Step 1: Dataset creation

1. Clone Github repositories and store all .java files in one folder
2. Use dataset_creation.py to get the resulting .csv file. Paths for input folder need to be modified before execution (with original code snippets, and output path for masked identifier code snippets is required)
3. The .csv file is used for training. Current dataset is present here

Step 2: Training

1. Modify the dataset path and model path in tokens_classifier.py and execute the code to train models. These models are used to get the ideal number of tokens for the masked variable. (not trained in time for paper submission and thus not included in the paper)
2. Similarly, modify the configurations such as number of epochs and the paths to dataset and storing the model in variable_predictor.py to train models for the actual variable prediction step

Step 3: Testing

1. Modify the test dataset path and the model path in classifier_eval.py to get the accuracy and MSE for the models trained with token_classifier.py.
2. In a similar manner to classifier_eval.py, stat_sampling.py evaluates the random sampling technique to predict the number of mask tokens required for a variable name. Using the generated dataset, the probabilities for random sampling are already hard-coded. Only the test dataset path needs to be modified.
3. model_eval.py contains similar code as variable_predictor.py but here, the losses are calculated for both the trained model and the base model. The % difference, and the PLL values for both the base and trained model are printed for each code snippet in the test dataset. In an additional step, the ground truth variable name is replaced with a random variable name. The losses are evaluated for this name as well. This is the mock variable name as described in detail in the paper. The output is the losses in the text printed to the nohup file.
4. Modify the path to where the nohup file is stored and procTest.ipynb (or procTest.py) uses that text to extract the losses and create a .csv file. Using this, it generates the graphs used to evaluate identifier names and the fine-tuned model in the paper.
5. model_test.csv is the subset of data used to evaluate the model
6. test.csv is the dataset used for the evaluation of the readability metric and the fine-tuned model
7. train_1.csv, train_2.csv, train_3.csv are the subdivisions of the training data