Nomes: Eliane Maciel, Guilherme Stedille

Trabalho de Implementação

Introdução

Nos foi proposto a criação de um gerador de bytecodes para python a partir de um programa do mesmo, O python utiliza a biblioteca DIS (Disassembler for Python bytecode) para gerar os bytecodes do mesmo e assim compilados em um .pyc que os contém. A nossa abordagem constituiu em tentar confeccionar nossa própria DIS, após entender a DIS original bem como partes do seu funcionamento e dos seus fundamentos, resultando assim no trabalho que implementamos.

Solução

O programa foi desenvolvido na linguagem de programação Python (Versão 3), baseados em princípios da biblioteca de **dis.**

O arquivo codes.py contêm as váriaveis e códigos do assembly.

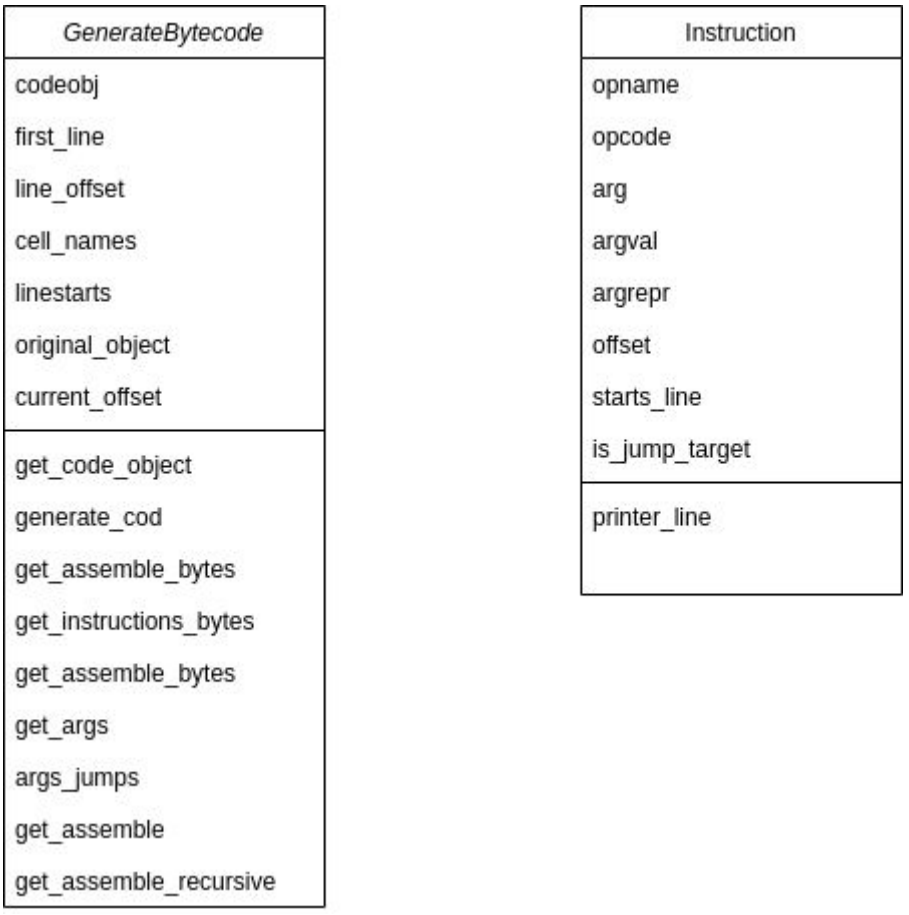
O programa faz a leitura de um arquivo texto e para ele utiliza o comando nativo do python **compile.**

O comando compile() que segue a seguinte sintaxe “compile(source, filename, mode, flags=0, dont_inherit=False, optimize=-1)” sendo Source, uma String, uma bytestring ou um objeto AST. Filename indicando o nome do arquivo que será fonte. Modo com três possibilidades, eval, aceitando apenas expressões únicas, exec, tomando um bloco de código com instruções do python e single, que consistem em uma única instrução iterativa. Flag e dont_inherit campos opcionais que controlam quais são as instruções futuras que apos o código compilado, atuam sobre ele. Essencialmente o comando Compile serve para pegar um codigo em string ou AST e transformá lo em um objeto tipo código, este novo objeto, retornado pela função compila pode, futuramente, ser chamado por outros métodos como por exemplo exec que poderão executar esse código de forma dinâmica.

Exemplo:

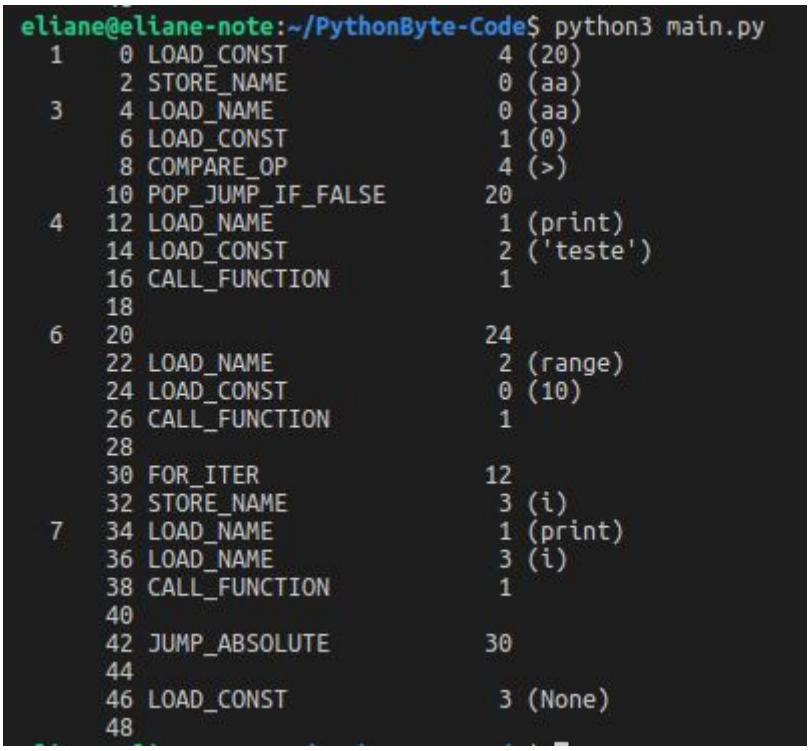
```
CODES[23] = 'BINARY_ADD'
CODES[24] = 'BINARY_SUBTRACT'
CODES[27] = 'BINARY_TRUE_DIVIDE'
CODES[29] = 'INPLACE_TRUE_DIVIDE'
```

Para a geração de códigos foi criada as classes que constam no diagrama abaixo:



A classe GenerateByteCode recebe um os comandos e para eles gera os códigos.

Exemplo da execução do programa:


eliane@eliane-note:~/PythonByte-Code$ python3 main.py
1 0 LOAD_CONST 4 (20)
2 STORE_NAME 0 (aa)
3 4 LOAD_NAME 0 (aa)
6 LOAD_CONST 1 (0)
8 COMPARE_OP 4 (>)
10 POP_JUMP_IF_FALSE 20
4 12 LOAD_NAME 1 (print)
14 LOAD_CONST 2 ('teste')
16 CALL_FUNCTION 1
18
6 20 24
22 LOAD_NAME 2 (range)
24 LOAD_CONST 0 (10)
26 CALL_FUNCTION 1
28
30 FOR_ITER 12
32 STORE_NAME 3 (i)
7 34 LOAD_NAME 1 (print)
36 LOAD_NAME 3 (i)
38 CALL_FUNCTION 1
40
42 JUMP_ABSOLUTE 30
44
46 LOAD_CONST 3 (None)
48