**SSN COLLEGE OF ENGINEERING**

UIT2201 – PROGRAMMING AND DATA STRUCTURES

PSP Exercise-02

U. Pranaav 3122 22 5002 093 IT-B

1.

I. AIM:

Write a Python program that reads all the Python source code files from a given Python package and reports the Lines Of Code (LOC). LOC excludes the lines that are only comments.

II. CODE:

# -\*- coding: utf-8 -\*-

'''

This module contains python functions use a series of functions

to find the total lines of code by excluding all comments, blank

lines as well as documentation in all files in a given folder

as well as its subdirectories.

In this source code I have executed my own logic. The code

follows good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 18 2023

Revised on Wed Apr 30 2023

Original Author: U. Pranaav <pranaav2210205@ssn.edu.in>

'''

import ast

import os

# Define a function to extract function names from the AST

*def* extract\_functions(*file\_path*):

    '''

    Given function retrieves all the file names in a

    given file path.

    args:

        file\_path: file in which we need to find functions.

    returns:

        list of function names.

    '''

    function\_list = []

    with open(*file\_path*,'r',*encoding*='utf-8',*errors*='replace') as file:

        node = ast.parse(file.read())

    for item in node.body:

        if isinstance(item, ast.FunctionDef):

            function\_list.append(item.name)

    return function\_list

*def* get\_function\_lengths(*filepath*):

    '''

    Counts the number of lines in a given function

    at the given file path.

    args:

        filepath: the path to the file in which function

        lines need to be counted.

    returns:

        Total number of lines in a function.

    '''

    with open(*filepath*, 'r') as file:

        lines = file.readlines()

        functions = {}

        start\_line = None

        for i, line in enumerate(lines):

            if line.startswith('def'):

                if start\_line is not None:

                    functions[name] = i - start\_line - 1

                name = line.split()[1].split('(')[0]

                start\_line = i

        if start\_line is not None:

            functions[name] = len(lines) - start\_line - 1

    return functions

*def* count\_ignore\_doc(*f\_path*,*fnames*):

    '''

    It takes in file path and a file name and finds

    whether a given line of code within each file

    is a docstring, comment or a blank line and also

    counts the number of lines in each function inside

    a class as well as normal functions.

    args:

        f\_path: the file path.

        fnames: the list containing names of

        functions.

    returns:

        Total number of lines and number of lines in a

        file of only functions in a tuple.

    '''

    counts = []

    mains\_list = []

    for fname in *fnames*:

        file\_path = os.path.join(*f\_path*,fname)

        functions = get\_function\_lengths(file\_path)

        with open(file\_path,'r',*encoding*='utf-8',*errors*='replace') as main\_file:

            main\_file.seek(0)

            is\_docstring = False

            count = 0

            comment\_count = 0

            space\_count = 0

            lines = main\_file.readlines()

            for line\_ind in range(len(lines)):

                lines[line\_ind]=lines[line\_ind].lstrip()

                if len(lines[line\_ind]) == 0:

                    if not is\_docstring:

                        space\_count += 1

                        continue

                if lines[line\_ind].startswith('#'):

                    comment\_count += 1

                    continue

                if lines[line\_ind].count("'''") == 1 or lines[line\_ind].count('"""') == 1:

                    is\_docstring = ~(is\_docstring)

                if not is\_docstring:

                    if lines[line\_ind].count('"""') == 2 or lines[line\_ind].count("'''") == 2:

                        continue

                    else:

                        count += 1

            counts.append(count)

            mains\_list.append(functions)

    return (counts,mains\_count(mains\_list))

*def* mains\_count(*main\_list*):

    '''

    Counts the number of lines in a given mains\_list

    containing dictionary of function lengths.

    args:

        main\_list: the list containing dictionary of

        lengths of functions.

    returns:

        Total number of lines.

    '''

    count = 0

    for i in *main\_list*:

        for j in i.values():

            count += j

    return count

*def* py\_LOC\_count(*file\_path*):

    '''

        This function takes in a file path as input and returns

        the count of number of lines of code and lines of

        functions.

        The input is not modified and there are no side effects.

        args:

            file\_path: the file path

        Returns:

            The counts of number of lines of code and number of

            lines of function in class as well as normal functions.

    '''

    global Total\_vals

    files\_path\_dict = {}

    for dir\_path, dir\_names, file\_names in os.walk(*file\_path*):

        files\_path\_dict[dir\_path] = file\_names

    Total\_lines = 0

    for path in files\_path\_dict.keys():

        pathway = path

        files\_to\_count = files\_path\_dict.get(path)

        val = count\_ignore\_doc(pathway,files\_to\_count)

        for i in val[0]:

            Total\_lines += i

        Total\_vals.append(val)

    return (Total\_lines,Total\_vals)

#driver code

if \_\_name\_\_ == '\_\_main\_\_':

    #this part of the code will only be run when the function is called directly

    #it will not be executed when it is imported as a module

    #writing the file path

    py\_path = *r*'D:\computer science\xml'

    # Open the Python file and read the contents

    Total\_vals = []

    total\_count, lines\_in\_each\_file = py\_LOC\_count(py\_path)

    print("Total number of lines is:",total\_count)

    print()

    print("Number of lines in each file in each directory is:\n")

    print()

    for val in lines\_in\_each\_file:

        print(val[0])

    print()

    print("Number of lines of functions in each directory is:\n")

    for val in lines\_in\_each\_file:

        print(val[1])

    print()

    #end of code

III. OUTPUT:

Total number of lines is: 5666

Number of lines in each file in each directory is:

[4, 2]

[60, 716, 48, 1535, 20, 284, 293, 83]

[1, 85, 304, 1242, 0]

[4, 1]

[334, 89, 242, 200, 62, 57]

Number of lines of functions in each directory is:

0

2445

2376

0

443