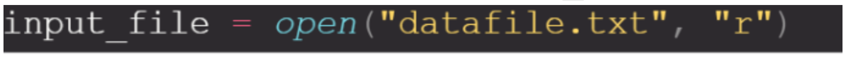
4CS001

Python Workshop 11: File Handling

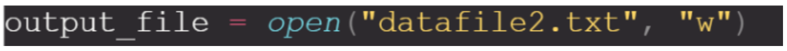
Part 1

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.



```python
input_file = open("datafile.txt", "r")
```

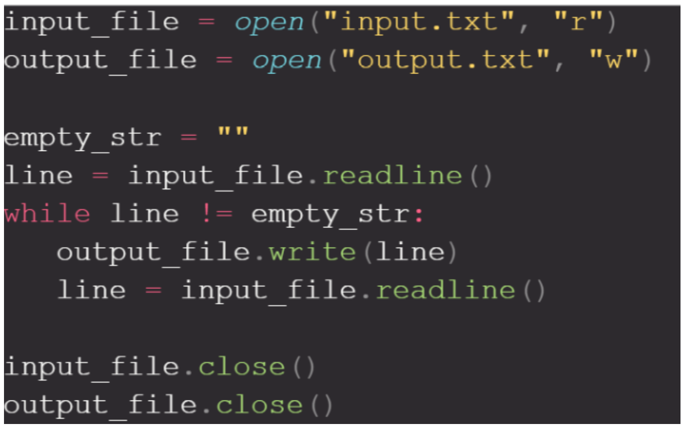
2. Create a program in Python that opens a file named `'datafile2.txt'` for writing and assigns identifier `output_file` to the file object created.



```python
output_file = open("datafile2.txt", "w")
```

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:

```python
empty_str = ''
line = input_file.readline()
while line != empty_str:
    output_file.write(line + '\n'
        line = input file.readline()
```

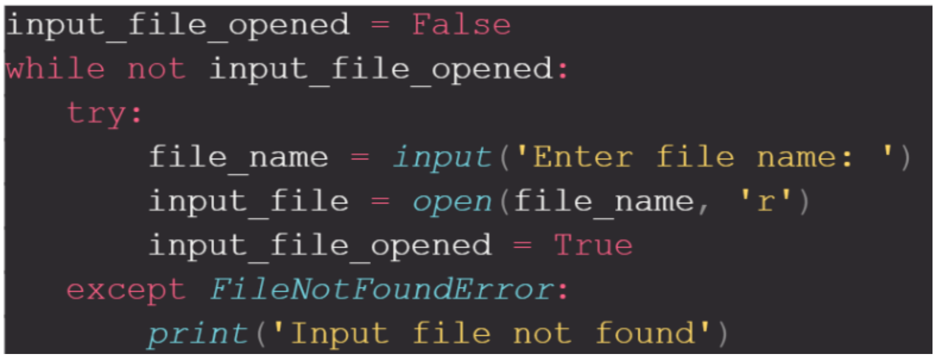


```python
input_file = open("input.txt", "r")
output_file = open("output.txt", "w")

empty_str = ""
line = input_file.readline()
while line != empty_str:
   output_file.write(line)
   line = input_file.readline()

input_file.close()
output_file.close()
```

The code opens two files, "input_file" for reading and "output_file" for writing. It uses a while loop to read the lines from "input_file" using the readline() method until an empty string is returned, indicating the end of the file. Each line read from "input_file" is written to "output_file" with a newline character (\n) using the write() method. Once the code is finished, "output_file" will contain a duplicate of "input_file" with lines separated by newline characters.

4. Identify the error in the following code:

```
input_file_opened = False
while not input_file_opened:
try:
     file_name = input('Enter file name: ')
     input_file = open(file_name, 'r')
     input_file_opened = True
except: print('Input file not found')
```

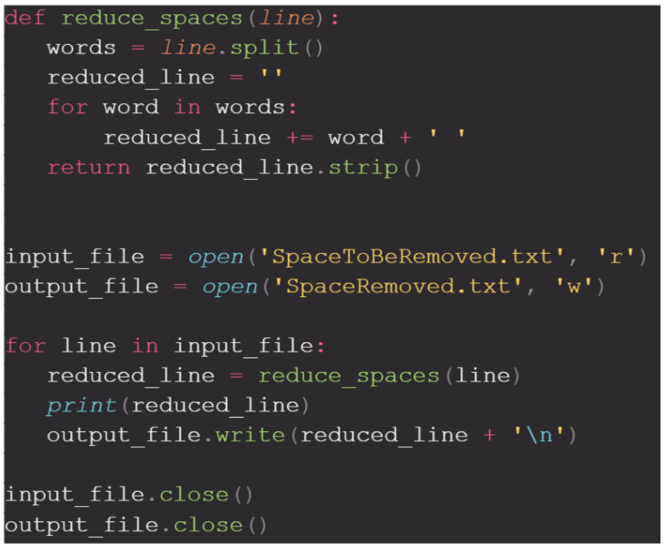


```
input_file_opened = False
while not input_file_opened:
   try:
       file_name = input('Enter file name: ')
       input_file = open(file_name, 'r')
       input_file_opened = True
   except FileNotFoundError:
       print('Input file not found')
```

The code has problems with its indentation. The try and except blocks need to be indented to match the while loop. Also, the print statement in the except block should be indented and on a new line. The exception handling is too general and does not specify what type of exception should be caught.

<u>Part 2</u>

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:

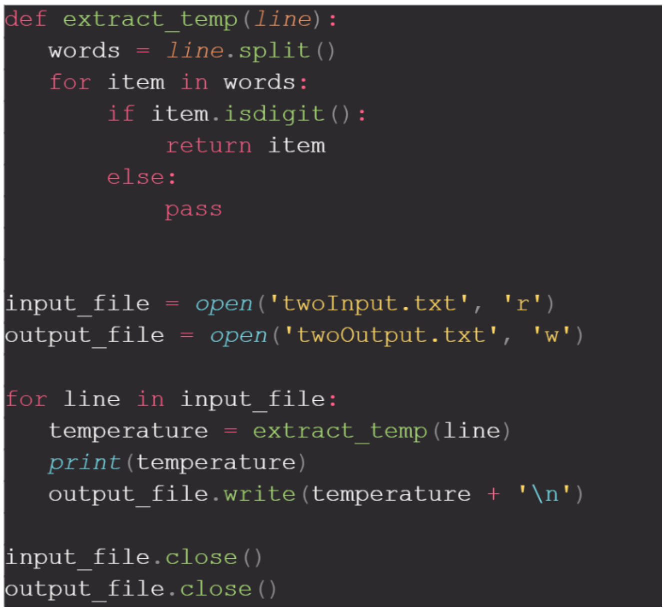


```
def reduce_spaces(line):
   words = line.split()
   reduced_line = ''
   for word in words:
       reduced_line += word + ' '
   return reduced_line.strip()


input_file = open('SpaceToBeRemoved.txt', 'r')
output_file = open('SpaceRemoved.txt', 'w')

for line in input_file:
   reduced_line = reduce_spaces(line)
   print(reduced_line)
   output_file.write(reduced_line + '\n')

input_file.close()
output_file.close()
```

‘This line has extra space characters’ → ’This line has extra space characters’

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:

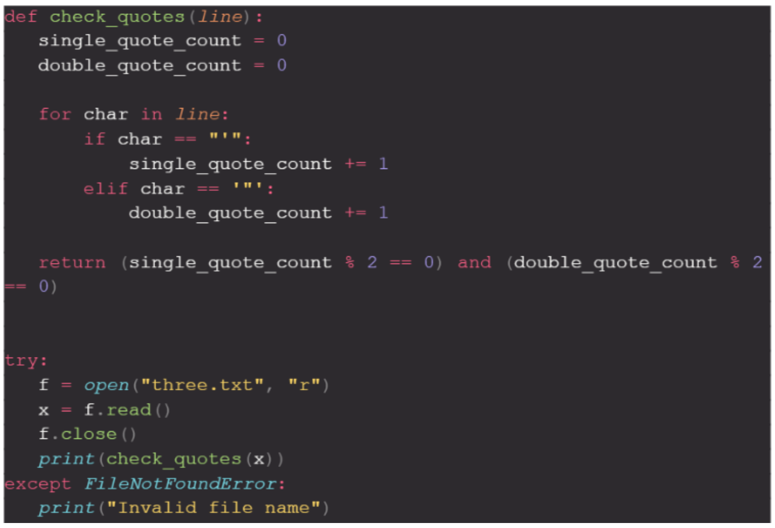
’The high today will be 75 degrees’ → 75.



```python
def extract_temp(line):
   words = line.split()
   for item in words:
       if item.isdigit():
           return item
       else:
           pass


input_file = open('twoInput.txt', 'r')
output_file = open('twoOutput.txt', 'w')

for line in input_file:
   temperature = extract_temp(line)
   print(temperature)
   output_file.write(temperature + '\n')

input_file.close()
output_file.close()
```

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

‘Today’s high temperature will be 75 degrees’ → False

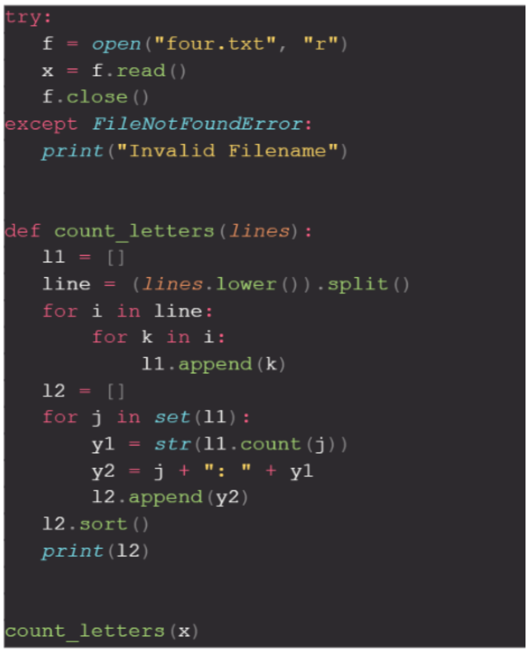


```python
def check_quotes(line):
   single_quote_count = 0
   double_quote_count = 0

   for char in line:
       if char == "'":
           single_quote_count += 1
       elif char == '"':
           double_quote_count += 1

   return (single_quote_count % 2 == 0) and (double_quote_count % 2
== 0)


try:
   f = open("three.txt", "r")
   x = f.read()
   f.close()
   print(check_quotes(x))
except FileNotFoundError:
   print("Invalid file name")
```

4. Write a Python function named `count_letters` that is given a line read

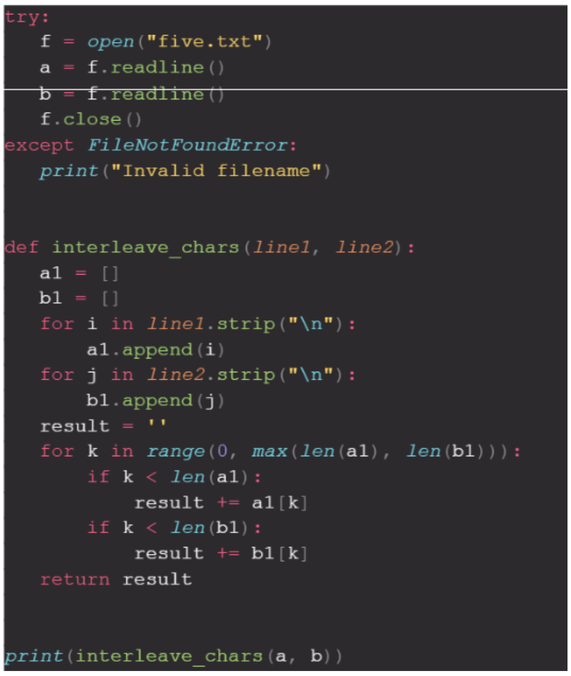
from a text file and returns a list containing every letter in the line and the number of times that each letter appears (with upper/lower case letters counted together)

‘This is a line’ → [(‘t’, 1), (‘h’, 1), (‘i’, 3), (‘s’, 2), (‘a’, 1), (‘l’, 1), (‘n’, 1), (‘e’, 1)]



```python
try:
   f = open("four.txt", "r")
   x = f.read()
   f.close()
except FileNotFoundError:
   print("Invalid Filename")


def count_letters(lines):
   l1 = []
   line = (lines.lower()).split()
   for i in line:
       for k in i:
           l1.append(k)
   l2 = []
   for j in set(l1):
       y1 = str(l1.count(j))
       y2 = j + ": " + y1
       l2.append(y2)
   l2.sort()
   print(l2)


count_letters(x)
```

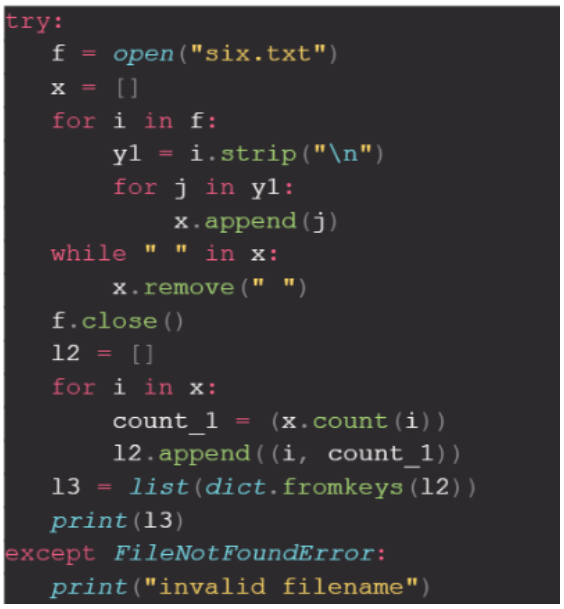
5. Write a Python function named `interleave_chars` that is given two lines read from a text, and returns a single string containing the characters of each string interleaved: ‘Hello’, ‘Goodbye’ → ‘HGeololdobye’



```python
try:
   f = open("five.txt")
   a = f.readline()
   b = f.readline()
   f.close()
except FileNotFoundError:
   print("Invalid filename")


def interleave_chars(line1, line2):
   a1 = []
   b1 = []
   for i in line1.strip("\n"):
       a1.append(i)
   for j in line2.strip("\n"):
       b1.append(j)
   result = ''
   for k in range(0, max(len(a1), len(b1))):
       if k < len(a1):
           result += a1[k]
       if k < len(b1):
           result += b1[k]
   return result


print(interleave_chars(a, b))
```

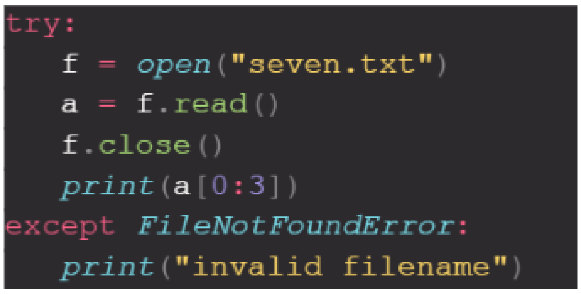
6. Give a for loop that counts all the characters in a string assigned to variable

`line`, except blanks and the newline character.



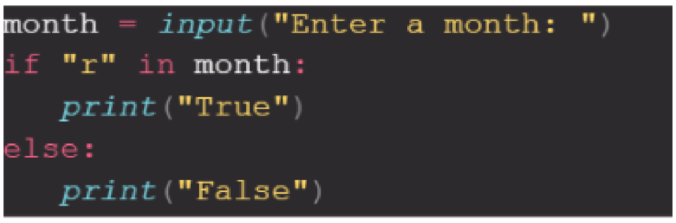
```
try:
   f = open("six.txt")
   x = []
   for i in f:
       y1 = i.strip("\n")
       for j in y1:
           x.append(j)
   while " " in x:
       x.remove(" ")
   f.close()
   l2 = []
   for i in x:
       count_1 = (x.count(i))
       l2.append((i, count_1))
   l3 = list(dict.fromkeys(l2))
   print(l3)
except FileNotFoundError:
   print("invalid filename")
```

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



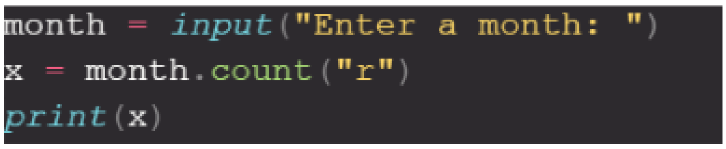
```
try:
   f = open("seven.txt")
   a = f.read()
   f.close()
   print(a[0:3])
except FileNotFoundError:
   print("invalid filename")
```

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



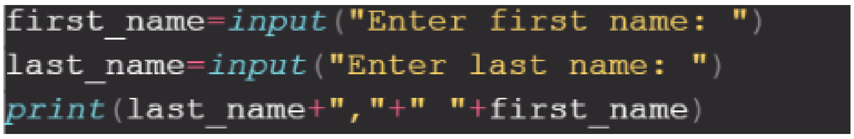
```
month = input("Enter a month: ")
if "r" in month:
   print("True")
else:
   print("False")
```

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



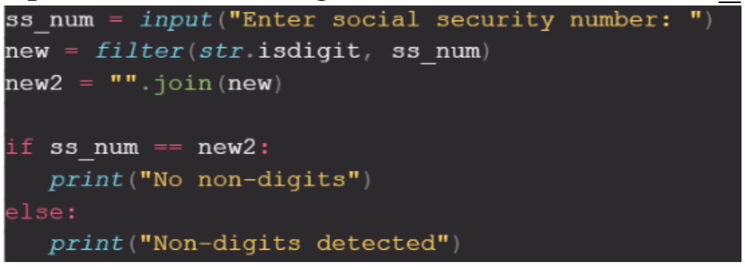
```
month = input("Enter a month: ")
x = month.count("r")
print(x)
```

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`.



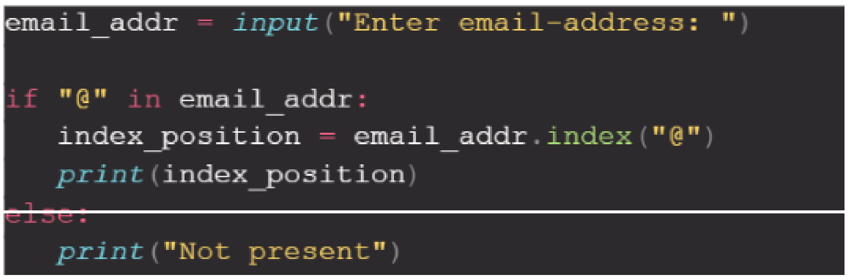
```
first_name=input("Enter first name: ")
last_name=input("Enter last name: ")
print(last_name+","+" "+first_name)
```

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.



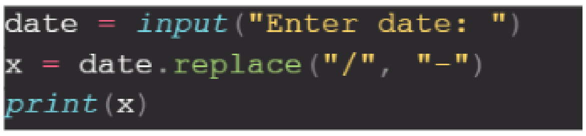
```
ss_num = input("Enter social security number: ")
new = filter(str.isdigit, ss_num)
new2 = "".join(new)

if ss_num == new2:
   print("No non-digits")
else:
   print("Non-digits detected")
```

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



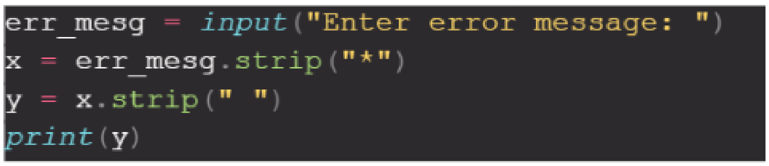
```
email_addr = input("Enter email-address: ")

if "@" in email_addr:
   index_position = email_addr.index("@")
   print(index_position)
else:
   print("Not present")
```

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 = input("Enter date: ")
x = date.replace("/", "-")
print(x)
```

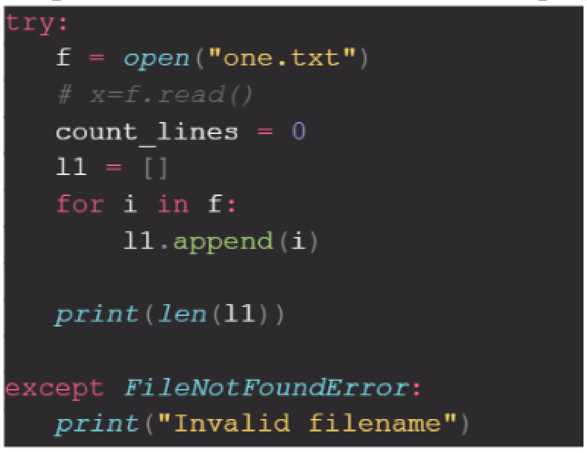
14. For a variable named `err_mesg` that contains error messages in the form ** error message **, give an expression that produces a string containing the error message without the leading and trailing asterisks and blank characters.



```
err_mesg = input("Enter error message: ")
x = err_mesg.strip("*")
y = x.strip(" ")
print(y)
```

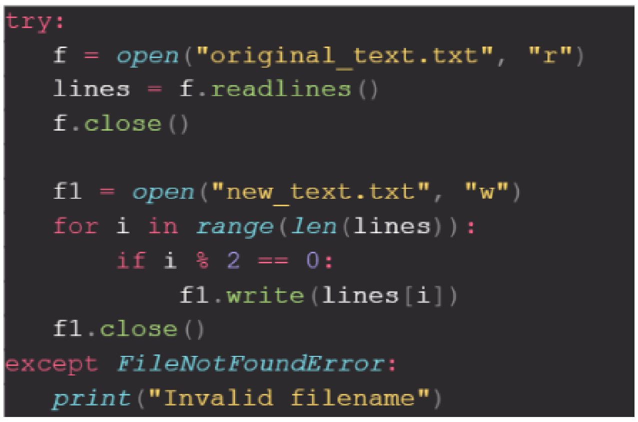
Part 3

1. Write a program that opens and reads a text file and displays how many lines of text are in the file.



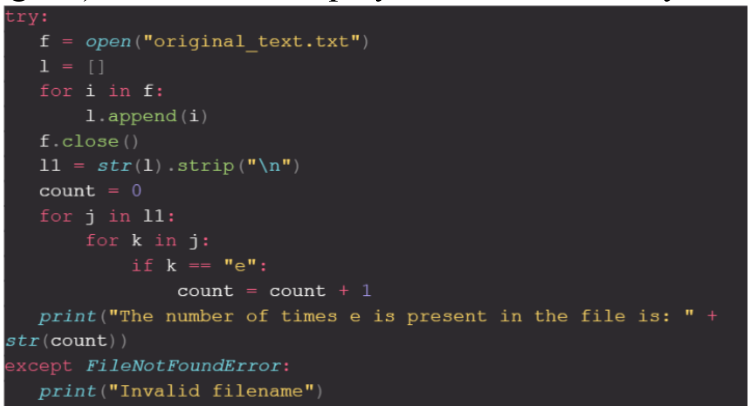
```
try:
    f = open("one.txt")
    # x=f.read()
    count_lines = 0
    l1 = []
    for i in f:
        l1.append(i)

    print(len(l1))

except FileNotFoundError:
    print("Invalid filename")
```

2. Write a program that reads a text file named `original_text`, and writes every other line, starting with the first line, to a new file named `new_text`.



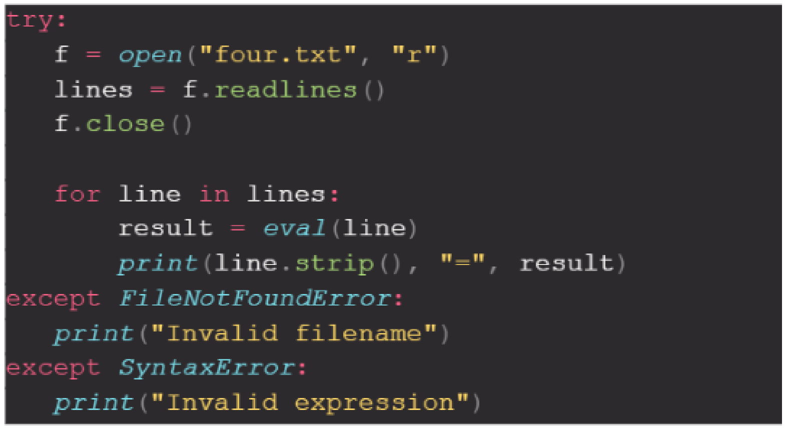
```
try:
    f = open("original_text.txt", "r")
    lines = f.readlines()
    f.close()

    f1 = open("new_text.txt", "w")
    for i in range(len(lines)):
        if i % 2 == 0:
            f1.write(lines[i])
    f1.close()
except FileNotFoundError:
    print("Invalid filename")
```

3. Write a program that reads a text file named `original_text`, and counts how many time the letter 'e' occurs (the most frequently occurring letter in English), and displays how many occurrences there are.



```
try:
    f = open("original_text.txt")
    l = []
    for i in f:
        l.append(i)
    f.close()
    l1 = str(l).strip("\n")
    count = 0
    for j in l1:
        for k in j:
            if k == "e":
                count = count + 1
    print("The number of times e is present in the file is: " +
str(count))
except FileNotFoundError:
    print("Invalid filename")
```

4. Write a program that reads a text file containing numerical expressions on each line and print them out along with the results. For example, for the numerical expression `4 + 2` in your file, your program should output: `4 + 2 = 6`.



```python
try:
   f = open("four.txt", "r")
   lines = f.readlines()
   f.close()

   for line in lines:
       result = eval(line)
       print(line.strip(), "=", result)
except FileNotFoundError:
   print("Invalid filename")
except SyntaxError:
   print("Invalid expression")
```

Part 4 (Optional)

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:

```
This program will encrypt and decrypt text files

Enter (e) to encrypt a password, and (d) to decrypt:
e
Enter the name of a text file to encrypt: hello.txt

Output written to: encrypted_hello.txt
```

Your program should catch exceptions and print helpful error messages. You should use your solution to Coding Challenge 03 to help you.