Python Project

You have to implement a python tool that checks deprecated code in the given files and replace it with the working code. Note: Put these files in a single folder.

The tool should not use any third party libraries. You are allowed to use any built-in libraries in python.

Explanation:

The files attached herewith have some lines of deprecated code, which will not work with a new version. Your task is to create filters that identify the deprecated code and replace them with the correct and working code, which will be compatible with the new version.

Filters:

Filter1 – find any code that imports “address” and prefix it with “name.”

1. “import address.city.state” should be replaced with “import name.address.city.state”
2. “from address import city” should be replaced with “from name.address import city”

Filter2 – find any import statements that import libBase and delete them

1. “from lib.base import libBase”, “from base import libBase as Base” etc., should be deleted.

Filter3 – find classes that inherit from any other class and replace them with “object”

1. “class A(LibBase)” should be replaced with “class A(object)”

Filter4 – find any init methods and super calls to init methods that has \*args, \*\*kw and replace them with “testInstance”

1. “def \_\_init\_\_(self, \*args, \*\*kw)” should be replaced with “def \_\_init\_\_(self, testInstance)”

Filter5 – find any calls with “self.dut” as the first argument and replace it with “self.dut.dut”

1. “self.DeviceContext(self.dut, whatever)” should be replaced with “self.DeviceContext(self.dut.dut, whatever)”

Filter6 – find any calls with “self.testInstance.dut” as the first argument and replace it with “self.testInstance.dut.dut”

1. “self.DeviceContext(self.testInstance.dut, whatever)” should be replaced with “self.DeviceContext(self.testInstance,.dut.dut, whatever)”

Your tool should support command line arguments and should accomplish all of the following features.

1. If the tool is run with no command line arguments, then it should scan the folder for all files and output the deprecated filter found in all of them. If it is run with one or more files in the command line then it should just report deprecation for those files only.
2. If the tool is run with “—cleanup [filtername]”as the command line argument then it should scan the folder for all files and replace them with the correct code. If it is run with “—cleanup [filtername] one or more files”, then it should replace those files with the correct code.

Logging Format:

[TimeStamp] [ToolModuleName] – Deprecations found in [filename]

filter1 – single line explanation of the filter

filter2 – single line explanation of the filter

[TimeStamp] [ToolModuleName] – Cleaned up Deprecations in [filename]

filter1, filter2 cleaned up.

Eg.,

05:08:2016:min:sec DeprecationTool – Deprecations found in file1.py

filter5 – replace self.dut with self.dut.dut

05:08:2106:min:sec DeprecationTool – Cleaned up Deprecations in file1.py

filter5 clean up.