Script

* Page 2
  + As an overall introduction of this project, we are going to split this presentation into four major sections, which are CFG, AST, Implementation, and benchmark comparison respectively.
* Page 3:
  + The control flow generator, (commonly referred as CFG in the rest of the presentation), is a tool that take certain input, mainly code-based input, to output a graphical representation of flow the program execution.
* Page 4:
  + The main idea of this project is to implement a control graph generator with a foundation build upon the concept of an abstract syntax tree. The project starts off using the python AST package as the parser to identify and parse the input program, which is a piece of read-in code written in python. Processed the code is then stored in the format of an abstract syntax tree, which essentially is a tree-like structure that contains partitioned statements along with their associated conditions and expressions. The control flow graph is generated by recursively iterate through the processed abstract syntax tree. Graphviz is used as the drawing tool to create the final graph representation. Compared with the previous implementation, a lot more details were implemented in our code, such as list comprehension and try catch functions
* Page 5:
  + As a way of quality control to prevent random errors during execution, uncompilable inputs are detected in pre-execution. Class structures and function calls are stored as models in the form of syntax tree. Final output is in an aggregate format that respectively show each model as an independent control-flow graph.
* Page 6: Implementation steps
  + Although the conceptual idea of this project is done by others, we completely rewrote the code to ensure functionality, expandability and enhance code to allow a more extensive coverage to suit the python raw grammar.
  + The internal structure of this project contains three major classes that interjoined with each other.
    - The BasicBlock class is a doubly linked list which records each line of the input code, plus any potential function calls within the block. Each block has a unique id.
    - CFG: Each CFG class represents a separate function. It contains a dictionary with key parallel to the basic block id and value to its instance. We can also use a pair of adjacent block id to retrieve the condition between these blocks and such condition is show on the edge in the final output.
    - CFG visitor: inherent from the python ast package. It mainly defines the behavior when each type of node is traversed.
    - This section is covered shortly due to the time requirement, we are willing to further talk about it in the code reviewing session.
  + The rest of this page shows the implementation progress throughout the semester.
* Page 7