AKTU B.Tech Second Year

PYTHON PROGRAMMING

UNIT 4

Python Program Flow Control Conditional blocks

LEC-1

Today's Target

AKTU PYQs

By
Pragya Rajvanshi

Gateway Classes

FILES

- 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

Gateway classes

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.

Complexity : File handling in Python can be complex, especially when working with more advanced file formats or operations. Careful attention must be paid to the code to ensure that files are handled properly and securely.

Performance : File handling operations in Python can be slower than other programming languages, especially when dealing with large files or performing complex operations.

Gateway classes

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.

Gateway classes

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.

Gateway classes

Open a file the to read the content

```
file = open("sales.txt", "r")
content = file.read()
print(content)
file.close()
```

main.py | sales.txt

```
1  hello
2  students
3  unit 4
```

```
hello
students
unit 4
```

Open a file to Write the content

```
file = open("myfile.txt", "w")
file.write("how are you?")
file.close()
```

main.py | myfile.txt

```
1  how are you?
```

Open a file to append the data

```
file = open("myfile.txt", "a")
file.write("hiiii")
file.close()
```

```
how are you?hiiii
```

Gateway classes

W+ mode example

```
file_name = "example.txt"
with open(file_name, "w+") as file:
    file.write("Hello, this is a test using w+ mode.\n")
    file.write("This mode allows both writing and reading.\n")
    file.seek(0)
    content = file.read()
    print("File content after writing:")
    print(content)
# Verify the file contents by reopening in read mode
with open(file_name, "r") as file:
    print("Verified content from the file:")
    print(file.read())
```

Gateway classes

main.py | example.txt

```
1  Hello, this is a test using w+ mode.
2  This mode allows both writing and reading.
3
```

```
File content after writing:
Hello, this is a test using w+ mode.
This mode allows both writing and reading.

Verified content from the file:
Hello, this is a test using w+ mode.
This mode allows both writing and reading.
```

Gateway Classes

Without seek ()

```
file_name = "example.txt"
with open(file_name, "w+") as file:
    file.write("Hello, this is a test using w+ mode.\n")
    file.write("This mode allows both writing and reading.\n")
    content = file.read()
    print("File content after writing:")
    print(content)
with open(file_name, "r") as file:
    print("Verified content from the file:")
    print(file.read())
```

```
File content after writing:

Verified content from the file:
Hello, this is a test using w+ mode.
This mode allows both writing and reading.
```

ay classes

r+ mode

main.py | example.txt

```
1  file_name = "example.txt"
2  with open(file_name, "w") as file:
3      file.write("This is the original content.\n")
4  with open(file_name, "r+") as file:
5      print("Original file content:")
6      print(file.read())
7      file.seek(0)
8      file.write("Updated content.\n")
9      file.seek(0)
10     print("\nUpdated file content:")
11     print(file.read())
12 # Verify the file content by reopening in read mode
13 with open(file_name, "r") as file:
14     print("\nVerified content from the file:")
15     print(file.read())
```

Gateway classes

```
Original file content:
This is the original content.


Updated file content:
Updated content.
nal content.


Verified content from the file:
Updated content.
nal content.
```

```
This is the original content.
```

```
Updated content.
nal content.
```

Gateway classes

a+ mode

```
file_name = "example.txt"
with open(file_name, "a+") as file:
    file.write("This is\n")
    print("\nUpdated file content:")
    file.seek(0)
    print(file.read())
```

```
Updated file content:
hi students
This is
```

main.py | example.txt

```
1  hi students
2  This is
3
```

Gateway Cl

Different ways read can be performed

Read()

main.py | example.txt

```
1  file_name = "example.txt"
2  with open(file_name, "w") as file:
3      file.write("Line 1: Hello\n")
4      file.write("Line 2: World\n")
5      file.write("Line 3: Python\n")
6  with open(file_name, "r") as file:
7      # Using read()
8      print("Using read():")
9      content = file.read()
10     print(content)
11
```

main.py | example.txt

```
1  Line 1: Hello
2  Line 2: World
3  Line 3: Python
4
```

```
Using read():
Line 1: Hello
Line 2: World
Line 3: Python
```

Gateway classes

Different ways read can be performed

readline()

```
file_name = "example.txt"
with open(file_name, "w") as file:
    file.write("Line 1: Hello\n")
    file.write("Line 2: World\n")
    file.write("Line 3: Python\n")

with open(file_name, "r") as file:
    print("\nUsing readline():")
    line = file.readline()
    print(line, end="")
    line = file.readline()
    print(line, end="")
```

```
Using readline():
Line 1: Hello
Line 2: World
```

main.py | example.txt

```
1  Line 1: Hello
2  Line 2: World
3  Line 3: Python
4
```

Gateway Class

Different ways read can be performed

readlines()

main.py example.txt

```
1  file_name = "example.txt"
2  with open(file_name, "w") as file:
3      file.write("Line 1: Hello\n")
4      file.write("Line 2: World\n")
5      file.write("Line 3: Python\n")
6  with open(file_name, "r") as file:
7      print("\nUsing readlines():")
8      lines = file.readlines()
9      print(lines)
```

```
Using readlines():
['Line 1: Hello\n', 'Line 2: World\n', 'Line 3
: Python\n']
```

Gateway Classes

AKTU B.Tech Second Year

PYTHON PROGRAMMING

UNIT 4

Python Program Flow Control Conditional blocks

LEC-2

Today's Target

AKTU PYQs

By
Pragya Rajvanshi

Gateway Classes

Read()	Readline()	Readlines()
• The read method reads the entire contents of a file and returns it as a string. • content = file.read()	➢ 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()

Gateway classes

Write in a 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()

Gateway classes

main.py example.txt

```
1  file_name = "example.txt"
2  # Using write()
3  with open(file_name, "w") as file:
4      file.write("This is a single line written using write().\n")
5  # Using writelines()
6  with open(file_name, "a") as file:  # Open in append mode
7      lines = ["Line 1: Hello\n", "Line 2: World\n", "Line 3: Python\n"]
8      file.writelines(lines)
9  # Verify file content
10 with open(file_name, "r") as file:
11     print("File content:")
12     print(file.read())
13
```

Gateway classes

main.py | example.txt

```
1  This is a single line written using write().
2  Line 1: Hello
3  Line 2: World
4  Line 3: Python
5
```

```
File content:
This is a single line written using write().
Line 1: Hello
Line 2: World
Line 3: Python
```

Gateway Classes

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

```python
def read_file_to_list(file_name):
    lines=[]
    with open(file_name,'r')as file:
        while True:
            k=file.readline()
            if not k:
                break
            lines.append(k.strip())
    return lines
file_name="example.txt"
a=read_file_to_list(file_name)
print(a)
```

main.py | example.txt

```
Line 1: Hello
Line 2: World
Line 3: Python
```

```
['Line 1: Hello', 'Line 2: World', 'Line 3: Python']
```

Gateway classes

Write a Python program to read an entire text file

main.py | example.txt

```
1 file_name = "example.txt"
2 with open(file_name, "r") as file:
3     content = file.read()
4 print("Content of the file:")
5 print(content)
6
```

```
Content of the file:
hello
while
hi
bye
```

main.py | example.txt

```
1 hello
2 while
3 hi
4 bye
```

Gateway classes

Write a Python program to read a file line by line store it into a variable.

main.py | example.txt

```
1 file_name = "example.txt"
2 k = ""
3 with open(file_name, "r") as file:
4     for line in file:
5         k=k+line
6 print("Content of the file stored in a variable:")
7 print(k)
```

main.py | example.txt

```
1 hello
2 while
3 hi
4 bye
```

```
Content of the file stored in a variable:
hello
while
hi
bye
```

Gateway classes

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

main.py | example.txt

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

```
enter the number
3
hello
while
hi
```

main.py | example.txt

```
1 hello
2 while
3 hi
4 bye
5 breakk
6 lambda
```

Gateway classes

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

main.py example.txt

```python
def read_last_n_lines(file_name, n):
    with open(file_name, "r") as file:
        lines = file.readlines()
        last_n_lines = lines[-n:]
    return last_n_lines
file_name = "example.txt"
n = 3
last_lines = read_last_n_lines(file_name, n)
print("Last", n, "lines from the file:")
for line in last_lines:
    print(line, end="")

```

```
Last 3 lines from the file:
bye
breakk
lambda
```

main.py example.txt

```
hello
while
hi
bye
breakk
lambda
```

Gateway class

Write a python program to find the longest words.

main.py | example.txt

```
def find_longest_word(file_name):
    longest_word = ""
    with open(file_name, "r") as file:
        for line in file:
            # Split the line into words
            words = line.split()
 # Check each word's length and update the longest word
            for word in words:
                if len(word) > len(longest_word):
                    longest_word = word
    return longest_word
file_name = "example.txt"
longest_word = find_longest_word(file_name)
print("The longest word in the file is:", longest_word)
```

Gateway Classes

```
The longest word in the file is: students
```

main.py | example.txt

```
1  hello students unit 4 start
```

Gateway Class

AKTU B.Tech Second Year

PYTHON PROGRAMMING

UNIT 4

Python file operations

LEC-3

Today's Target

- **File program**
- **AKTU PYQs**

Gateway Classes

By
Pragya Rajvanshi

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

```
1  hello hi1 123 11
```

```
enter the sequence of word seprated by whitespaces
hello hi1 123 11
words composed of digit only


123
11
```

classes

main.py | input.txt

```
1  def create_input_file(filename):
2      user_input=input("enter the sequence of word seprated by whitespaces\n")
3      with open(filename,'w') as file:
4          file.write(user_input)
5  def find_digit_words(filename):
6      with open(filename,'r') as file:
7          content=file.read()
8      x=content.split()
9      digit_words=[word for word in x if word.isdigit()]
10     print("words composed of digit only\n")
11     for y in digit_words:
12         print(y)
13 filename='input.txt'
14 create_input_file(filename)
15 find_digit_words(filename)
```

Gateway classes

Write a program that a take space separated words and convert the digit into English word
Like
Hello 1 2
Output hello one two

```
1 hello 1 2 good 3
```

main.py | test.txt

```
1 hello one two good three
```

```
hello one two good three
```

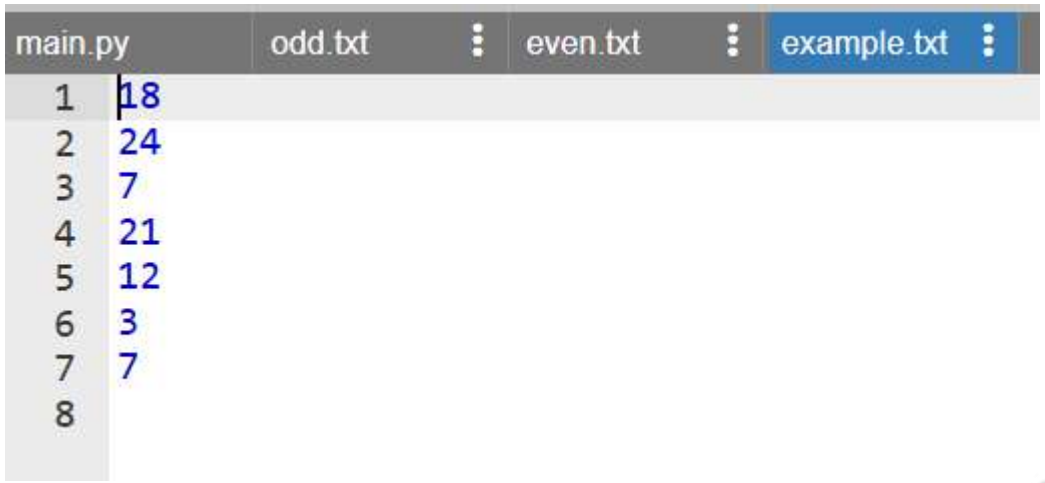
Gateway classes

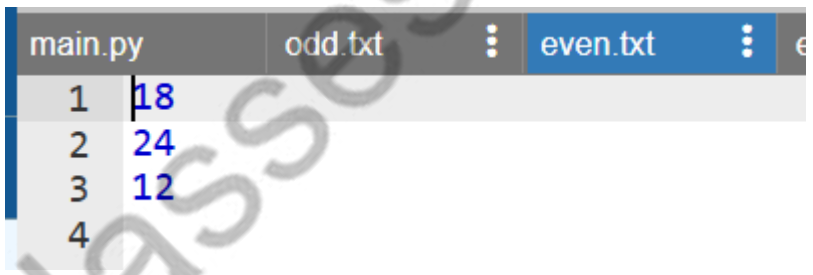
main.py | test.txt

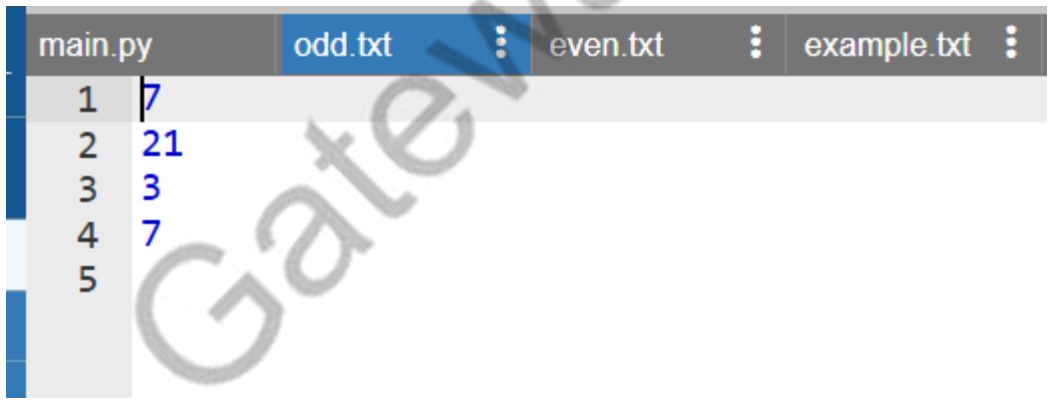
```python
num_to_text={
    '0':'zero','1':'one' ,'2':'two','3':'three','4':'four',
    '5':'five', '7':'seven','8':'eight','9':'nine'
}
def replace_number_with_text(text):
    result=[]
    for char in text:
        if char.isdigit():
          result.append(num_to_text[char])
        else:
              result.append(char)
    return ''.join(result)
def process_file(filename):
    with open(filename,'r') as file:
        content=file.read()
    modified_content=replace_number_with_text(content)
    with open(filename,'w') as file:
        file.write(modified_content)
    print(modified_content)
process_file('test.txt')
```

Gateway classes

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







Gateway classes

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

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

Gateway classes

Write a python program to find the longest words.

main.py example.txt

```
def find_longest_word(file_name):
    longest_word = ""
    with open(file_name, "r") as file:
        for line in file:
            # Split the line into words
            words = line.split()
 # Check each word's length and update the longest word
            for word in words:
                if len(word) > len(longest_word):
                    longest_word = word
    return longest_word
file_name = "example.txt"
longest_word = find_longest_word(file_name)
print("The longest word in the file is:", longest_word)
```

Gateway classes

```
The longest word in the file is: students
```

main.py | example.txt

```
1  hello students unit 4 start
```

Gateway class

AKTU B.Tech Second Year

PYTHON PROGRAMMING

UNIT 4

Python file operations

LEC-4

Today's Target

- **File program**
- **AKTU PYQs**

Gateway Classes

By
Pragya Rajvanshi

Example:

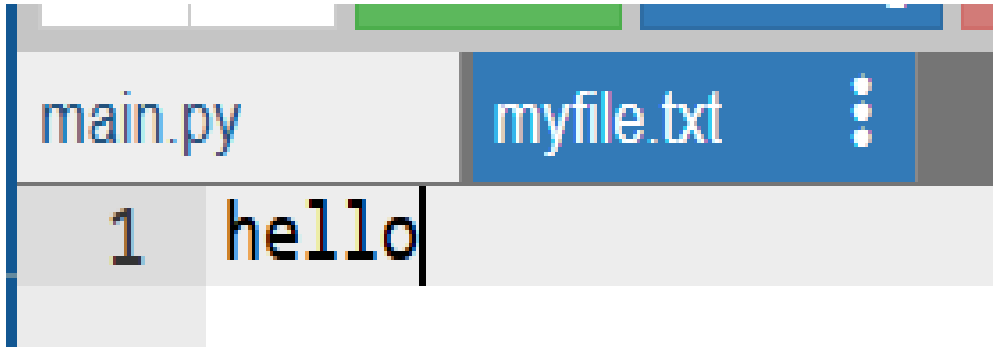
main.py

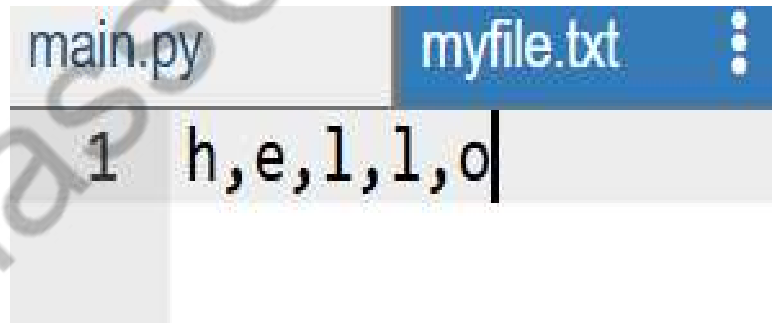
```
1  #example1
2  words = ["Hello", "world", "Python"]
3  result = " ".join(words)
4  print(result)
5  #example 2
6  k="hello"
7  result2 = ",".join(k)
8  print(result2)
```



Gateway classes

Construct a program to change the content of the fie by separating each character by comma





Gateway classes

main.py myfile.txt

```
1 def seprate_character_by_comma(file_name):
2     with open(file_name,'r') as file:
3         content=file.read()
4         seprated_content=','.join(content)
5     with open (file_name,'w') as file:
6         file.write(seprated_content)
7 file_name="myfile.txt"
8 seprate_character_by_comma(file_name)
```

Gateway classes

What will be the output:

main.py

```
1 def count(s):
2     for str in string.split():
3         s="&".join(str)
4         return s
5 print(count("python is fun to learn"))
6
```

input

```
Traceback (most recent call last):
  File "/home/main.py", line 5, in <module>
    print(count("python is fun to learn"))
  File "/home/main.py", line 2, in count
    for str in string.split():
NameError: name 'string' is not defined
```

```
1 def count(s):
2     for str in s.split():
3         s="&".join(str)
4         return s
5 print(count("python is fun to learn"))
6
```



classes

What will be the output:

main.py

```
1 def count(s):
2     for str in s.split():
3         s="&".join(str)
4     return s
5 print(count("python is fun to learn"))
6
```

input

```
l&e&a&r&n
```

main.py

```
1 def count(s):
2     for str in s.split():
3         s="&".join(str)
4         print(s)
5 print(count("python is fun to learn"))
6
```

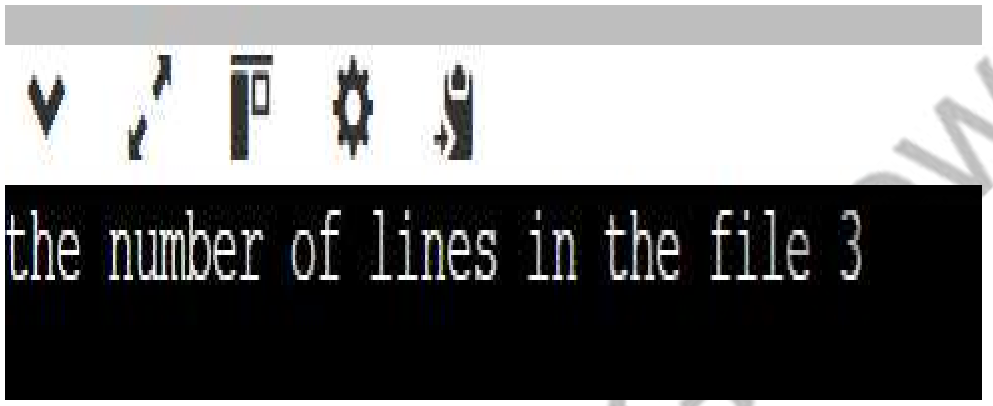
```
p&y&t&h&o&n
i&s
f&u&n
t&o
l&e&a&r&n
None
```

Gateway classes

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

main.py | test.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)
```



main.py | test.txt

```
1 hello
2 while
3 bye
```

Gateway classes

write a program to read a file ABC.TXT AND count the line word character in a file.

main.py ABC.TXT

```
1  def count_file_content(filename):
2      with open(filename,'r') as file:
3          lines=file.readlines()
4          line_count=len(lines)
5          word_count=0
6          char_count=0
7          for line in lines:
8              words=line.split()
9              word_count+=len(words)
10             char_count+=len(line)
11     print("line",line_count)
12     print("words",word_count)
13     print("character",char_count)
14 filename="ABC.TXT"
15 count_file_content(filename)
16
```

```
line 2
words 4
character 12
```

main.py ABC.TXT

```
1  h i break
2  jj
```

Gateway classes

write a program to read a file and capitalize first letter of every word in a file

main.py ABC.TXT

```
1  def capitalize_words_in_file(filename):
2      with open(filename, 'r') as file:
3          lines = file.readlines()
4
5      with open(filename, 'w') as file:
6          for line in lines:
7              capitalized_line = line.title()
8              file.write(capitalized_line)
9
10 # Specify the file name
11 filename = "ABC.TXT"
12 capitalize_words_in_file(filename)
13
```

main.py ABC.TXT

```
1 hello bye
2 while hi
3 bye
```

main.py ABC.TXT

```
1 Hello Bye
2 While Hi
3 Bye
```

Gateway classes

Write a python program to count the frequency of words in a file

main.py test.txt

```
1 from collections import Counter
2 def word_count(fname):
3     with open(fname) as f:
4         return Counter(f.read().split())
5 print(word_count("test.txt"))
6
```

main.py test.txt

```
1 hello students hello
```

```
Counter({'hello': 2, 'students': 1})
```

Gateway classes

AKTU B.Tech Second Year

PYTHON PROGRAMMING

UNIT 4

Python file operations

LEC-5

Today's Target

- **File program**
- **AKTU PYQs**

Gateway Classes

By
Pragya Rajvanshi

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

main.py | source.txt | destination.txt

```
1  hi students how are you?
2
```

main.py | source.txt | destination.txt

```
1  hi students how are you?
2
```

input

```
Content copied from source.txt to destination.txt.
```

Gateway classes

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

main.py | source.txt | destination.txt

```
1  def copy_file(source_file, destination_file):
2      with open(source_file, 'r') as src:
3          content = src.read()
4      with open(destination_file, 'w') as dest:
5          dest.write(content)
6      print(f"Content copied from {source_file} to {destination_file}.")
7  source_file = 'source.txt'
8  destination_file = 'destination.txt'
9  copy_file(source_file, destination_file)
10
```

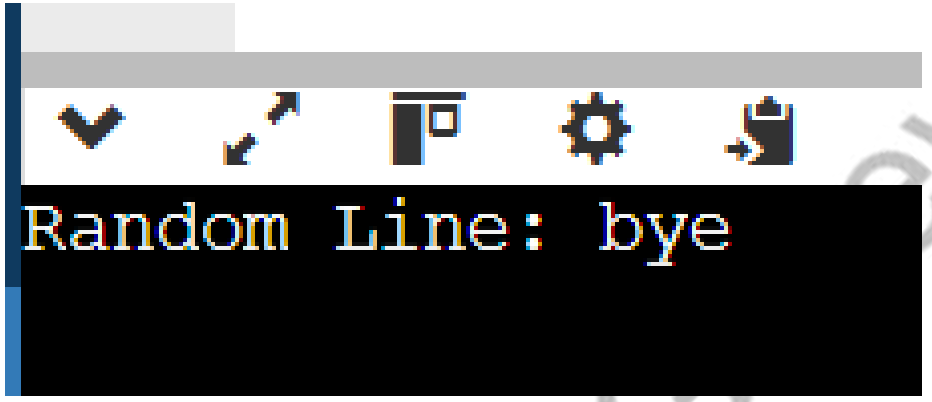
Gateway classes

Write a program to read the random line in a file

```python
import random
# Read a random line from a file
with open('example.txt', 'r') as file:
    lines = file.readlines()
    if lines:
        print("Random Line:", random.choice(lines).strip())
    else:
        print("The file is empty.")
```

main.py | example.txt

```
1  h0
2  hello
3  bye
4  hi
```

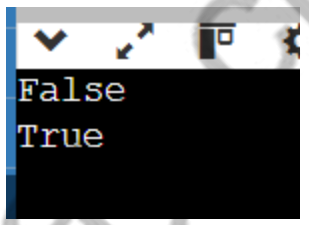


© Geeway classes

Write a python program to asses if file is closed or not

main.py | example.txt

```
1  f=open('example.txt',"r")
2  print(f.closed)
3  f.close()
4  print(f.closed)
5
```

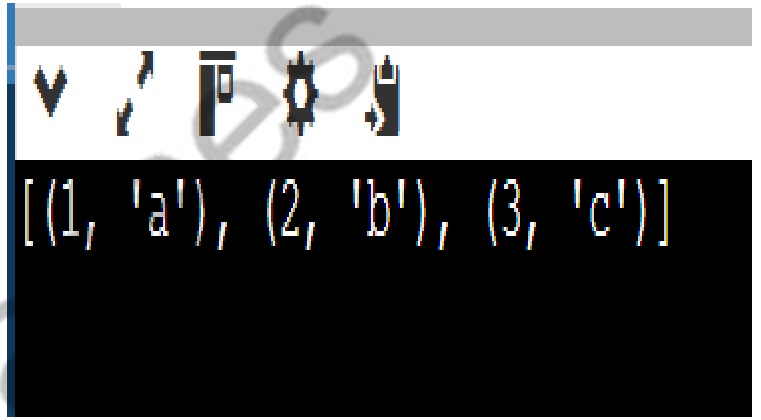


Gateway Classes

Example of zip()

main.py

```
1 list1 = [1, 2, 3]
2 list2 = ['a', 'b', 'c']
3 result = zip(list1, list2)
4 print(list(result))
```



Gateway Classes

Write a python program to combine each line from first file wit the corresponding line in second file

main.py | file1.txt | file2.txt | combined.txt

```
1
2  def combine_files(file1, file2, output_file):
3      with open(file1, 'r') as f1, open(file2, 'r') as f2, open(output_file, 'w') as output:
4          for line1, line2 in zip(f1, f2):
5              output.write(line1.strip() + ' ' + line2)
6
7      print(f"Combined content written to '{output_file}'.")
8
9  file1 = 'file1.txt'
10 file2 = 'file2.txt'
11 output_file = 'combined.txt'
12 combine_files(file1, file2, output_file)
13
```

Gateway classes

Write a python program to combine each line from first file wit the corresponding line in second file

main.py | file1.txt | file2.txt

```
1 hello
2 bye
```

main.py | file1.txt | file2.txt | combined.txt

```
1 break
2 objet
```

main.py | file1.txt | file2.txt | combined.txt

```
1 hello break
2 bye objet
```

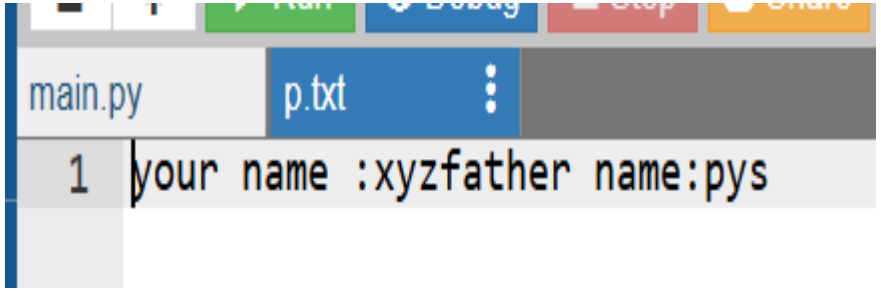
Gateway classes

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

main.py | p.txt

```
1 with open ("p.txt","w") as file:
2     file.write("your name :xyz")
3     file.write("father name:pys")
4 print("file p.txt craeted and names written")
5 with open('p.txt','r') as file:
6     contents=file.read()
7     print("content of p.txt")
8     print(contents)
```

```
file p.txt craeted and names written
content of p.txt
your name :xyzfather name:pys
```



Gay classes

Write a python program to get the file size of a plane file

main.py | example.txt

```
1  import os
2  def get_file_size(filename):
3      if os.path.exists(filename):
4          size=os.path.getsize(filename)
5          print(f"size ofthe file'{filename}':{size} bytes")
6      else:
7          print(f" the file'{filename}'' does not exist")
8  filename="example.txt"
9  get_file_size(filename)
```



Gateway classes

Write a program to remove newline character from file

main.py | example.txt | modified_example.txt

```
1  def remove_newlines(input_file, output_file):
2      with open(input_file, 'r') as file:
3          lines = file.readlines()  # Read all lines
4
5      # Remove newline characters from each line
6      modified_lines = [line.strip() for line in lines]
7
8      # Write the modified lines back to a new file
9      with open(output_file, 'w') as file:
10         file.write(' '.join(modified_lines))  # Join lines with a space
11
12     print(f"Newline characters removed and saved to '{output_file}'.")
13
14 input_file = 'example.txt'
15 output_file = 'modified_example.txt'
16 remove_newlines(input_file, output_file)
```

Gateway classes

main.py | example.txt | modi

```
1 hello
2 bye
3 while
4 hu
```

main.py | example.txt | modified_example.txt

```
1 hello bye while hu
```

Gateway classes

Construct the python program to count the number of letter and digits in given input string into a file object

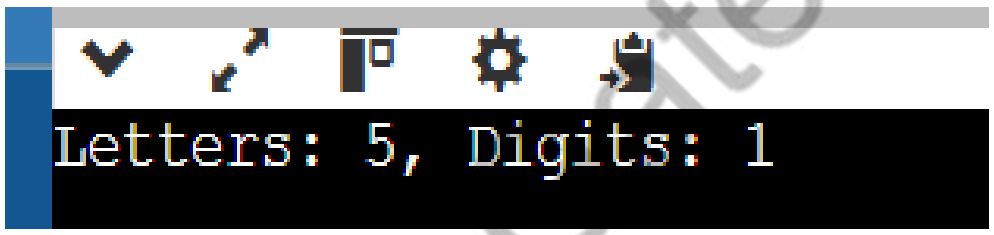
main.py | example.txt

```python
def count_letters_and_digits(file_obj):
    content = file_obj.read()
    letters = sum(c.isalpha() for c in content)
    digits = sum(c.isdigit() for c in content)
    return letters, digits
# Example usage
with open('example.txt', 'r') as file:
    letters, digits = count_letters_and_digits(file)
    print(f"Letters: {letters}, Digits: {digits}")

```

main.py | exam

```
while
2
```



Gateway classes

AKTU B.Tech Second Year

PYTHON PROGRAMMING

UNIT 4

Python file operations

LEC-6

Today's Target

- **File program**
- **AKTU PYQs**

Gateway classes

By
Pragya Rajvanshi

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), where f is file pointer*

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

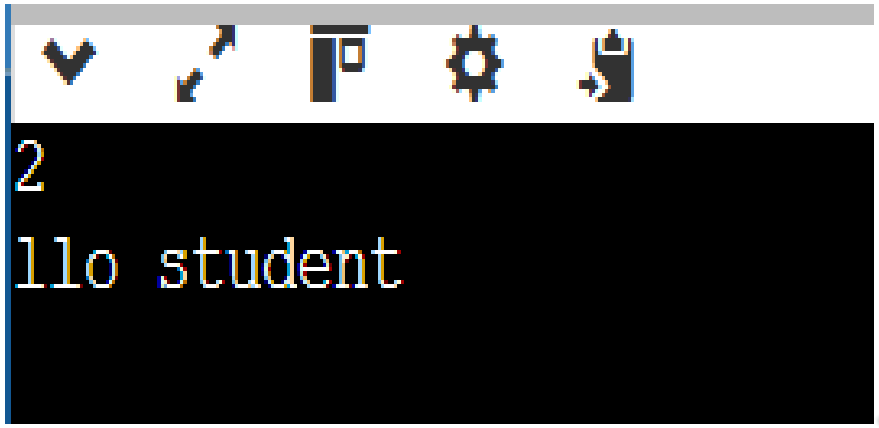
2: sets the reference point at the end of the file

By default from_what argument is set to 0.

Gateway classes

main.py example.txt

```
1  f = open("example.txt", "r")
2  f.seek(2)
3  print(f.tell())
4  print(f.readline())
5  f.close()
```



main.py example.txt

```
1  hello student
```

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

main.py example.txt

```
1  f = open("example.txt", "r+")
2  f.write("this isthe first line\n")
3  f.seek(0)
4  f.write("this is the new content\n")
5  f.seek(0)
6  content=f.read()
7  print(content)
8
9
```

main.py example.txt

```
1  this is the new content
2
```



Gateway classes

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

main.py example.txt

```
1 with open("example.txt","r+")as file:
2     file.write("this is the first line\n")
3     file.write("this is te 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)
```

input

```
this is the first lithis is the new second line
```

main.py example.txt

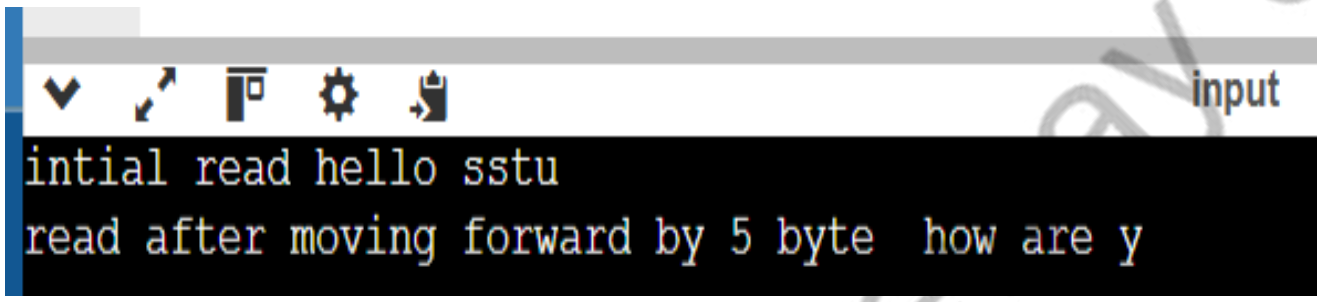
```
1 this is the first lithis is the new second line
2
```

Gateway classes

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

main.py | example.txt

```
1  with open("example.txt","r+")as file:
2      print("intial read",file.read(10))
3      current_position=file.tell()
4      file.seek(current_position+5)
5      print("read after moving forward by 5 byte",file.read(10))
6
7
```



main.py | example.txt

```
1  hello sstudents how are you?
2  i an fine
3
```

Gate Academy classes

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

main.py example.txt

```
1  with open("example.txt","r+")as file:
2      file.write("this is the intial content\n")
3      file.seek(0,2)
4      end_position=file.tell()
5      file.seek(end_position-10)
6      file.write("new contentat position\n")
7      file.seek(0)
8      content=file.read()
9      print(content)
```

main.py example.txt

```
1  how are you
2   i am fine
3   hi students
4    bbytearray
5    hi
6
```

```
this is the intial content
?
new contentat position
```

Gateway classes


Thank
you