

# COL100 Assignment 5

Due date: 16 February 2022

Our task is to build an interpreter for a simple Python program with a limited syntax, simulating the computer's execution. Let us start by interpreting an input text file consisting of a sequence of statements, each statement on a separate line. The informal syntax of a statement is indicated below.

## Informal Grammar

A STATEMENT is of the form: VARIABLE = EXPRESSION

EXPRESSION is one of:

- TERM
- UNARY\_OPERATOR TERM
- TERM BINARY\_OPERATOR TERM

BINARY\_OPERATOR is one of: +, −, \*, /, >, <, >=, <=, ==, !=, and, or

UNARY\_OPERATOR is one of: −, not

TERM is one of: VARIABLE, INTEGER\_CONSTANT, True, False

VARIABLE is a sequence of one or more letters

INTEGER\_CONSTANT is a sequence of one or more numeric characters ('0' to '9')

## Interpreting the Input Program

Read the input file one statement at a time and INTERPRET (“execute”) the statement by maintaining all the variables and values encountered so far in a list called DATA.

To read a file in Python, use the built-in open() method. The open() function returns a file object, which has a read() method for reading the content of the file:

Example:

```
f = open("demofile.txt", "r")
print(f.read())
```

If your input file is located in a different location,

```
f = open("filepath/demofile.txt", "r")
```

```
print(f.read())
```

You can also use the open() method to create a new text file as follows-

```
f= open("demofile.txt","w+")
```

- We declared the variable “f” to open a file named demofile.txt. Open takes 2 arguments, the file that we want to open and a string that represents the kinds of permission or operation we want to do on the file
- We used “w” letter in our argument, which indicates Python write to file and it will create file in Python if it does not exist in library
- Plus sign indicates both read and write for Python create file operation.

You can learn more about I/O methods in Python:

<https://docs.python.org/3/tutorial/inputoutput.html>

<https://www.programiz.com/python-programming/file-operation>

## Example 1

Here is a possible list of actions taken by the interpreter for the input program:

```
x = 1 + 3
```

```
y = 4
```

```
x = 5
```

- On reading “x = 1 + 3”:
  - insert elements 1 and 3 into the DATA list.
  - Add 1 + 3 to get 4. Insert 4 into DATA. Let this be stored in list position  $i$  (that is, DATA[ $i$ ] contains 4).
  - insert element  $(x, i)$  into DATA. This is a way to implement “reference” ( $x$  refers to object 4).
- On reading “y = 4”:
  - search for 4 in the DATA list. Locate it in position  $i$  (DATA[ $i$ ] already contains 4).
  - insert list element  $(y, i)$  into DATA.
- On reading “x = 5”:
  - search for 5 in DATA. Since it is not present, insert new list element 5. Let this be in list position  $j$  (that is, DATA[ $j$ ] contains 5).
  - search for  $x$  in DATA. Replace  $(x, i)$  by  $(x, j)$ .  $x$  now refers to 5.
- After parsing the above lines final DATA list should be as follows:

<i>index</i>	0	1	2	3	4	5
<i>value</i>	1	3	4	$(x, 5)$	$(y, 2)$	5

- The final values of the variables in the DATA list are  $x = 5$  and  $y = 4$ . Garbage value are 1 and 3.

## Example 2

Here is a possible list of actions taken by the interpreter for the input program:

```
q = 6
p = q + 5
r = p
```

- On reading “ $q = 6$ ”:
  - insert element 6 into DATA. Let this be stored in list position  $i$  (DATA[ $i$ ] contains 6).
  - insert element  $(q, i)$  into DATA.
- On reading “ $p = q + 5$ ”:
  - search for 5 in DATA. Since it is not present, insert new list element 5. Let this be in list position  $j$  (that is, DATA[ $j$ ] contains 5).
  - search for  $q$  in DATA. Since  $(q, i)$  is present, follow the reference  $i$  and read the value 6 from DATA[ $i$ ].
  - Add 5 and 6 to obtain 11. Search for 11 in DATA. Since it is not present, insert new element 11. Let this be stored in list position  $k$  (that is, DATA[ $k$ ] contains 11).
  - search for  $p$  in DATA. Since it not present, insert list element  $(p, k)$  into DATA.
- On reading “ $r = p$ ”:
  - search for  $p$  in DATA. You will find  $(p, k)$ .
  - search for  $r$  in DATA. Since it is not present, insert  $(r, k)$  into DATA.
- After parsing the above lines, the final DATA list should be as follows:

<i>index</i>	0	1	2	3	4	5
<i>value</i>	6	$(q, 0)$	5	11	$(p, 3)$	$(r, 3)$

- The final values of the variables in the DATA list are  $q = 6$ ,  $p = 11$  and  $r = 11$ . Garbage value is 5.

## Example with error

Here are the possible actions taken by the interpreter for the input program:

```
x = 5
y = x + z
```

- On reading “ $x = 5$ ”:
  - insert element 5 into DATA. Let this be stored in list position  $i$  (DATA[ $i$ ] contains 5).
  - insert element  $(x, i)$  into DATA.
- On reading “ $y = x + z$ ”:
  - search for  $x$  in DATA. Since  $(x, i)$  is present, follow the reference  $i$  and read the value 5 from the list position  $i$  (that is, DATA[ $i$ ] contains 5).
  - search for  $z$  in DATA. Since it is not present, we can’t find the value of this expression. Issue the error message “**Variable ‘z’ is not defined**”.

## Output

When the program completes, print out:

- The name and current value of all the variables used in the input program.
- The list of GARBAGE integer objects used in the program but not referred to any more by any variable at the end of the program.

## Notes

- Number of lines in the input program file can take any value, your code should execute/interpret till it reaches the end of the file.
- The above language supports two types of variables: integer and boolean. All variables are scalar (no aggregate types such as list).
- If there is any error in the input, print the error and terminate the program.
- Use the Python **open**, **readlines**, and **split** functions to help in reading the input file. Assume that spaces separate the different syntactic “tokens” in each line (VARIABLE, TERM, =, +, etc.)
- Assume that there are no type mismatches (e.g., adding an integer to a boolean value).

About the split function

- Whenever there is a need to break bigger strings or a line into several small strings, you need to use the split() function in Python.
- The split() function still works if the separator is not specified by considering white spaces, as the separator to separate the given string or given line.
- The syntax of the split function is  
str.split(separator, maxsplit)

where,

- separator represents the delimiter based on which the given string or line is separated,
- maxsplit represents the number of times a given string or a line can be split up. The default value of max is -1. In case the max parameter is not specified, the split() function splits the given string or the line whenever a separator is encountered

You can learn more about the split function here

<https://www.programiz.com/python-programming/file-operation>