AKTU B.Tech II-Year

GATEWAY CLASSES

ONE SHOT Revision

Python Programming

Common to all Branches

Unit-4

Python File Operations

Pragya ma'am

PYTHON PROGRAMMING

AKTU UNIT-3 Syllabus

Python File Operations: Reading files, Writing files in python, Understanding read functions, read(), readline(), readlines(). Understanding write functions, write() and writelines() Manipulating file pointer using seek Programming, using file operations.

Topic wise All PYQs Covered

What are files? How they are useful?

- A file is a container in computer storage devices used for storing data.
- When a program is terminated, the entire data is lost.
- Storing in a file will preserve your data even if the program terminates.
- If you have to enter a large number of data, it will take a lot of time to enter them all.
- However, if you have a file containing all the data, you can easily access the contents of the file using a few commands .
- Advantages of File Handling in Python

1.Versatility: File handling in Python allows you to perform a wide range of operations, such as creating, reading, writing, appending, renaming, and deleting files.

2. Flexibility: File handling in Python is highly flexible, as it allows you to work with different file types (e.g. text files, binary files, CSV files etc.), and to perform different operations on files (e.g. read, write, append, etc.).

CSV files are plain text files that store tabular data in a simple format where each line represents a row, and values within the row are separated by commas.

File handling

3.User–friendly: Python provides a user-friendly interface for file handling, making it easy to create, read, and manipulate files

Disadvantages of File Handling in Python

1.Error-prone: File handling operations in Python can be prone to errors, especially if the code is not carefully written or if there are issues with the file system (e.g. file permissions, file locks, etc.).

2.Security risks: File handling in Python can also pose security risks, especially if the program accepts user input that can be used to access or modify sensitive files on the system.

3. Complexity: File handling in Python can be complex, especially when working with more advanced file formats or operations.

Text file	*Binary file*
Store information in ASCII Or Unicode character	Store information in the form of 0 and 1
Can store only plain text	can store different type of data audio, image, text in a single file
Each line end with using special character is EOL(END OF LINE) character which is the new line character ('\n') in python by default.	There is no delimiter for lines

File open

The Python open() function is used to open internally stored files.

It returns the contents of the file as Python objects.

Python open() Function Syntax

Syntax: open(file_name, mode)

Parameters:.

file name:, the name of the file that we want to open.

mode: This parameter is a string that is used to specify the mode in which the file is to be opened.

File opening modes

1.r: open an existing file for a read operation.

2 w: open an existing file for a write operation. If the file already contains some data, then it will be overridden but if the file is not present then it creates the file as well.

3.a: open an existing file for append operation. It won't override existing data

4. r+: To read and write data into the file. This mode does not override the existing data, but you can modify the data starting from the beginning of the file.

File opening modes

5 w+: To write and read data. It overwrites the previous file if one exists, it will truncate the file to zero length or create a file if it does not exist.

6.a+: To append and read data from the file. It won't override existing data.

7. rb : Open the file for reading in binary format. Raises an I/O error if the file does not exist.

8.rb+ Open the file for reading and writing in binary format. Raises an I/O error if the file does not exist

9. wb: Open the file for writing in binary format. Truncates the file if it already exists. Creates a new file if it does not exist

10. wb+ Open the file for reading and writing in binary format. Truncates the file if it already exists. Creates a new file if it does not exist.

11.ab Open the file for appending in binary format. Inserts data at the end of the file. Creates a new file if it does not exist.

12. ab+ Open the file for reading and appending in binary format. Inserts data at the end of the file. Creates a new file if it does not exist

file input-output(I/O) operations

open a file and open a file in read mode

main.py | myfile.txt

```
1 hello
2 welcome to gateway classes
```

main.py | myfile.txt

```
1  file1 = open("myfile.txt")
2
3  # Reading from file
4  print(file1.read())
5
6  file1.close()
7
8
9
```

```
hello
welcome to gateway classes
```

Writing to an Existing File in Python

main.py | myfile.txt

```
1 hello
2
```

main.py | myfile.txt

```
1  file1 = open("myfile.txt", "a")
2
3  # Writing to file
4  file1.write("\nWriting to file:)")
5
6  # Closing file
7  file1.close()
8
```

main.py | myfile.txt

```
1 hello
2 Writing to file:)
```

main.py | myfile.txt | file.txt | file1

```
1 file1=open("myfile.txt","a")
2 file1.write("bye")
3 file1.close()
```

main.py | myfile.txt | file.txt | file1

```
1 hello
```

main.py | myfile.txt | file.txt | file1

```
1 hellobye
```

Writing on a file

main.py | myfile.txt | test.txt

```
1 welcome to gatwayclasses
2 welcome to python classes
```

main.py | myfile.txt | test.txt

```
1 Hello, world!
```

main.py | myfile.txt | test.txt

```
1  # Open a file for writing and reading
2  file = open('test.txt', 'w+')
3
4  # Write some data to the file
5  file.write('Hello, world!')
6
7  #Move the file pointer back to the beginning of the file
8  file.seek(0)
9
10 # Read the data from the file
11 data = file.read()
12
13 # Print the data to the console
14 print(data)
15
16 # Close the file when you're done
17 file.close()
18
19
```

input

```
Hello, world!
```

Writing on a file

```
1 # Python code to create a file
2 file = open('myfile.txt','w')
3 file.write("This is the write command")
4 file.seek(0)
5 data=file.read()
6 print(data)
7 file.close()
8
```

```
Traceback (most recent call last):
  File "/home/main.py", line 5, in <module>
    data=file.read()
io.UnsupportedOperation: not readable
```

main.py | myfile.txt | file.txt | file1 | geek.txt

```
1 # Python code to create a file
2 file = open('myfile.txt','w+')
3 file.write("This is the write command")
4 file.seek(0)
5 data=file.read()
6 print(data)
7 file.close()
8
```

```
This is the write command
```

Difference between write and writelines()

main.py | myfile.txt | test.txt

```
1  file1 = open('myfile.txt', 'w')
2  L = ["This is Delhi \n", "This is Paris \n", "This is London \n"]
3  s = "Hello\n"
4  file1.write(s)
5
6  # Writing multiple strings
7  # at a time
8  file1.writelines(L)
9
10 # Closing file
11 file1.close()
12
13 # Checking if the data is
14 # written to file or not
15 file1 = open('myfile.txt', 'r')
16 print(file1.read())
17 file1.close()
18
```

input

```
Hello
This is Delhi
This is Paris
This is London
```

main.py | myfile.txt | test.txt

```
1 Hello
2 This is Delhi
3 This is Paris
4 This is London
5
```

Difference between write and writelines()

Writing to file

There are two ways to write in a file.

write() : Inserts the string str1 in a single line in the text file.

File_object.write(str1)

writelines() : For a list of string elements, each string is inserted in the text file. Used to insert multiple strings at a single time.

File_object.writelines(L) for L = [str1, str2, str3]

CLOSE A FILE

close() function closes the file and frees the memory space acquired by that file. It is used at the time when the file is no longer needed or if it is to be opened in a different file mode.

Syntax: File_object.close()

Difference between

Read()	readline()	readlines()
The read method reads the entire contents of a file and returns it as a string.	The readline method reads a single line from a file and returns it as a string. This means that if you use readline, you can read the contents of a file line by line, which can be useful for processing large files that do not fit in memory file = open("filename.txt", "r") file.readline()	the readlines method reads the entire contents of a file and returns it as a list of strings, where each element of the list is a single line of the file file = open("filename.txt","r") file.readlines()

main.py | example.txt | file.txt | file1

```
with open("example.txt","r")as file:
    line=file.readline()
    print(line)

```

```
hello student
```

main.py | example.txt | file.txt

```
hello student
welcome
to gatewayclasses
unit 4
file handling
```

main.py | example.txt | file.txt | file1 | geek.txt

```
with open("example.txt","r")as file:
    line=file.readlines()
    print(line)

```

input

```
['hello student \n', 'welcome\n', 'to gatewayclasses\n', 'unit 4\n', 'file handling']
```

main.py | example.txt | file.txt | file1

```
1 with open("example.txt","r")as file:
2     line=file.read()
3     print(line)
4
```

```
hello student
welcome
to gatewayclasses
unit 4
file handling
```

write a python program to read a file line by line store it into a list

main.py | example.txt

```python
def read_file_to_list(filename):
    lines = []
    with open(filename, 'r') as file:
        while True:
            line = file.readline()
            if not line:  # Check if line is empty (EOF)
                break
            lines.append(line.strip())  # Remove newline characters and add to the list
    return lines

# Example usage
filename = 'example.txt'
a = read_file_to_list(filename)
print(a)
```

input

```
['hello', 'gateway classes', 'unit2', 'unit3']
```

write a python program to read a file line by line store it into a list

main.py | example.txt

```
1 hello
2 gateway classes
3 unit2
4 unit3
```

write a python program to read a file line by line store it into a variable(AKTU 2019-20)

```
1 file1 = open('myfile.txt', 'r')
2 file_content = ""
3 for line in file1:
4     file_content += line
5 file1.close()
6 print(file_content)
7 print(type(file_content))
8
```

```
hello student
welcome
to gatewayclasses
unit 4
file handling
<class 'str'>
```

main.py | myfile.txt | file.txt | file1

```
1 hello student
2 welcome
3 to gatewayclasses
4 unit 4
5 file handling
```

.Write a Python program to read an entire text file

main.py | myfile.txt | file.txt | file1

```
1 with open('myfile.txt', 'r') as file:
2     file_content = file.read()
3 print(file_content)
```

```
hello student
welcome
to gatewayclasses
unit 4
file handling
```

main.py | myfile.txt | file.txt | file1

```
1 hello student
2 welcome
3 to gatewayclasses
4 unit 4
5 file handling
```

. Write a Python program to read first n lines of a file

main.py example.txt

```
1
2  def read_first_n_lines(file_name, n):
3      with open(file_name, 'r') as file:
4          for i in range(n):
5              line = file.readline()
6              if not line:
7                  break
8              print(line, end="")
9  file_name = 'example.txt'
10 n = 5
11 read_first_n_lines(file_name, n)
12
```

```
hello
gateway classes
unit2
unit3
hello
```

main.py example.txt

```
1 hello
2 gateway classes
3 unit2
4 unit3
5 hello
6 python
7 gateway
```

. Write a Python program to append text to a file and display the text

main.py | myfile.txt | file.txt | file1 | geek.txt

```python
def append_text_to_file(file_name, text):
    with open(file_name, 'a') as file:
        file.write(text + '\n')
def display_file_content(file_name):
    with open(file_name, 'r') as file:
        file_content = file.read()
    print(file_content)
file_name = 'myfile.txt'
text_to_append = "This is the new text to append."
append_text_to_file(file_name, text_to_append)
display_file_content(file_name)
```

input

```
hello student
welcome
to gatewayclasses
unit 4
file handling
hi
bye
hello
bbbbThis is the new text to append.
```

main.py | myfile.txt | file.txt

```
hello student
welcome
to gatewayclasses
unit 4
file handling
hi
bye
hello
bbbb
```

main.py | myfile.txt | file.txt | file1

```
hello student
welcome
to gatewayclasses
unit 4
file handling
hi
bye
hello
bbbbThis is the new text to append.
```

Write a Python program to read last n lines of a file

main.py | myfile.txt | test.txt

```python
def read_last_n_lines(file_name, n):
    with open(file_name, 'r') as file:
        # Read all lines from the file
        lines = file.readlines()
        # Get the last n lines
        last_n_lines = lines[-n:]
        # Print the last n lines
        for line in last_n_lines:
            print(line, end='')

# Example usage
file_name = 'myfile.txt'
n = 5
read_last_n_lines(file_name, n)
```

```
by
Hello
This
then
This is the new text to append.
```

```
1  Hello
2  This is Delhi
3  This is Paris
4  This is London
5  by
6  Hello
7  This
8  then
9  This is the new text to append.
10
```

Construct a program to change the contents of the file by separating each character by comma:

Hello!! Output H,e,1,1,0,!(AKTU 2023-24)

```
Hello
This is Delhi
This is Paris
This is London
by
Hello
This
then
This is the new text to append.

```

```
H,e,l,l,o,
,T,h,i,s, ,i,s, ,D,e,l,h,i,,
,T,h,i,s, ,i,s, ,P,a,r,i,s,,
,T,h,i,s, ,i,s, ,L,o,n,d,o,n, ,
,b,y,
,H,e,l,l,o,
,T,h,i,s,
,t,h,e,n,
,T,h,i,s, ,i,s, ,t,h,e, ,n,e,w, ,t,e,x,t, ,t,o, ,a,p,p,e,n,d,.,

```

Construct a program to change the contents of the file by separating each character by comma: Hello!! Output H,e,l,l,0,!(AKTU 2023-24 odd)

main.py | myfile.txt | test.txt

myfile.txt

```python
def separate_characters_by_comma(file_name):
    # Open the file in read mode to read the content
    with open(file_name, 'r') as file:
        # Read the entire content of the file
        content = file.read()

    # Create a new string with each character separated by a comma
    separated_content = ','.join(content)

    # Open the file in write mode to overwrite with the modified content
    with open(file_name, 'w') as file:
        # Write the modified content to the file
        file.write(separated_content)

# Example usage
file_name = 'myfile.txt'
separate_characters_by_comma(file_name)
```

Construct a program which accepts a sequence of words separated by whitespace as file input. Print the words composed of digits only.(AKTU 2023-24 ODD)

main.py | input.txt | file.txt | file1 | geek.txt

```python
def create_input_file(filename):
    user_input = input("Enter a sequence of words separated by whitespace: ")
    with open(filename, 'w') as file:
        file.write(user_input)
def find_digit_words(filename):
    # Open the file and read its contents
    with open(filename, 'r') as file:
        content = file.read()
    # Split the content into words
    words = content.split()
    # Find words composed of digits only
    digit_words = [word for word in words if word.isdigit()]
  # Print the words composed of digits only
    print("Words composed of digits only:")
    for word in digit_words:
        print(word)
filename = 'input.txt'
create_input_file(filename)
find_digit_words(filename)
```

Construct a program which accepts a sequence of words separated by whitespace as file input. Print the words composed of digits only.(AKTU 2023-24 ODD)

main.py | input.txt | file.txt | file

```
1 hello 1233 hu 45 mkobye 55
```

input

```
Enter a sequence of words separated by whitespace: hello 1233 hu 45 mkobye 55
Words composed of digits only:
1233
45
55
```

what is the output of the following question

AKTU (2023-24 Odd)

```
def count(s):
    for str in string.split():
        s = "&".join(str)
        return s
print(count("Python is fun to learn."))
```

main.py

```
def count(s):
    for str in s.split():
        s ='&'.join(str)
        return s
print(count("hello"))
```

```
h&e&l&l&o
```

describe python program to write the number of letters and digits in given input string into a file object

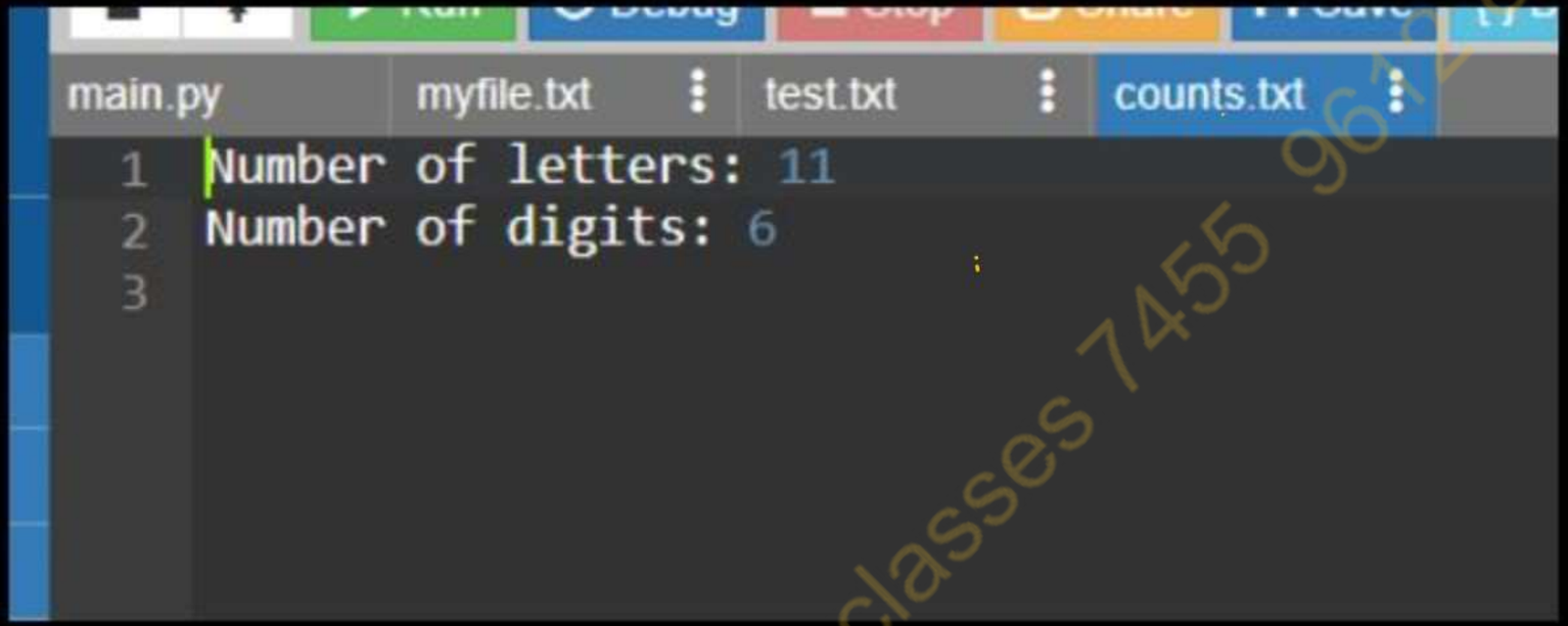
Input_string: "Python 3.9 is fun! 2024"

```python
def count_letters_and_digits(input_string):
    # Initialize counters for letters and digits
    letter_count = 0
    digit_count = 0

    # Iterate through each character in the input string
    for char in input_string:
        if char.isalpha():
            letter_count += 1
        elif char.isdigit():
            digit_count += 1

    return letter_count, digit_count
def write_counts_to_file(file_name, letter_count, digit_count):
    # Open the file in write mode
    with open(file_name, 'w') as file:
        # Write the counts of letters and digits
        file.write(f"Number of letters: {letter_count}\n")
        file.write(f"Number of digits: {digit_count}\n")
input_string = "Python 3.9 is fun! 2024"
file_name = 'counts.txt'
#Get the counts
letter_count, digit_count = count_letters_and_digits(input_string)
# Write the counts to the file
write_counts_to_file(file_name, letter_count, digit_count)
```

describe python program to write the number of letters and digits in given input string into a file object



Write a Python program to read a file line by line store it into an array

main.py example.txt

```python
def read_file_into_array(file_name):
    # Initialize an empty list to store the lines
    lines_array = []

    # Open the file in read mode
    with open(file_name, 'r') as file:
        # Read the file line by line
        for line in file:
            # Strip leading/trailing whitespace and add the line to the
            lines_array.append(line.strip())

    return lines_array

# Example usage
file_name = 'example.txt'
lines = read_file_into_array(file_name)

# Print the lines stored in the array
for line in lines:
    print(line)
```

```
hello gateway


...Program finished with exit code
```

main.py example.txt

```
1 hello gateway
2
```

Write a Python program to find out the longest word in a file

main.py | test.txt | file.txt | file1 | geek.txt

```python
def find_longest_word(filename):
    with open(filename, 'r') as file:
        content = file.read()

    words = content.split()
    longest_word = max(words, key=len)

    return longest_word

filename = 'test.txt'
longest_word = find_longest_word(filename)
print(f"The longest word in the file '{filename}' is: {longest_word}")

```

input

```
The longest word in the file 'test.txt' is: gatewayclasses
```

main.py | test.txt | file.txt | file1

```
hello students gatewayclasses
python programming
unit 4
important
easy one
```

Write a Python program to read a file line by line store it into an array array module's typecode 'u' is not used in recent Python versions as it was deprecated and eventually removed. In modern Python versions (3.x), array typecodes for storing text are no longer supported

main.py example.txt

```python
import array

def read_file_to_array(filename):
    lines = array.array('u')  # 'u' typecode is for Unicode characters
    with open(filename, 'r') as file:
        for line in file:
            lines.append(line.strip())  # Remove newline characters and add line to the array
    return lines

# Example usage
filename = 'example.txt'
lines = read_file_to_array(filename)
print(lines)
```

Write a Python program to count the number of lines in a text file.

main.py | test.txt | file.txt | file1 | geek.txt

```
1  def count_lines(filename):
2      with open(filename, 'r') as file:
3          lines = file.readlines()
4      return len(lines)
5  filename = 'test.txt'
6  num_lines = count_lines(filename)
7  print("The number of lines in the file",num_lines)
8
9
10
```

input

```
The number of lines in the file 5
```

Run | Debug | Stop | Share

main.py | test.txt

```
1 hello gateway python programming
2 hi
3 python
4 unit4
5 file handling
6
```

Write a Python program to count the frequency of words in a file.

main.py | test.txt

```python
from collections import Counter
def word_count(fname):
        with open(fname) as f:
                return Counter(f.read().split())

print("Number of words in the file :",word_count("test.txt"))
```

main.py | test.txt

```
hello gateway python programming
hi
python
unit4
file handling unit4
python
hello
```

input

```
Number of words in the file : Counter({'python': 3, 'hello': 2, 'unit4': 2, 'gateway': 1, 'programming': 1, 'hi': 1, 'file': 1, 'handling': 1})
```

Write a Python program to copy the content of one file to another

```
1 from shutil import copyfile
2 copyfile('test.py', 'abc.py')
3
```

Write a Python program to read the random line in a file

main.py | test.txt

```
1 import random
2 def random_line(fname):
3     lines = open(fname).read().splitlines()
4     return random.choice(lines)
5 print(random_line('test.txt'))
6
```

input

```
hello
```

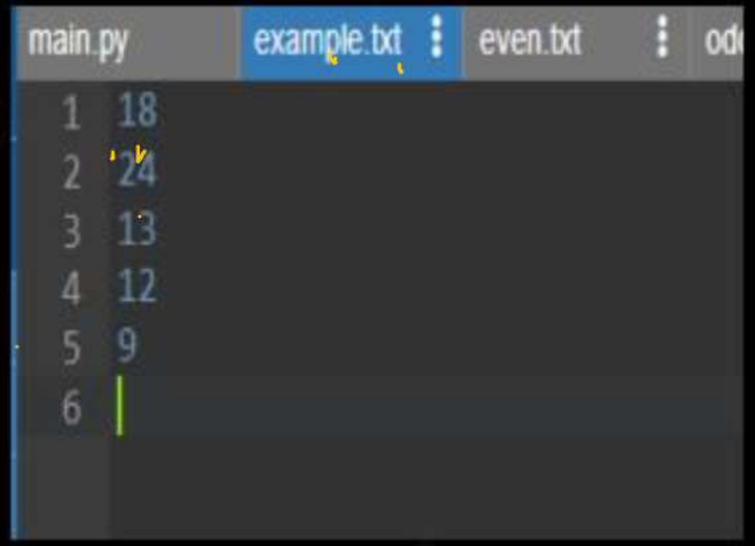
main.py | test.txt

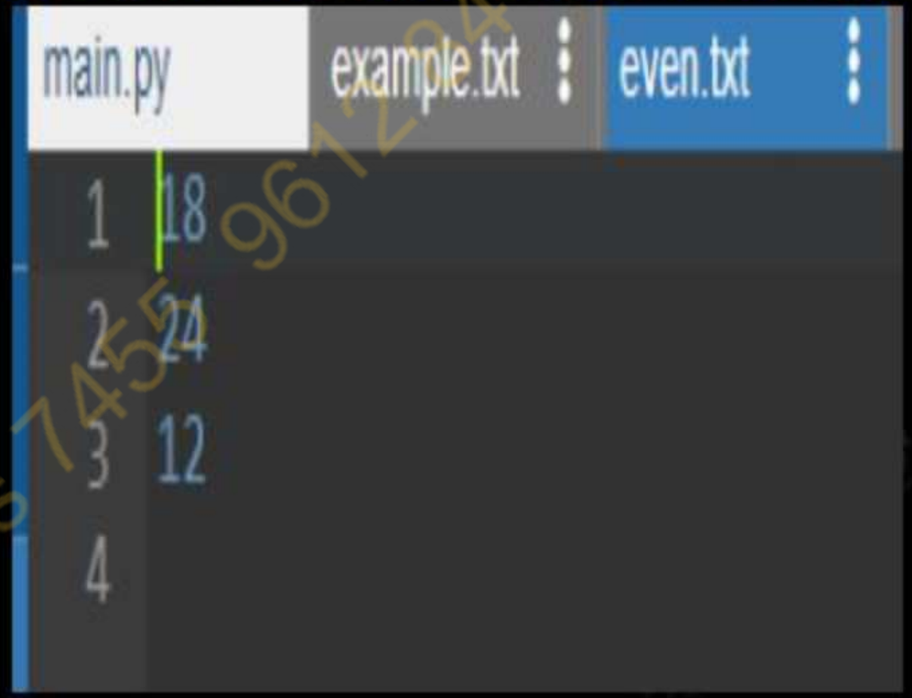
```
1 hello gateway python programming
2 hi
3 python
4 unit4
5 file handling unit4
6 python
7 hello
```

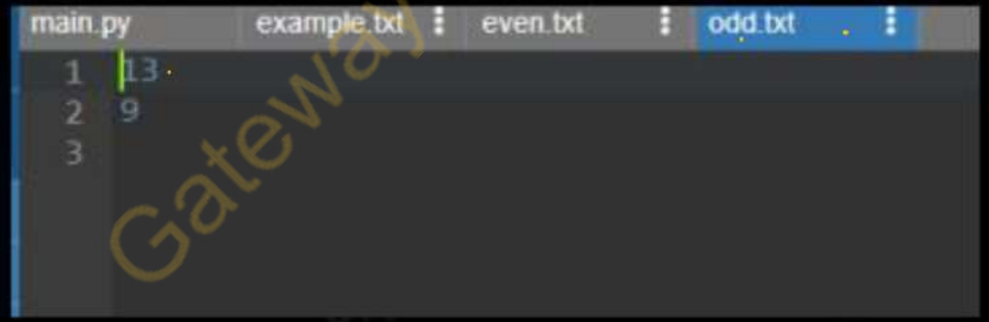
```
hello
```

There is a file named ~~input.txt~~ Example.txt. Enter some positive number into the file name ~~input.txt~~ Example.txt . Read the content of the file and if it is an odd number write it to odd.txt and if the number is even, write it to even.txt

(AKTU 2018-19)







main.py | example.txt | even.txt | odd.txt

```
1  with open("example.txt","w")as file:
2      file.write("18\n")
3      file.write("24\n")
4      file.write("13\n")
5      file.write("12\n")
6      file.write("9\n")
7  file1=open("example.txt","r")
8  for i in file1:
9      if i.strip:
10         num=int(i)
11         if num%2==0:
12             even=open("even.txt","a")
13             even.write(str(num))
14             even.write("\n")
15         else:
16             odd=open("odd.txt","a")
17             odd.write(str(num))
18             odd.write("\n")
19 file.close()
20
21
```

Write a Python program to assess if a file is closed or not.

main.py | test.txt

```
1  f = open('test.txt','r')
2  print(f.closed)
3  f.close()
4  print(f.closed)
5
6
7
8
```

```
False
True
```

Write a Python program to get the file size of a plain file

main.py abc.txt

```
1  def file_size(fname):
2          import os
3          statinfo = os.stat(fname)
4          return statinfo.st_size
5
6  print("File size in bytes of a plain file: ",file_size("abc.txt"))
7
```

input

```
File size in bytes of a plain file:  7
```

main.py abc.txt

```
1  he
2
3
```

Write a Python program to combine each line from first file with the corresponding line in second file

main.py | test.txt | abc.txt

```
with open('abc.txt') as fh1, open('test.txt') as fh2:
    for line1, line2 in zip(fh1, fh2):
        # line1 from abc.txt, line2 from test.txtg
        print(line1.strip() + line2.strip())
```

input

```
hhhhhello gateway python programming
hellobye
```

main.py | test.txt | abc.txt

```
hello gateway python programming
bye
```

main.py | test.txt | abc.txt

```
hhhh
hello
```

write a python code to create a file with p.txt name and write your name and father name in this file and read this file to print it (AKTU

```python
# Write to the file p.txt
with open('p.txt', 'w') as file:
    file.write("Your Name: John Doe\n")
    file.write("Father's Name: Richard Roe\n")

print("File p.txt created and names written.")

# Read from the file p.txt and print its contents
with open('p.txt', 'r') as file:
    contents = file.read()
    print("Contents of p.txt:")
    print(contents)
```

input

```
File p.txt created and names written.
Contents of p.txt:
Your Name: John Doe
Father's Name: Richard Roe
```

main.py | p.txt | abc.txt

```
Your Name: John Doe
Father's Name: Richard Roe
```

Change all the numbers in the file to text. Construct a program for the same.Example:Given 2 integer numbers, return their product only if the product is equal to or lower than 10.And the result should be :Given two integer numbers, return their product only if the product is equal to or lower than one zero

main.py | input.txt | abc.txt

```
num_to_text = {
    '0': 'zero', '1': 'one', '2': 'two',
    '3': 'three', '4': 'four', '5': 'five', '6': 'six', '7': 'seven',
    '8': 'eight', '9': 'nine'
}
def replace_numbers_with_text(text):
    result = []

    for char in text:
        if char.isdigit():
            result.append(num_to_text[char])
        else:
            result.append(char)

    return ''.join(result)
def process_file(filename):
    with open(filename, 'r') as file:
        content = file.read()
    modified_content = replace_numbers_with_text(content)
    with open(filename, 'w') as file:
        file.write(modified_content)
    print("Modified content of the file:")
    print(modified_content)
process_file('input.txt')
```

AKTU 2023-24 one

main.py | input.txt | file1.txt | file2.txt

```
1 hello unit 3 unit 4 unit 5 are most important unit
2
```

main.py | input.txt | abc.txt

```
1 hello unit three unit four and unit five  are most important unit
2
```

input

```
Modified content of the file:
hello unit three unit four and unit five  are most important unit
```

Process finished with exit code 0

Write a python function, searchMany(s,x,k) that takes an argument a sequence s and integersx,k(k>0).The function returns true if there are at most k occurrences of x in s .otherwise it returns false searchMany ([10,17,15,12],15,1) return true, searchMany([10,12,12,12],12,2) return falseA

main.py | input.txt | abc.txt

```
def searchMany(s, x, k):
    count = s.count(x)
    return count <= k

# Test cases
print(searchMany([10, 17, 15, 12], 15, 1))
print(searchMany([10, 12, 12, 12], 12, 2))
print(searchMany([10, 12, 12, 12,12,12,12], 12, 7))
```

input

```
True
False
True
```

Seek()

In Python, seek() function is used to **change the position of the File Handle** to a given specific position. File handle is like a cursor, which defines from where the data has to be read or written in the file

syntax: f.seek(offset, from_what),.

F is file

Parameters:

Offset: *Number of positions to move forward*

from_what: *It defines point of reference.*

Returns: *Return the new absolute position*

The reference point is selected by the **from_what** argument. It accepts three values.

0: sets the reference point at the beginning of the file

1: sets the reference point at the current file position

2: sets the reference point at the end of the file

By default from_what argument is set to 0.

How you can use seek() to move the read position to the beginning of the file

```python
with open('example.txt', 'r') as file:
    line = file.readline()
    print("First read:", line)
    file.seek(0)
    line = file.readline()
    print("After seek:", line)
```

```
First read: hello students

After seek: hello students
```

main.py | example.txt | abc.txt

```
hello students
welcome to gateway classes
unit4
important file handling
```

How do you use seek() to move write position to the beginning of file

main.py | example.txt | abc.txt

```
1  # Open the file in read/write mode ('r+')
2  with open('example.txt', 'r+') as file:
3      file.write("This is the first line.\n")
4      file.seek(0)
5      file.write("This is the new content.\n")
6      file.seek(0)
7      content = file.read()
8      print(content)
9
```

```
This is the new content.
ateway classes
unit4
important file handling
```

main.py | example.txt | abc.txt

```
1  hello student welcome
2   gateway classes
3  unit4
4  important file handling
5
```

main.py | example.txt | abc.txt

```
1  This is the first line.
2  gateway classes
3  unit4
4  important file handling
5
```

main.py | example.txt | abc.txt

```
1  This is the new content.
2  ateway classes
3  unit4
4  important file handling
5
```

How do you use seek() to reset the read/write position to specific location

main.py | example.txt | abc.txt

```
1 with open('example.txt', 'r+') as file:
2     file.write("This is the first line.\n")
3     file.write("This is the second line.\n")
4     file.seek(20)
5     file.write("This is the new second line.\n")
6     file.seek(0)
7     content = file.read()
8     print(content)
9
10
```

input

```
This is the first liThis is the new second line.
ne.
```

example.txt

```
1 This is the first line.
2 This is the new second line.
3
```

main.py | example.txt | abc.txt

```
1 This is the first liThis is the new second line.
2 ne.
3
4
```

How do you use seek() to reset the read position relative to current position

main.py | example.txt | abc.txt

```
with open('example.txt', 'r+') as file:
    print("Initial read:", file.read(10))
    current_position = file.tell()
    file.seek(current_position + 5)
    print("Read after moving forward by 5 bytes:", file.read(10))

```

input

```
Initial read: this is th
Read after moving forward by 5 bytes: mple of se
```

main.py | example.txt | abc.txt

```
this is the example of seek() hello students welcome to gateway
```

how do you use seek() to reset the write position relative to the current position

main.py | example.txt | abc.txt

```python
# Open the file in read/write mode ('r+')
with open('example.txt', 'r+') as file:
    # Write initial content
    file.write("This is the initial content\n")

    # Move the pointer to the end of the file
    file.seek(0, 2)  # 2 means from the end of the file

    # Get the current position (end of file)
    end_position = file.tell()

    # Move the pointer back by 10 bytes from the end
    file.seek(end_position - 10)

    # Write new content at this new position
    file.write("New content at position.\n")

    # Move the pointer to the beginning to read the file content
    file.seek(0)

    # Read and print the file content
    content = file.read()
    print(content)

```

```
This is the initialNew content at position.
```

Thank You