Goals:

- Implement a basic recursive descent parser for MyPL (to check syntax but *not* build the AST);
- Practice working with and writing your own unit tests.

Instructions:

1. Use the GitHub Classroom link (posted in Piazza) to copy the starter code into your own repository. Clone the repository in the directory where you will be working on the assignment.
2. Copy your homework files from `HW-1`, except for `mypl.py`, `mypl`, and `hw1_tests.py`, into your repository. You *must* ensure that all files needed to run your code are added, committed, and pushed to your repository. If the graders try to run your code and there are missing files (which make it so that your parser can't run), you will receive a 0 for the assignment.
3. Complete the functions marked with `TODO` in `mypl_simple_parser.py`. This includes defining the additional functions needed to implement the recursive descent parser.
4. Ensure your code passes the unit tests provided in `hw2_tests.py`. (Note you will want to do steps 3 and 4 iteratively.)
5. Ensure your parser implementation correctly handles the example file within the `examples` subdirectory. Note that running `./mypl --parse` over this file should result in no output (implying the program is syntactically well formed).
6. Create additional interesting "positive" and "negative" expression tests in `hw2_tests.py`. See the comments marked `TODO` in the file. Note that there are two separate locations in the file to add the tests. You will need to add ten unit tests in total, five positive and five negative tests. By interesting we mean that the tests should be non-trivial, different to existing tests given, and be useful to ensure correctness of your implementation.
7. Copy the `parser.mypl` file, rename it to `parser_error.mypl`, save it in the main directory of your repo (not in `examples`), and add a hard-to-find syntax error to it in the second half of the file. When running `./mypl --parse` on this new file, it should result in `mypl` finding the syntax error.
8. Create a short write up as a **pdf file** named `hw2-writeup.pdf`. For this assignment, your write up should provide a short description of any challenges and/or issues you faced in finishing the assignment and how you addressed them along with a brief description of the unit tests you created as well as the syntax error you added to your `parser_error.mypl` example file.
9. Submit your program by ensuring all of your code, test file, and writeup is pushed to your GitHub repo. You can verify that your work has been submitted via the GitHub page for your repo.

Additional Requirements: Note that in addition to items listed below, details will also be discussed in class and in lecture notes.

1. Your recursive descent functions should generally follow the MyPL syntax rules given in the lecture notes. It is okay to deviate slightly in terms of the choice of recursive descent functions, but overall they should follow the general pattern described in class and induced by the grammar.
2. Since the MyPL grammar requires more than one lookahead in a few places, you will need to adapt some of your recursive descent functions respectively. In my implementation, e.g., there are a few functions that assume than an `ID` token has already been read.
3. You may not add additional helper functions in your implementation. You also cannot deviate from the general recursive descent approach in your implementation (see point 1 above).
4. You must use the `eat()` helper function when necessary, but should not over use it. In many cases `advance()` is all that is needed, namely, when you already know the type of token in the stream (e.g., after a `match()` call).
5. You will need to allow yourself enough time to think through some of the trickier parts of the parser. If you start this assignment too close to the deadline you will likely run out of time.
6. If you use any print statements for debugging, you must remove these from your final solution. In addition, you must remove all commented out code from your final submission.

Homework Submission and Grading. Your homework will be graded using the files you have pushed to your GitHub repository. Thus, you must ensure that all of the files needed to compile and run your code have been successfully pushed to your GitHub repo for the assignment. Note that this also includes your homework writeup. This homework assignment is worth a total of 40 points. The points will be allocated according to the following.

1. **Correct and Complete (30 points).** Your homework will be evaluated using a variety of different tests (for most assignments, via unit tests as well as test runs using specific input files). Each failed test will result in a loss of 2 points. If 15 or more tests fail, but some tests pass, 6 points (out of the 30) will be awarded as partial credit. Note that all 30 points may be deducted if your code does not run, large portions of work are missing or incomplete (e.g., stubbed out), and/or the specified techniques, design, or instructions were not followed.
2. **Evidence and Quality of Testing (5 points).** For each assignment, you must provide additional tests that you used to ensure your program works correctly. Note that for most assignments, a specific set of tests will be requested. A score of 0 is given if no additional tests are provided, 1–4 points if the tests are only partially completed (e.g., missing tests) or the tests provided are of low quality, and 5 if the minimum number of tests are provided and are of sufficient quality.

3. **Clean Code (2 points).** In this class, “clean code” refers to consistent and proper code formatting (indentation, white space, new lines), use of appropriate comments throughout the code, no debugging output, no commented out code, meaningful variable names and helper functions (if allowed), and overall well-organized, efficient, and straightforward code that uses standard coding techniques. A score of 0 is given if there are major issues, 1 if there are minor issues, and 2 if the “cleanliness” of the code submitted is satisfactory for the assignment.

4. **Writeup (3 points).** Each assignment will require you to provide a small writeup addressing challenges you faced and how you addressed them as well as an explanation of the tests you developed. Additional items may also be requested depending on the assignment. Homework writeups do not need to be long, and instead, should be clear and concise. A score of 0 is given if no writeup is provided, 1 if parts are missing, and 2 if the writeup is satisfactory.