Python Assignment

Cian Herlihy – R00205604

# Task 1

*"""  
Cian Herlihy - R00205604 - Python Assignment  
  
To complete this task. I wanted to take a list and check if the words (Strings) have '?' in them.  
I did this by iterating through the list of words and use '.count()' to check how many times it includes  
the character '?'. If it contains the character then I print out that word saying it contains it.  
  
I also had to check what letters were common, so I took the first word on the list and compared it against  
the other words in the list. I check if the first letter is in all words and if it is then I add it to a list.  
I increment a counter to show it was in the word, but I only increment it by 1 even if it is in twice or more.  
This is because I only want to know if it was in the word or not for a later check to see if it was in all the words.  
I do this by checking if the counter is equal to the list size - 1. -1 is because I already know it is included in the  
first word, so I don't check it again. This gives me a list of all the common letters. However, I need to remove  
duplicates, so I only add it to a common letter list if it's not already in it.  
  
Lastly, I check to see how many times that letter appears in that word and that's a simple for loop print statement.  
I use the same '.count()' on each word and print out the count to show how many times it appears.  
"""*list\_of\_strings = ["barack", "obar?ma", "war", "russia?", "mak?er"]  
list\_size = len(list\_of\_strings)  
  
  
def check\_for\_question\_mark(list\_strings):  
 for i in range(list\_size):  
 if list\_strings[i].count("?") > 0:  
 print(f'{list\_strings[i]} does contain a "?"')  
  
  
def common\_letters(list\_strings):  
 common\_chars = []  
 for char in list\_strings[0]: # Iterate through characters in first word  
 matching\_char\_count = 0  
 for i in range(1, list\_size): # Iterate through words in list and check if it contains character  
 if list\_strings[i].count(char) > 0:  
 matching\_char\_count = matching\_char\_count + 1  
 continue  
 else:  
 break  
  
 if matching\_char\_count >= list\_size - 1:  
 if common\_chars.count(char) == 0: # Does not add unique char if it is already in list  
 common\_chars.append(char)  
  
 for letter in range(len(common\_chars)):  
 print(f'\nCharacter {common\_chars[letter]} appears in all items')  
 for i in range(list\_size):  
 print(f'{list\_strings[i]} contains "{common\_chars[letter]}" '  
 f'{list\_strings[i].count(common\_chars[letter])} time(s)')  
  
  
def main():  
 check\_for\_question\_mark(list\_of\_strings)  
 common\_letters(list\_of\_strings)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# Task 2

*"""  
Cian Herlihy - R00205604 - Task 2  
  
I started off by getting the names of the players off the user. I could have done this is main() but it is cleaner  
to get this in its own separate function and then return the list. This was not required, but I think it is an  
improvement. I then ask how many players they would like to eject ranging from 1-6. This then gets input as a parameter  
for the list\_manipulate() function along with the list, so I can then start interacting with the list. I start by making  
a copy, so I can see visually at the end that the players got removed and show the old list intact before any removal.  
I then just do a simple for while loop to loop the amount of time the person wanted players ejected. I return a new list  
with this function and handle the printing in main since it is not that much printing to need its own function.  
  
In the print statements I used the keyword tuple() to print the list as a tuple but this can be done earlier in the  
function if needed.  
"""*import random  
import time  
  
  
def creating\_list():  
 all\_players = []  
 i = 0  
 for i in range(12):  
 player = input(f'Player {i + 1} Name: ')  
 all\_players.append(player)  
 i += 1  
 return all\_players  
  
  
def list\_manipulate(all\_players, amount):  
 new\_players = all\_players.copy()  
 i = 0  
 while i < amount:  
 player\_name = random.choice(new\_players)  
 new\_players.remove(player\_name)  
 print(f'Step Forth {player\_name}')  
 time.sleep(30)  
 i += 1  
 return new\_players  
  
  
def main():  
 print("Task 2")  
 print("")  
 all\_players = creating\_list()  
 result = int(input("How many players would you like to Remove? (1-6) >>> "))  
 new\_players = list\_manipulate(all\_players, result)  
 print(f'{"="\*60}')  
 print(f'New List: {tuple(new\_players)}')  
 print(f'Old List: {all\_players}')  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# Task 3

*"""  
Cian Herlihy - R00205604 - Task 3  
  
For Task 3 I needed to get an integer off the user and halve it if it is an Even number and times 3 + 1 if it is odd.  
So to accomplish this I set up the converge() function to take in the int from the user and then determine if it is  
Even, then I should divide by 2. I used modulus (%) for the mathematics to check if it is even. The only other option  
then is odd so the else statement handles the multiplication by 3 and adding 1 to it. That is all the maths required   
for this function.  
  
To make the program run in a loop until it reaches 1. Then I put it in a while loop. I simply put everything in a   
while loop for the exception handling to loop you back to input again and then another nested while loop to do the   
recursive function. I made the program sleep for 1/2 a second everytime it uses the function so it slows the program   
down to see the effect it has on the input number. This is not needed but more aesthetically pleasing in my opinion.  
  
Exception handling includes handling of a ValueError which is a sting or float instead of an int for example. I have  
an AssertionError to determine if the integer input is greater than 1. This is because the program would not need to  
begin if the input was the number 1 since that is the end goal of the program. This makes the lowest possible  
int to be 2 and it uses the function exactly once before exiting.   
  
I use sys.exit() to exit the program then when it has reached 1. However, it can be implemented to be an option to   
start the loop again if I wanted to allow multiple inputs at the users discretion. This would be simple by changing  
the while loop to take a variable and only continue if they select the right option like in a menu.  
"""*import sys  
import time  
  
  
def converge(recursive\_int):  
 if recursive\_int % 2 == 0:  
 recursive\_int = recursive\_int / 2  
 print(int(recursive\_int))  
 return int(recursive\_int)  
 else:  
 recursive\_int = (recursive\_int \* 3) + 1  
 print(int(recursive\_int))  
 return int(recursive\_int)  
  
  
def main():  
 print("Task 3")  
 print(f'{"=" \* 30}')  
 while True:  
 try:  
 print("")  
 print("Even numbers are halved. Odd numbers are x3 and +1.")  
 recursive\_int = int(input("Enter an Integer >>> "))  
 assert recursive\_int > 1  
  
 while True:  
 time.sleep(0.5)  
 recursive\_int = converge(recursive\_int)  
 if recursive\_int != 1:  
 continue  
 else:  
 sys.exit()  
  
 except AssertionError:  
 print("Input number was too low.")  
 except ValueError:  
 print("Please Enter an Integer")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# Task 4

*"""  
Cian Herlihy - R00205604 - Task 3  
  
I started with having a constant of the parent directory that can be changed to a location of your choosing and having  
it as a constant makes it easier to manage. I then created some strings to insert into text files just to provide  
content rather than blank files.  
  
I get the input off the user of the folder name and then use os.path.join() to make it 1 complete path to use.  
I create the folder and then create 2 new folders using the os.mkdir() command. I did this in a separate function  
to improve readability of the code. I also check if the folder exists first by using shutil.rmtree().  
If it exists I delete the directory with all its contents and then remake the directories. I make folders in folders  
as requested and then finally enter the directory 'docs' and create my files there. I use another function to create  
these files by using a file writing connection. I use 'w' to indicate that it will be overwritten (Not that it matters).  
I input the content I stated above into each file and then close each connection respectively. I create 2 more Directories  
in docs and then return to my original function create\_folder(). This function then returns to main to keep the  
hierarchy of the functions.  
  
My next step was to change the files names from uppercase to lower case. The .lower() function replaces the original  
text therefore I used os.replace() to replace the old file name with the new name. This does not affect the content in  
the file. Before I do all this though I created time.sleep() so you can visually see in the folder the file names  
changing if you like. This is not needed, but I find it useful for testing purposes.  
"""*import os  
import shutil  
import time  
  
PARENT\_DIR = "C:/Users/Cian/OneDrive - mycit.ie/Sys\_Scripting/Python Assignment/"  
coronaText = "This is CORONAVIRUS.txt"  
dangerText = "This is DANGEROUS.txt"  
keepsText = "This is KEEPSAFE.txt"  
stayhomeText = "This is STAYHOME.txt"  
hygieneText = "This is HYGIENE.txt"  
  
  
def create\_folder(folder\_path):  
 backup = os.path.join(folder\_path, "backup")  
 working = os.path.join(folder\_path, "working")  
 if os.path.exists(folder\_path):  
 shutil.rmtree(folder\_path)  
 os.mkdir(folder\_path)  
 print("Folder Deleted and then Re-Created")  
 else:  
 os.mkdir(folder\_path)  
 print("Created Folder")  
  
 os.mkdir(backup)  
 os.mkdir(working)  
  
 # Working Directory  
 workingSubDirs = ["pics", "docs", "movie"]  
 for folders in workingSubDirs:  
 os.makedirs(os.path.join(working, folders))  
  
 docs\_folder = os.path.join(working, "docs")  
 write\_files\_to\_docs(docs\_folder)  
 return docs\_folder  
  
  
def write\_files\_to\_docs(docs\_folder):  
 # Write to Coronavirus.txt  
 c = open(os.path.join(docs\_folder, "CORONAVIRUS.txt"), mode="w")  
 c.write(coronaText)  
 c.close  
 # Write to Dangerous.txt  
 d = open(os.path.join(docs\_folder, "DANGEROUS.txt"), mode="w")  
 d.write(dangerText)  
 d.close  
 # Write to KeepSafe.txt  
 k = open(os.path.join(docs\_folder, "KEEPSAFE.txt"), mode="w")  
 k.write(keepsText)  
 k.close  
 # Write to StayHome.txt  
 s = open(os.path.join(docs\_folder, "STAYHOME.txt"), mode="w")  
 s.write(stayhomeText)  
 s.close  
 # Write to Hygiene.txt  
 h = open(os.path.join(docs\_folder, "HYGIENE.txt"), mode="w")  
 h.write(hygieneText)  
 h.close  
  
 # docs Directory  
 docsSubDirs = ["school", "party"]  
 for folders in docsSubDirs:  
 os.makedirs(os.path.join(docs\_folder, folders))  
 return  
  
  
# This function is to change text files to lowercase  
def adjust\_capitalisation(docs\_folder):  
 for file in os.listdir(docs\_folder):  
 print(file)  
 os.rename(docs\_folder + "/" + file, docs\_folder + "/" + file.lower())  
  
  
def main():  
 print("Task 4")  
 folder\_name = input("Enter the folder name \n>>> ")  
 print("")  
 path = os.path.join(PARENT\_DIR, folder\_name)  
 docs\_dir = create\_folder(path)  
 time.sleep(5) # Delay second function to see effect in folders  
 adjust\_capitalisation(docs\_dir)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()