#part 1 # 1) Explain the concept of file modes in Python, Provide a brief description of at least three file modes commonly used in file handling, How does the choice of mode affect the way a file is opened and processed?

The 3 writing modes are read,write and append. Read reads contents of the file, write overwrites contents in a file, and append appends data to the end of the file rather then overwriting it.

 # 2) Discuss the difference between writing and appending to a file in Python, When would you choose one over the other, Provide an example scenario where appending to a file is more suitable than writing to it?

Writing overwrites any existing content in a file, whilst appending appends to it. Appending would be suitable when you have to add on to some other content like say a list of names in a file, while writing could be suited when data doesnt have to be preserved across different timed program execution.

 # 3) Enumerate and explain three best practices for proper file handling in Python, Consider aspects such as opening and closing files, error handling, and resource management.

Proper closing of an opened file is necessary to help in resource management and more improtantly to protect the data, Until a file is closed there is a file descriptor that is open which can be used to view contents of a root protected file. Proper Error handling is needed to make sure the file that is targeted, is found and opened or atleast the exception is handled smoothly.

 # 4) Discuss the potential consequences of not closing a file explicitly, How does Python handle file closing when using the with statement, and why is it considered a good practice in file handling?

A Not Closed file leaves a dangling file description that could be used to read contents of a root protect file. The with keyword convienently automatically closes the file after the code inside the block of code inside the with keyword is done, or throws an error. This is a good practice as the file is closed on either the success or faliure of the operation.

#Part 2 # 1) Create a program in Python that opens a file named 'datafile_txt' for reading and assigns identifier input_file to the file object created with open('datafile.

```
txt','r') as input_file:
    for line in input_file:
        print(line.strip())


LasLas
```

 # 2) Create a program in Python that opens a file named 'datafile2_txt' for writing and assigns identifier output_file to the file object created.

```
output_file = open('datafile.txt','w')
output_file.write("PewPew")
output_file.close()
```

 # 3) Assume that input_file is a file object for a text file open for reading, and output_file is a file object for a text file open for writing, Explain the contents of the output after the following code terminates:

```
# This writes the contents from input file to the output file.
```

 # 4) Identify the error in the following code:

```
# Indentation levels are wrong.
```

#Part 3 # 1) Write a Python function called reduce_spaces that is given a line read from a text file and returns the line with all extra space characters removed.

```
def reduce_spaces(line):
    return line.strip()
print(reduce_spaces("  \n pew pew \n"))


pew pew
```

 # 2) Write a Python function named extract_temp that is given a line read from a text file and displays the one number (integer) found in the string:

```
import re
def extract_temp(line):
    match = re.search(r'\d+',line)
    print(match.group())
extract_temp("The high today will be 75 degrees")


75
```

 # 3) Write a Python function named check_quotes that is given a line read from a text file and returns True if each quote character in the line has a matching quote (of the same type), otherwise returns False.

```
def check_quotes(line):
    dbquotes = (line.count("\""))
    squotes = (line.count("\'"))
    if squotes % 2 == 0 or dbquotes % 2 == 0 and squotes != 0 and dbquotes !=0 :
        print("Match")
    else:
        print("Doesnt Match")
check_quotes("\' MEOW \'")


Match
```

 # 4) Write a Python function named count_letters that is given a line read from a text file and returns a list containing every letter in the line and the number of times that each letter appears (with upper/lower case letters counted together).

```
def count_letters(string):
    string = string.lower()
    z = list(set(map(lambda x:(x,string.count(x)),string)))
    for i in z:
        if i[0] != ' ':
            print(i)
count_letters("This is a line")


('a', 1)
```

```
('h', 1)
('e', 1)
('i', 3)
('l', 1)
('t', 1)
('n', 1)
('s', 2)
```

 # 5) Write a Python function named interleave_chars that is given two lines read from a text, and returns a single string containing the characters of each string interleaved:

```
def interleave_chars(str1,str2):
    longer,shorter = (str1,str2) if len(str1) > len(str2) else (str2,str1)
    interweaved = ""
    print(longer,shorter)
    for index,val in enumerate(shorter):
        interweaved += (val+longer[index])
    print(interweaved+longer[len(shorter):])
interleave_chars("Hello","Goodbye")
```

```
Goodbye Hello
HGeololdobye
```

 # 6) Give a for loop that counts all the characters in a string assigned to a variable line, except blanks and the newline character.

```
donts = [" ","\n"]
line = "meow meow meow"
z = sum(1 for i in line if i not in donts)
print(z)
```

```
12
```

 # 7) For a variable month which contains the full name of any given month, give an expression to display just the first three letters of the month.

```
month = "January"
print(month[:3])
```

```
Jan
```

 # 8) Give an expression that displays True if the letter ‘r’ appears in a given month name stored in a variable month, otherwise displays False.

```
monthname = "January"
print("True" if "r" in monthname else "False")


True
```

 # 9) Give an expression for determining how many times the letter ‘r’ appears in a given month name stored in a variable month.

```
monthname = "Januarrry"
print(sum(1 for i in monthname if "r" == i ))


3
```

 # 10) For a person’s first name stored in variable first_name, and last name stored in variable last_name, give an expression that displays the person’s name formatted exactly as follows, Jones, William.

```
first_name,last_name = "Jhon","Jones"
print(f"{last_name},{first_name}")

Jones,Jhon
```

 # 11) Give an instruction that determines if a given social security number represented as a string and stored in variable ss_num, contains any non- digits.

```
import re
ss_num="00000123"
print("Contains non digits") if re.search(r'\D',ss_num) else print("Contains only digits")

Contains only digits
```

 # 12) Give an instruction that determines the index of the '@' character in an email address stored in variable email_addr.

```
email = "meowmeow@gmail.com"
print(email.index('@'))


8
```

 # 13) For a variable named date containing a date in the form 12/14/2012, give an expression that replaces all slashes characters with dashes.

```
date = "12/14/2012"
print(date.replace("/","-"))

12-14-2012
```

 # 14) For a variable named err_mesg that contains error messages in the form ** error message **, give an expression that produces a string containing the error message without the leading and trailing asterisks and blank characters.

```
err_msg = "** error message **"
print(err_msg.replace("*","").strip())

error message
```

 # 15) Write a Python program that encrypts and decrypts text files using a substitution cipher, Your program should ask the user for the name of a text file and whether they would like to encrypt or decrypt, Once the process is complete, you should write the output to a new text file with a modified name:

```
import string
import os

def print_file(filename):
    with open(filename,"r") as file:
        print(file.read())
```

```
def substution_cipher():
    all_letters = string.ascii_letters
    mode = input("Enter e to encrypt or d to decrypt: ")
    filename = input("Enter filename: ")
    try:
        if mode == "e":
            with open(filename,'r') as input_file:
                sum_string = input_file.read().strip()
                encrypted_string= ''.join([all_letters[all_letters.index(x)+4] if x not in [" ","\"",
"'","\n"] and not x.isdigit() else x for x in sum_string ])
                with open("temp.txt","w") as temp_file:
                    temp_file.write(encrypted_string)
            os.replace("temp.txt",filename)
            print_file(filename)
        elif mode == "d":
            with open(filename,'r') as input_file:
                sum_string = input_file.read().strip()
                decrypted_string= ''.join([all_letters[all_letters.index(x)-4] if x not in [" ","\"",
"'","\n"] and not x.isdigit() else x for x in sum_string ])
                with open("temp.txt","w") as temp_file:
                    temp_file.write(decrypted_string)
            os.replace("temp.txt",filename)
            print_file(filename)
        else:
            print("Incorrect Mode ... Terminating")
    except FileNotFoundError:
        print(f" File {filename} not found, make sure you have an existing file.")
    except Exception as e:
        print(f"Some error has occured {e}")
substution_cipher()


Enter e to encrypt or d to decrypt: e
Enter filename: datafile.txt
TiATiA
```
