Mohamed Mohamed

SE 211

Professor Vokolos

28 January 2020

Documentation for Main Application: Assignment #1

Main Layout:

The test application's interface is rather simple in its layout. The application comprises of a single function that can called within the outer scope. Within the function, we see have a total of 4 if-elif statements that depending on the user's selection, the function will branch into a different set of executions. For example, if the user inputs a "1" that corresponds to the menu's list of tasks, then the function will execute the task that listed within the main menu. The main program utilizes recursion to simulate a task finishing execution and returning the main menu. If a user input an invalid input, the program will exit as a form of error handling.

Task #1: Changing the separator

```
if maininp == "1": #First option that changes the separator to make the file more readable.
    while coninp != "":
        coninp = input("What files do you want converted? (Press Enter at the next prompt to start execution: ")
        parsedcsvs.append(coninp) #Library function that changes the separator for the CSV.
    if len(parsedcsvs) < 1:
        sys.exit()
    parsedcsvs.pop()
    for file in parsedcsvs: #Check if file is in current directory.
        if os.path.isfile(file):
            csvconvert.csvcon(file)
        else:
            print("One of your files might not exist. Exitting.")
            menuOpp()
    print("Files have been converted successfully.")
    menuOpp()
```

The first task executes when the user inputs “1”, within the main menu. For the users to select the files that they want to modify, an input is looped and appends the inputs into a list. The input variable serves as the variable that determines whether the loop finish execution. Once the user inputs an empty string, the program removes the list’s last element, which is an empty string that could lead to a file error. The program enters a second loop that iterates through the list of inputs that the user has inputted. It will check whether all files within that list exist. If so, the program called the CSV library’s “csvcon” function, and changes the separator within all files and sends the user back to the main menu. If one of the files doesn’t exist (using the os.path module), the program will cease functionality and print an error message stating “One of your files might not exist. Exiting.”. It will then recursively called the “menuOpp” function, which will return the user back to the main menu.

Task #2: Read a file

```
elif mainInp == "2": #Reads the file.
    readinp = input("What file would you like to read?: ")
    if os.path.isfile(readinp):
        readfile = open(readinp, "r")
        print("\n")
        print("File read:")
        print("-------------------------------------------")
        csvconvert.csvprint(readinp) #Prints out the file's contents.
        menuOpp()
    else:
        print("File does not exist. Exitting.")
        menuOpp()

    menuOpp()
```

The second task deals in reading files into the Python interpreter. The second task executes when the user inputs “2” within the main menu. The user will be asked for an input, which is the file they want to read. The program will then check whether the input is a valid file within the folder the program is in. If the program sees that the file exists, it will output the file’s contents. If the file did not exist, the program will print a statement stating that the file does not exist, and will recursively call the function, menuOpp to return the user into the main menu.

Task #3: Read a file and replace a value

```
elif maininp == "3": #Reads a file and replaces any input noted by the user.
    readinp = input("What file would you like to change?: ")
    if os.path.isfile(readinp):
        readfile = open(readinp, "r")
        print("\n")
        csvconvert.csvprint(readinp)
        ogwordinput = input("What value do you want to replace?: ")
        newwordinput = input("What value do you want to insert?: ")
        csvconvert.csvreplace(readinp, ogwordinput, newwordinput) #Function that replace old word with new word.
        print("\n")
        print("File")
        print("-------------------------------------------------")
        csvconvert.csvprint(readinp)
        menuOpp()
    else:
        print("File does not exist. Exitting.")
        menuOpp()
```

The third task begins similarly to the previous task as it reads a file, whoever the program will prompt the user in giving an input that you want replaced and an input that you want to replace. The library's function "csvreplace" will execute the task of replace the inputs. The file will output the file once the replaced word has been incorporated into the file. If the file that you stated does not exist, then the program will exit out of the if statement, and recursively call menuOpp to return to the main menu.

Task #4: Append into a file

```
elif maininp == "4": #Option that appends input
    readinp = input("What file do you wish to append to?: ")
    if os.path.isfile(readinp):
        print("File selected:")
        print("-------------------------------------------------")
        csvconvert.csvprint(readinp)
        while appinp != "":
            appinp = input("What text do you wish to append onto the file (Press Enter to cease appending)?: ")
            csvconvert.csvappend(readinp, appinp + "\n") #Appends a row written by the user in the selected file.
            print("-------------------------------------------------")
            print("File has been modified.")
            print("-------------------------------------------------")
            csvconvert.csvprint(readinp)
        menuOpp()
    else:
        print("File does not exist. Exitting.")
        menuOpp()
```

The fourth task deals in appending information into a preexisting file. The user will give an input that will be tested to see if it's an existing file through the os module's method. If the file exists, the file will be printed into the console. What follows is an input that loops over. For every input that is entered, within the loop, a new row is added into an existing file. Once the

loop breaks, the file with the appended information will be printed and will send the user into the main menu. If the file does not exist, the program will print a message about the file not existing and will call menuOpp to return the user to the main menu.

Issues with Library/Application

Due to the overall lack of support in Python, the library does not share any native support with CSV that are associated in Excel. The library and application will not print CSV in Excel in the traditional method. Instead, the library will take the content of the selected file and instead print the contents inside one single cell.