

# WEEK5 Assignment- Python File Input and Output

Q.1 Problem Statements: Create a python program for taking user notes in text file. When a user starts it up, it should prompt them for a filename. If they enter a file name that doesn't exist, it should prompt them to enter the text they want to write to the file. After they enter the text, it should save the file and exit. If they enter a file name that already exists, it should ask the user if they want: A) Read the file B) Append the file C) Delete the file and start over

If the user wants to read the file it should simply show the contents of the file on the screen. If the user wants to start over then the file should be deleted and another empty one made in its place. If a user elects to append the file, then they should be able to enter more text, and that text should be added to the existing text in the file.

Note: The software Development Team created partial python code for the above problem statement, you try to help them to complete the code wherever required and produce successful output

In [2]:

```
import os
filename=input("Enter The File Name : ")
# Here we check if the file exists
if os.path.isfile('./'+filename+".txt"):
    print(f"Looking for file:{filename}.txt")
    print("Found it")
    action = input("Enter the possible file operation: read, append, delete, replace\n")

    if action=="read":
        file=open(filename+".txt",'r')
        f1=file.read()
        print(f1)
        """To Do : write the code for file read operation"""

    elif action=="append":
        file=open(filename+".txt","a")
        cnt=input("write:")
        file.write(cnt+"\n")
        file.close()
        """To Do : write the code for file append operation"""

    elif action == "delete":
        os.remove(filename+".txt")
        """To Do : write the code for file delete and another empty one made in its place o
```

```
elif action == "replace":
    file=open(filename+".txt","w")
    file.write("")
    file.close()

else:
    print("Sorry, unrecognized action 😞")

else:
    print("Nope, this file does not exist, I'm going to create it for you! 😊")
    file=open(filename+".txt","w")
    file.write("")
    file.close()
    """To Do : write the code for file create and write the notes """
```

Enter The File Name : Messi

Looking for file:Messi.txt

Found it

Enter the possible file operation: read, append, delete, replace

read

G.O.A.T

## Q2.Text File Analytics:

Q2.1 Assume you as Data Analyst, you need to collect all text file name from current working directory and store the all text file name into "txtfile\_namelist.txt" and read this file and print How many files are available in that namelist.

In [14]:

```
import os
data=open("txtfile_namelist.txt","w+")
for file in os.listdir():
    if file.endswith(".txt"):
        print(file)
        data.write(file + "\n")
```

```
b.txt
bala.txt
file2.txt
Messi.txt
txtfile_namelist.txt
zenofpython.txt
```

In [15]:

```
num_files = sum(1 for line in open('txtfile_namelist.txt'))  
print("How many files are available in that namelist :", num_files)
```

How many files are available in that namelist : 6

Q2.2 write the following contents to zenofpython.txt content: The Zen of Python, by Tim Peters Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated. Flat is better than nested. Sparse is better than dense. Readability counts. Special cases aren't special enough to break the rules. Although practicality beats purity. Errors should never pass silently. Unless explicitly silenced. In the face of ambiguity, refuse the temptation to guess. There should be one and preferably only one obvious way to do it. Although that way may not be obvious at first unless you're Dutch. Now is better than never. Although never is often better than right now. If the implementation is hard to explain, it's a bad idea. If the implementation is easy to explain, it may be a good idea. Namespaces are one honking great idea let's do more of those!

an perform the following Operation: 1.print number of lines in the zenofpython.txt file 2.print number of words in the zenofpython.txt file 3.Print how may python keyword present in the zenofpython.txt file 4.print the all content in uppcase

In [9]:

```
import keyword
key=0
num_words=0
with open("zenofpython.txt","w+") as f2:
    f2.write("The Zen of Python, by Tim Peters\nBeautiful is better than ugly.\nExplicit is
    f2.seek(0)
    data=f2.read()
    f=data.replace("--", " ")
    lines = f.split()
    num_words += len(lines)
    num_lines=sum(1 for line in open('zenofpython.txt'))
    caps=data.upper()
    for j in lines:
        if j in keyword.kwlist:
            key+=1
f2.close()
print("Number of line           : ",num_lines)
print("Number of words         : ",num_words)
print("How may python keyword present : ",key)
print("The all content in uppercase : \n",caps)
```

```
Number of line           : 20
Number of words         : 143
How may python keyword present : 14
The all content in uppercase :
```

THE ZEN OF PYTHON, BY TIM PETERS  
BEAUTIFUL IS BETTER THAN UGLY.  
EXPLICIT IS BETTER THAN IMPLICIT.  
SIMPLE IS BETTER THAN COMPLEX.  
COMPLEX IS BETTER THAN COMPLICATED.  
FLAT IS BETTER THAN NESTED.  
SPARSE IS BETTER THAN DENSE.  
READABILITY COUNTS.  
SPECIAL CASES AREN'T SPECIAL ENOUGH TO BREAK THE RULES.  
ALTHOUGH PRACTICALITY BEATS PURITY.  
ERRORS SHOULD NEVER PASS SILENTLY.  
UNLESS EXPLICITLY SILENCED.  
IN THE FACE OF AMBIGUITY, REFUSE THE TEMPTATION TO GUESS.  
THERE SHOULD BE ONE -- AND PREFERABLY ONLY ONE -- OBVIOUS WAY TO DO IT.  
ALTHOUGH THAT WAY MAY NOT BE OBVIOUS AT FIRST UNLESS YOU'RE DUTCH.  
NOW IS BETTER THAN NEVER  
ALTHOUGH NEVER IS OFTEN BETTER THAN RIGHT NOW.  
IF THE IMPLEMENTATION IS HARD TO EXPLAIN, IT'S A BAD IDEA.  
IF THE IMPLEMENTATION IS EASY TO EXPLAIN, IT MAY BE A GOOD IDEA.  
NAMESPACES ARE ONE HONKING GREAT IDEA -- LET'S DO MORE OF THOSE!

In [ ]:

