Python Belt Challenge

AUTHOR
Thomas Zwiller

–READ ME–

Hello Seth! This is my submission for the Python Belt Challenge.

My main goal was elegance. I orignally thought about using tkinter to make a GUI but figured keeping it simple was the main focus. So my code comprises of a few libraries, then a loop that calls two defined function.

Here's a brief run-through of what everything does.

File Namer: Pulls the names from the file by getting the ticker, the quarter and the year. It then concatenats them and then returns the fully attached name with a .csv

```
def file_namer(list_name: str):

  ticker = re.search(r"(?<=Calls/)[a-z]{3}", list_name)
  quarter = re.search(r"(?<=Calls/[a-z]{3}_).", list_name)
  year = re.search(r"(?<=Calls/[a-z]{3}_\w\d\_)....", list_name)

  ticker = ticker.group().upper()
  quarter = quarter.group().upper()
  year = year.group()

  name = ticker + "-" + quarter + "-" + year + ".csv"
  return name
```

Transcript Creator: This function needs a list to function.

Once it does, it opens the file, pulls in the contents and then begins to iterate through it, performing a series of checks.

The first check is if the the file has a newline ahead of it, a newline behind it, doesn't include a period and i is less than the length of the file

This then makes it the current speaker

The function then checks to make sure that the text includes a period. If it does, it becomes the current text and is sent out to the speaker list. If it doesn't have a period (meaning it's just a speaker) then it is skipped and proceeds to the next line.

Once this process is done for the file, the file is named, calling the file namer function.

The text is then converted to a dataframe, then written out as a .csv and the file is closed.

The function doesn't need to return anything.

```
def transcript_creator (file_to_clean: list):

  speaker_list = []
  file = open(file_to_clean, "r")
  file_text = file.readlines()
  i = 2

  while i < len(file_text):
    string_check = file_text[i]
    cleaned_string = string_check.strip()

    if (i < len(file_text) - 1) and (file_text[i - 1] == '\n' and file_text[i + 1] == '\n
      curr_speaker = cleaned_string

    if re.search(r'^(.*\.)$', cleaned_string):
      curr_text = cleaned_string
      speaker_list.append({'Speaker': curr_speaker, 'Text': curr_text})
    i += 1

  name = file_namer(file_to_clean)
  speaker_df = pd.DataFrame(speaker_list, columns = ['Speaker', 'Text'])
  speaker_df.to_csv(name, index=False)
  file.close()
  return None
```

And this is the actual body of the code.

The function are imported, and the path has to be manually set by the user.

The path then has a *.txt added so it can be input into glob.

The glob call is then fed to a loop, and the loop runs for the duration of the .txt files found in the folder.

```
import glob as glob, pandas as pd, re, platform, os

#Import path here:
files = '/Users/TomTheIntern/Desktop/Mendoza/Mod 3/Unstructured/Homework 1/Calls'

path = os.path.join(files, '*.txt')

calls = glob.glob(path)

i = 0

for transcript in calls:
  transcript_creator(transcript)
  i += 1
  print(f'File {i} created.')
```

```
print('Thank you, have a nice day.')
```

```
File 1 created.
File 2 created.
File 3 created.
File 4 created.
Thank you, have a nice day.
```