در تعریف گرامر، در $Simple\_stmt$ ها تابع پرینت قرار گرفته است. این تابع باید لیستی از $Arguments$ دریافت و آنها را پرینت کند. در این بخش باید این تابع را به شکل صحیح پیاده‌سازی کنید. دقت کنید که پیاده‌سازی این تابع نمره جداگانه از پیاده‌سازی مفسر زبان دارد.

۶ Lazy Evaluation (۲۰ نمره)

در این بخش شما باید Lazy Evaluation را پیاده‌سازی کنید که در زبان ما شامل موارد زیر می‌شود:

- در هنگام * کردن، در صورتی که سمت چپ ضرب ۰ بود، سمت راست محاسبه نشده و مقدار ۰ برگردانده می‌شود.
- یک متغیر که در Assignment مقدار دهی می‌شود، تا وقتی که از آن استفاده نشود، محاسبه نمی‌شود.
- تا وقتی از یک ورودی تابع استفاده نشده است، آن مقدار محاسبه نمی‌شود.

موفق باشید :")