In [28]:

```
import os
import re
import sys
import glob
import shutil
import pandas as pd
from pathlib import Path



class FileSearchCopy:

    """
    A Simple class implementation to check if a source directory exists.
    If source does not exist, then the working directory is change to the current
    directory of the Python_tag module

    :param source: source of the file location
    :param destination: the destination directory where you want to copy the files

    """

    def __init__(self, source, destination):

        self.source = source
        self.destination = destination
        self.not_found_files = []

        try:
            os.chdir(self.source)

        except FileNotFoundError as fe:
            print(fe,'\n\n')
            # Forcing directory change to this path but up one directory
            self.source = os.getcwd()
            print('Directory changed to {}\n\n'.format(self.source))
            # Caution this could have unintended file transfers though


    def search_files(self, file_type="*.html", search_string="<html>"):


        """
        Search for files by glob.glob(<regex | extension type|> )
        This function is not for any file associated as excel spreadsheets

        :param file: type:string   <regex>.extension
        :param search_string: type:string   regex
        """

        file_list = []
        garbage = []
        self.found_files = []
        file_check = glob.glob(file_type)


        if not file_type == '*.xlsm' or '*.xlsx':

            for root, dirs, files in os.walk(".", topdown=True):
                for file in files:
                    if file in file_check:
                        if not file in self.found_files:
                            self.found_files.append(file)


            for file in self.found_files:
                with open(os.path.join(self.source, file)) as f:
```

```python
                    if re.findall(search_string, f.read()):
                        file_list.append(file)
                    else:
                        garbage.append(file)

            return (file_list, garbage)


    def search_xlsm_files(self, file_type="*.xlsm", search_string=None):


        """
        Search for files by glob.glob(*.xlsm | *.xlsx )
        This function is only for any file associated as excel spreadsheets

        :param file: type:string   <regex>.extension
        :param search_string: type:string   regex
        """

        xls_list = []
        garbage = []
        self.found_files = []
        file_check = glob.glob(file_type)

        if file_type == '*.xlsm' or '*.xlsx':

            for root, dirs, files in os.walk(".", topdown=True):
                for file in files:
                    if file in file_check:
                        if not file in self.found_files:
                            self.found_files.append(file)

            for xls_file in self.found_files:
                df = pd.read_excel(xls_file)
                if search_string in df:
                    xls_list.append(xls_file)

            return (xls_list, garbage)



    def copy_good_files(self, file_list):

        """
        Copy all the files to source destination
        :param file_container: This container should
        """
        if Path(self.destination).exists:
            for file in file_list:
                shutil.copy(file, self.destination)

        else:
            print("Directory doesn't exist")



    def copy_good_xls_files(self, xls_list):
        """
        Copy all the files to source destination
        """
        if Path(self.destination).exists:
            for file in xls_list:
                shutil.copy(file, self.destination)

        else:
            print("Directory doesn't exist")
    @staticmethod
    def which_dir():
        print("Directory::::: {} ::::::".format(os.getcwd()))

    @staticmethod
    def walk_dir(directory):

        try:
            walk = os.walk(directory, topdown=True)
            print("Walk : {}".format(next(walk)))
        except StopIteration:
            print('No directory to walk')

    @staticmethod
    def change_dir(directory):
        try:
            os.chdir(directory)

        except FileNotFoundError as fe:
            print(fe,'\n\n')

    @staticmethod
    def make_dir(directory):
        pass


def run_main():
    #Replace directories before publishing
    rs = FileSearchCopy(source='C:\\Users\\ccurrent\\Desktop\\New folder\\bash_practice\\', \
           destination="C:\\Users\\ccurrent\\Desktop\\New Folder\\bash_practice\\testfolder\\")

    file_list, _ = rs.search_files(file_type="*.txt", search_string="(regex)")

    rs.copy_good_files(file_list)

    xls_list, _ = rs.search_xlsm_files(file_type="*.xlsx", search_string="(regex)")

    rs.copy_good_xls_files(xls_list)


if __name__== "__main__":
    run_main()
```