Interpreter Project Description

In this project you are required to implement a basic Interpreter for Python.

Before Starting your project

1. Find group members.
2. Number of group members are recommended to be 3-5.
3. Find a cool name for your team.
4. Choose your programming language (OOP language), that you will implement your interpreter. Example: Java, C++, etc.
5. Create a Github repository for your interpreter project
6. Email your information. Deadline for sending the information above is November 2, 8:00pm.
7. Give **us** the necessary privileges to your Github repositories. Your commit history will be evaluated to grade your individual contribution to your project.
    a. Who is us?
        i. Caleb Heinzman (aka. the Haskell Slayer)
        ii. Rohit Nouduri (aka. the Prolog Tamer)
        iii. Ekincan Ufuktepe (aka. Javatar)

Tasks and requirements for the Interpreter

Please remember that we are not expecting a complete interpreter from you. You should implement your interpreter for Python 3.x, and should only support specific features of Python programming language. To learn more about which features are expected to be implemented, please check section "Expected Features for your Python Interpreter". Once you have checked the expected features for your interpreter, you must notice that there is no function or class implementation. You are mostly expected to implement a basic script.

NOTE: Remember that Python is indentation sensitive!

Expected Features for your Python Interpreter

- `if`/`else` blocks
- Variable definitions
- `while` and `for` Loops
- Arithmetic operators (+, -, *, /, %, ^)
- Assignment operators (=, +=, -=, *=, /=, ^=, %=)
- Conditional statements (<, <=, >, >=, ==, !=)
- Comments (lines that start with a '#')
- Support Output operation (**print** function).
- **BONUS:** Syntax error message (this is where we did the accept/reject string. If the given code aka. grammar is not a Python language, reject it. In other words, throw a syntax error message).

Grading Criteria

- A working interpreter – 60%
    - (15 pts)`if`/`else` blocks
    - (15 pts) Variable definitions

    - (15 pts) `while` and `for` Loops
    - (15 pts) Arithmetic operators (+, -, *, /, %, ^)
    - (15 pts) Assignment operators (=, +=, -=, *=, /=, ^=, %=)
    - (15 pts) Conditional statements (<, <=, >, >=, ==, !=)
    - (5 pts) Comments (lines that start with a '#')
    - (5 pts) Support Output operation (**print** function).
- Report (Writing README.md on your Github repo., see "Github Repository Page" section) – 10%
- Individual contribution – 30%
- Syntax error message (10 pts directly added to you project grade)

Rules & Recommendations

- You can have support from external libraries like **flex** and **bison**, which we mentioned in our class before. There should be C, C++ and Java version of these libraries. Please check the "Parsers for Compiler Design & Interpreters" slide for the references and links.
- While evaluating and grading your projects, we will be using a web-based plagiarism tool to detect if you have got the code from somewhere else. Any code we detect that has been obtained from another repository, the overall Interpreter Project will be graded as 0.
- The deadline for the project is, **December 10th, 2020, 11:59pm**. A script will be executed on December 10th, 2020, 11:59pm that will clone your projects automatically and your grades will be evaluated based on the cloned instances. However, you are also required to upload your projects as a zip file on Canvas for official grading. You can simply use the "Code → Download Zip" feature on Github and upload that version on Canvas.
- No presentation or extra documentation is required. The README.md file will be used as a documentation.

Github Repository Page

- Use your README.md file.
    - Explain your project.
    - Include your team members.
    - Include the requirements that is required for your interpreter.
    - Write a "How to use/run" the interpreter.

For inspiration

Python Interpreters

- Java - https://github.com/xia-st/JPython
- Java - https://github.com/nejeoui/interpreter
- C++ - https://github.com/SeTSeR/TinyPython
- C++ - https://github.com/cullum/piethon
- C++ - https://github.com/jackweatherford/PythonInterpreter

Haskell Interpreters

- Java - https://github.com/DavidWz/Haskell-Interpreter
- Java - https://github.com/kly4222/Advanced-Haskell-Interpreter
- Java - https://github.com/romulovff/MiniHaskell
- C# - https://github.com/ajlopez/Husky

- C++ - https://github.com/shooshx/zippypy
- Python - https://github.com/Zotek/haskell-interpreter