Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2019

Homework 4 – Python Program File Indentation Check with Stacks

Due: 27/3/2019, Wednesday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!

Introduction

In this homework, you are asked to implement a program which analyzes whether a given .py file (python source code) contains indentation errors. Hence, you must check if each line can be bounded by an indentation level correctly. To achieve this, the program must use a *stack* data structure. The details of the methods, algorithms and the data structure to be used will be given in the subsequent sections of this homework specification.

Program Flow, Input and Output

At first, the program asks for the name of the .py file to be analyzed. You should check whether the file has been opened successfully or not. In case of a file open failure, ask for a different file name until it is opened successfully. While reading the content of the .py file, your program is going to print the content information and indentation correctness for each line. As soon as a syntax error related to indentation is detected, your program will display an error message and will stop without checking the rest of the file. If all of the lines are checked correctly, then a general success message will be displayed at the end.

In Python language, code blocks are grouped according to their indentation levels, instead of using symbols like `{` and `}`, which are used to identify blocks in C++. Thus, there are specific rules for indentation in Python.

Indentation Rules to be Checked by Your Program and Other Assumptions:

1. Initial indentation level has depth zero (i.e. there must not be any leading blanks to start with).
2. Code lines belonging to the same indentation level (i.e. the code lines of the same block) must have the same number of spaces at the beginning of the lines.
3. The lines proceeding a keyword-containing line (keywords are explained below) must start a new indentation level. This new level is going to have a greater depth than the previous level. As long as the new level has more spaces than the previous one, new level depth is going to be valid.

4. A keyword-containing line is defined as a line that starts with a keyword (after some blanks, if any). You are responsible for checking 5 keywords, which are `if`, `else`, `elif`, `for` and `while`.
    a. You are not required to check the syntax related to the correctness of the order of these keywords. For example, your program will not check whether an `else` or `elif` has a preceding `if`.
5. It is possible to have nested blocks. That means, before ending a block, a new block may start. This way, nested indentation levels may be created.
6. When a block is finished, a python code may continue with one of the outer blocks (i.e. one of the previous indentation levels; one or more previous levels can be gone back in one line).
7. You are not going to check for Python syntax errors other than indentation errors. For instance, checking for incomplete assignment (x =), etc. is not in the scope of this homework (we are not going to create a compiler ☺).
8. You can assume that the input file does not contain any comments and does not contain any empty lines.
9. You can assume that no tab characters are used in .py files. Indentation level depths are counted by the number of leading spaces (blank characters) at the beginning of the lines.

Data Structure to be Used

In this homework you are going to use *STACKS ONLY*. It is used to keep track of the indentation level depth of the input .py file. The stack class that we provide together with the lecture notes, **DynIntStack.cpp** and **DynIntStack.h**, must be used in your program and they **must not be modified**. Your main program must be written as another cpp file.

You should delete all of the remaining nodes in the stack before finishing the program.

You cannot use any other multiple data container (vector, built-in array, dynamic array, other linked lists, etc.) in this homework.

Algorithmic Hint

You will only need a single *stack* to keep track of the indentation levels. At first, push the initial indentation level, which is 0, to the stack. Then,

process the input file line by line.

if the current line contains more leading spaces than the current indentation level and a KEYWORD is observed just BEFORE this line.

the line is correctly indented with a new indentation level. Push the new indentation level (number of leading spaces of the current line) to the stack.

else if a KEYWORD is observed just BEFORE this line, but the amount of spaces at the beginning of this line is less than or same as the current indentation level,

this is an error case.

else if NO KEYWORD is observed just BEFORE this line and the current indentation level has more spaces than the current line's number of leading spaces,

pop the current indentation level(s) from the stack until a correct block is found. If a correct level cannot be found, then this is an error.

else if NO KEYWORD is observed just BEFORE this line and the current indentation level has less spaces than the current line's number of leading spaces,

this is an error case.

else if NO KEYWORD is observed just BEFORE this line and the current indentation level has the same amount spaces as the current line's number of leading spaces,
the line is correctly indented and belongs to current indentation level.

Sample Runs

Some sample runs are given below, but these are not comprehensive, therefore you have to consider **all possible cases** to get full mark. The sample input files are provided in the .zip package of this homework.

Sample Run 1:
File: t1.py

```
import sys
import random
ans = True
while ans:
    question = raw_input("Ask the magic 8 ball a question: (press enter to quit) ")
    answers = random.randint(1,8)
    if question == "":
        sys.exit()
    elif answers == 1:
        print "It is certain"
    elif answers == 2:
        print "Outlook good"
    elif answers == 3:
                                                print "You may rely on it"
    elif answers == 4:
        print "Ask again later"
    elif answers == 5:
              print "Concentrate and ask again"
    elif answers == 6:
          print "Reply hazy, try again"
    elif answers == 7:
                        print "My reply is no"
    elif answers == 8:
        print "My sources say no"
print "End"
```

Please enter the file name that is going to be analyzed.
t1.py
Starting file analysis...
Initial indentation level is pushed to the stack as 0.

Line number: 1
Line: import sys
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.

Line number: 2
Line: import random
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.

Line number: 3
Line: ans = True
0 number of spaces observed before the start of the line.

Current Level = 0 This Line = 0
Line belongs to current block.

Line number: 4
Line: while ans:
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.
Keyword while found on this line.

Line number: 5
Line: question = raw_input("Ask the magic 8 ball a question: (press enter to quit) ")
4 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 0 This Line = 4
Line correct. Depth 4 added to the stack.

Line number: 6
Line: answers = random.randint(1,8)
4 number of spaces observed before the start of the line.
Current Level = 4 This Line = 4
Line belongs to current block.

Line number: 7
Line: if question == "":
4 number of spaces observed before the start of the line.
Current Level = 4 This Line = 4
Line belongs to current block.
Keyword if found on this line.

Line number: 8
Line: sys.exit()
8 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 4 This Line = 8
Line correct. Depth 8 added to the stack.

Line number: 9
Line: elif answers == 1:
4 number of spaces observed before the start of the line.
Current Level = 8 This Line = 4
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 4 This Line = 4
Line belongs to outer block.
Keyword elif found on this line.

Line number: 10
Line: print "It is certain"
8 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 4 This Line = 8
Line correct. Depth 8 added to the stack.

Line number: 11
Line: elif answers == 2:
4 number of spaces observed before the start of the line.
Current Level = 8 This Line = 4
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 4 This Line = 4
Line belongs to outer block.
Keyword elif found on this line.

Line number: 12
Line: print "Outlook good"
8 number of spaces observed before the start of the line.

This line proceeds a keyword containing line.
Current Level = 4 This Line = 8
Line correct. Depth 8 added to the stack.

Line number: 13
Line: elif answers == 3:
4 number of spaces observed before the start of the line.
Current Level = 8 This Line = 4
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 4 This Line = 4
Line belongs to outer block.
Keyword elif found on this line.

Line number: 14
Line: print "You may rely on it"
49 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 4 This Line = 49
Line correct. Depth 49 added to the stack.

Line number: 15
Line: elif answers == 4:
4 number of spaces observed before the start of the line.
Current Level = 49 This Line = 4
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 4 This Line = 4
Line belongs to outer block.
Keyword elif found on this line.

Line number: 16
Line: print "Ask again later"
8 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 4 This Line = 8
Line correct. Depth 8 added to the stack.

Line number: 17
Line: elif answers == 5:
4 number of spaces observed before the start of the line.
Current Level = 8 This Line = 4
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 4 This Line = 4
Line belongs to outer block.
Keyword elif found on this line.

Line number: 18
Line: print "Concentrate and ask again"
15 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 4 This Line = 15
Line correct. Depth 15 added to the stack.

Line number: 19
Line: elif answers == 6:
4 number of spaces observed before the start of the line.
Current Level = 15 This Line = 4
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 4 This Line = 4
Line belongs to outer block.
Keyword elif found on this line.

Line number: 20
Line: print "Reply hazy, try again"
10 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.

Current Level = 4 This Line = 10
Line correct. Depth 10 added to the stack.

Line number: 21
Line: elif answers == 7:
4 number of spaces observed before the start of the line.
Current Level = 10 This Line = 4
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 4 This Line = 4
Line belongs to outer block.
Keyword elif found on this line.

Line number: 22
Line: print "My reply is no"
24 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 4 This Line = 24
Line correct. Depth 24 added to the stack.

Line number: 23
Line: elif answers == 8:
4 number of spaces observed before the start of the line.
Current Level = 24 This Line = 4
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 4 This Line = 4
Line belongs to outer block.
Keyword elif found on this line.

Line number: 24
Line: print "My sources say no"
8 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 4 This Line = 8
Line correct. Depth 8 added to the stack.

Line number: 25
Line: print "End"
0 number of spaces observed before the start of the line.
Current Level = 8 This Line = 0
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 4 This Line = 0
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 0 This Line = 0
Line belongs to outer block.

Finished file analysis. File structure is correct!
Stack emptied and program ending.

Sample Run 2:
File: t2.py

```
  print "wrong start"
if error:
 print "error"
```

Please enter the file name that is going to be analyzed.
t2.txt
Unable to open file please enter a different file name.
t2.cpp
Unable to open file please enter a different file name.
t2.py
Starting file analysis...
Initial indentation level is pushed to the stack as 0.

Line number: 1
Line: print "wrong start"
2 number of spaces observed before the start of the line.
Current Level = 0 This Line = 2
Incorrect file structure.
Current line cannot be greater than the Current indentation level.
Stopping file analysis...

Stack emptied and program ending.

Sample Run 3:
File: t3.py

```
line_start = 0
if line_start == 0
 line_start = 1
else
       print "error"
 print "here"
```

Please enter the file name that is going to be analyzed.
t3.py
Starting file analysis...
Initial indentation level is pushed to the stack as 0.

Line number: 1
Line: line_start = 0
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.

Line number: 2
Line: if line_start == 0
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.
Keyword if found on this line.

Line number: 3
Line: line_start = 1
1 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 0 This Line = 1
Line correct. Depth 1 added to the stack.

Line number: 4
Line: else
0 number of spaces observed before the start of the line.
Current Level = 1 This Line = 0
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 0 This Line = 0
Line belongs to outer block.
Keyword else found on this line.

Line number: 5
Line: print "error"
7 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 0 This Line = 7
Line correct. Depth 7 added to the stack.

Line number: 6
Line: print "here"

1 number of spaces observed before the start of the line.
Current Level = 7 This Line = 1
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 0 This Line = 1
Incorrect file structure.
An indentation level same as the Current line is not found in outer blocks.
Stopping file analysis...

Stack emptied and program ending.

Sample Run 4:
File: t4.py

```
x = 1
else
y = 0
if x == 1
    x = 2
    if x == 2
         print "here"
```

Please enter the file name that is going to be analyzed.
p4.py
Unable to open file please enter a different file name.
t4.py
Starting file analysis...
Initial indentation level is pushed to the stack as 0.

Line number: 1
Line: x = 1
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.

Line number: 2
Line: else
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.
Keyword else found on this line.

Line number: 3
Line: y = 0
0 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 0 This Line = 0
Incorrect file structure.
Current line must be greater than the Current indentation level.
Stopping file analysis...

Stack emptied and program ending.

Sample Run 5:
File: t5.py

```
if True:
     while True:
          for x in range(1,20):
                    else
                          l =
```

Please enter the file name that is going to be analyzed.
t5.py
Starting file analysis...
Initial indentation level is pushed to the stack as 0.

Line number: 1
Line: if True:
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.
Keyword if found on this line.

Line number: 2
Line: while True:
5 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 0 This Line = 5
Line correct. Depth 5 added to the stack.
Keyword while found on this line.

Line number: 3
Line: for x in range(1,20):
10 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 5 This Line = 10
Line correct. Depth 10 added to the stack.
Keyword for found on this line.

Line number: 4
Line: else
20 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 10 This Line = 20
Line correct. Depth 20 added to the stack.
Keyword else found on this line.

Line number: 5
Line: l =
26 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 20 This Line = 26
Line correct. Depth 26 added to the stack.

Finished file analysis. File structure is correct!
Stack emptied and program ending.

Sample Run 6:
File: t6.py

```
if True:
     while True:
          for x in range(1,20):
                  if True:
                      while True:
                          else
                                    l =
print "End"
```

Please enter the file name that is going to be analyzed.
t6.py
Starting file analysis...
Initial indentation level is pushed to the stack as 0.

Line number: 1
Line: if True:
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.
Keyword if found on this line.

Line number: 2
Line: while True:
5 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 0 This Line = 5
Line correct. Depth 5 added to the stack.
Keyword while found on this line.

Line number: 3
Line: for x in range(1,20):
10 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 5 This Line = 10
Line correct. Depth 10 added to the stack.
Keyword for found on this line.

Line number: 4
Line: if True:
18 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 10 This Line = 18
Line correct. Depth 18 added to the stack.
Keyword if found on this line.

Line number: 5
Line: while True:
22 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 18 This Line = 22
Line correct. Depth 22 added to the stack.
Keyword while found on this line.

Line number: 6
Line: else
26 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 22 This Line = 26
Line correct. Depth 26 added to the stack.
Keyword else found on this line.

Line number: 7
Line: l =
35 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 26 This Line = 35
Line correct. Depth 35 added to the stack.

Line number: 8
Line: print "End"
0 number of spaces observed before the start of the line.
Current Level = 35 This Line = 0
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 26 This Line = 0
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 22 This Line = 0
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 18 This Line = 0

Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 10 This Line = 0
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 5 This Line = 0
Current line is smaller than Current indentation level; checking if line belongs to outer indentation.
Current Level = 0 This Line = 0
Line belongs to outer block.

Finished file analysis. File structure is correct!
Stack emptied and program ending.

Sample Run 7:
File: t7.py

```
if True:
     while True:
          a = 78
          for x in range(1,20):
              if x=1:
          print "Muslera"
print "End"
```

Please enter the file name that is going to be analyzed.
t7.py
Starting file analysis...
Initial indentation level is pushed to the stack as 0.

Line number: 1
Line: if True:
0 number of spaces observed before the start of the line.
Current Level = 0 This Line = 0
Line belongs to current block.
Keyword if found on this line.

Line number: 2
Line: while True:
5 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 0 This Line = 5
Line correct. Depth 5 added to the stack.
Keyword while found on this line.

Line number: 3
Line: a = 78
10 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 5 This Line = 10
Line correct. Depth 10 added to the stack.

Line number: 4
Line: for x in range(1,20):
10 number of spaces observed before the start of the line.
Current Level = 10 This Line = 10
Line belongs to current block.
Keyword for found on this line.

Line number: 5
Line: if x=1:
14 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 10 This Line = 14
Line correct. Depth 14 added to the stack.
Keyword if found on this line.

```
Line number: 6
Line:           print "Muslera"
10 number of spaces observed before the start of the line.
This line proceeds a keyword containing line.
Current Level = 14 This Line = 10
Incorrect file structure.
Current line must be greater than the Current indentation level.
Stopping file analysis...

Stack emptied and program ending.
```

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate. Since using classes is mandated in this homework, a proper object oriented design and implementation will also be considered in grading.

Since you will use dynamic memory allocation in this homework, it is very crucial to properly manage the allocated area and return the deleted parts to the heap whenever appropriate. Inefficient use of memory may reduce your grade.

When we grade your homework we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases**. Of course, your program should work in *Debug* mode as well.

You are allowed to use sample codes shared with the class by the instructor and TAs. However, you cannot start with an existing .cpp or .h file directly and update it; you have start with an empty file. Only the necessary parts of the shared code files can be used and these parts must be clearly marked in your homework by putting comments like the following. Even if you take a piece of code and update it slightly, you have to put a similar marking (by adding "`and updated`" to the comments below.

```
/* Begin: code taken from ptrfunc.cpp */

…

/* End: code taken from ptrfunc.cpp */
```

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade. Name your solution, project, cpp file that contains your main program using the following convention (the necessary file extensions such as .sln, .cpp, etc, are to be added to it):

"SUCourseUserName_YourLastname_YourName_HWnumber"

Your SUCourse user name is actually your SUNet user name which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw4

In some homework assignments, you may need to have more than one .cpp or .h files to submit. In this case add informative phrases after the hw number. However, do not add any other character or phrase to the file names.

Now let us explain which files will be included in the submitted package. Visual Studio 2012 will create two *debug* folders, one for the solution and the other one for the project. You should delete these two *debug* folders. Moreover, if you have run your program in release mode, Visual Studio may create *release* folders; you should delete these as well. Apart from these, Visual Studio 2012 creates a file extension of *.sdf*; you will also delete this file. The remaining content of your solution folder is to be submitted after compression. Compress your solution and project folders using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains all of the solution, project and source code files that belong to the latest version of your homework. Especially double-check that the zip file contains your cpp and (if any) header files that you wrote for the homework.

Moreover, we strongly recommend you to check whether your zip file will open up and run correctly. To do so, unzip your zip file to another location. Then, open your solution by clicking the file that has a file extension of .sln. Clean, build and run the solution; if there is no problem, you could submit your zip file. Please note that the deleted files/folders may be regenerated after you build and run your program; this is normal, but do not include them in the submitted zip file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct files. The naming convention of the zip file is the same. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example, zubzipler_Zipleroglu_Zubeyir_hw4.zip is a valid name, but

Hw4_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!

Albert Levi, Taha Atahan Akyıldız