FILE HANDLING WITH PYTHON

CLASS XII

FILE HANDLING PART-II

FILE OPERATIONS

- Opening a File (**open()** function)
- Closing a File (**close()** method)
- Reading for File (**read()** method)
- Writing to a File (**write()** method
- Appending to a File
- Other File Object Methods

OPENING A FILE

The foundation of file operations is the **open()** function. It establishes a connection between your Python program and a file.

Syntax to open a file

file_object = open(filename, mode)

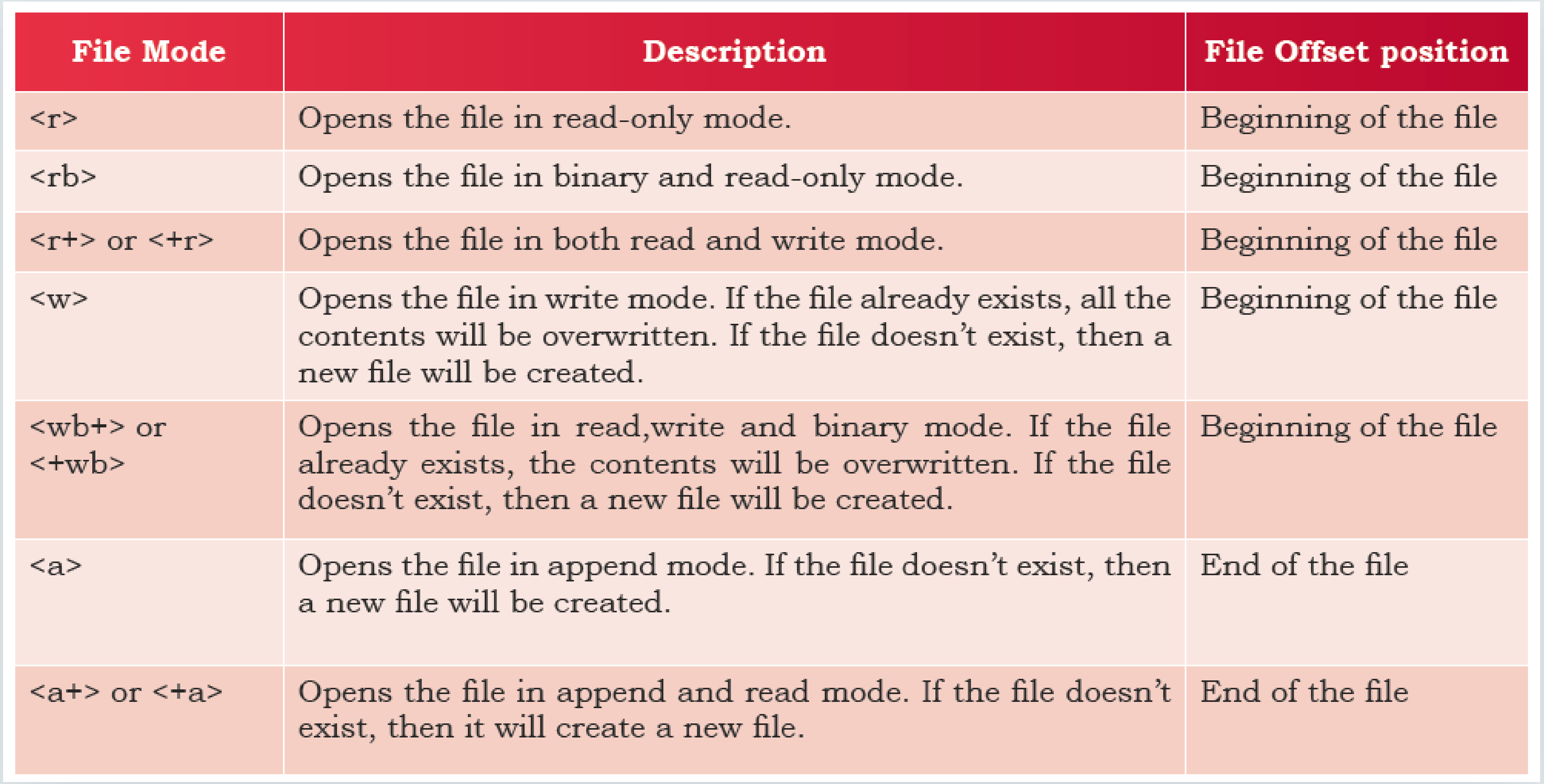
filename: A string specifying the path to the file (absolute or relative).

mode: A string indicating the purpose of opening the file. Common modes include:

'r': Read mode (default). Opens the file for reading.

'w': Write mode. Opens the file for writing. If the file exists, its content is truncated. If it doesn't exist, a new file is created.

File Open Modes



File Mode	Description	File Offset position
<r>	Opens the file in read-only mode.	Beginning of the file
<rb>	Opens the file in binary and read-only mode.	Beginning of the file
<r+> or <+r>	Opens the file in both read and write mode.	Beginning of the file
<w>	Opens the file in write mode. If the file already exists, all the contents will be overwritten. If the file doesn't exist, then a new file will be created.	Beginning of the file
<wb+> or <+wb>	Opens the file in read,write and binary mode. If the file already exists, the contents will be overwritten. If the file doesn't exist, then a new file will be created.	Beginning of the file
<a>	Opens the file in append mode. If the file doesn't exist, then a new file will be created.	End of the file
<a+> or <+a>	Opens the file in append and read mode. If the file doesn't exist, then it will create a new file.	End of the file

FILE OPERATIONS ON TEXT FILE

Opening a Text File (open() function with text modes)

'r' (Read): Opens the file for reading. The file pointer is at the beginning.

'w' (Write): Opens the file for writing. If the file exists, its content is truncated. If it doesn't exist, a new file is created.

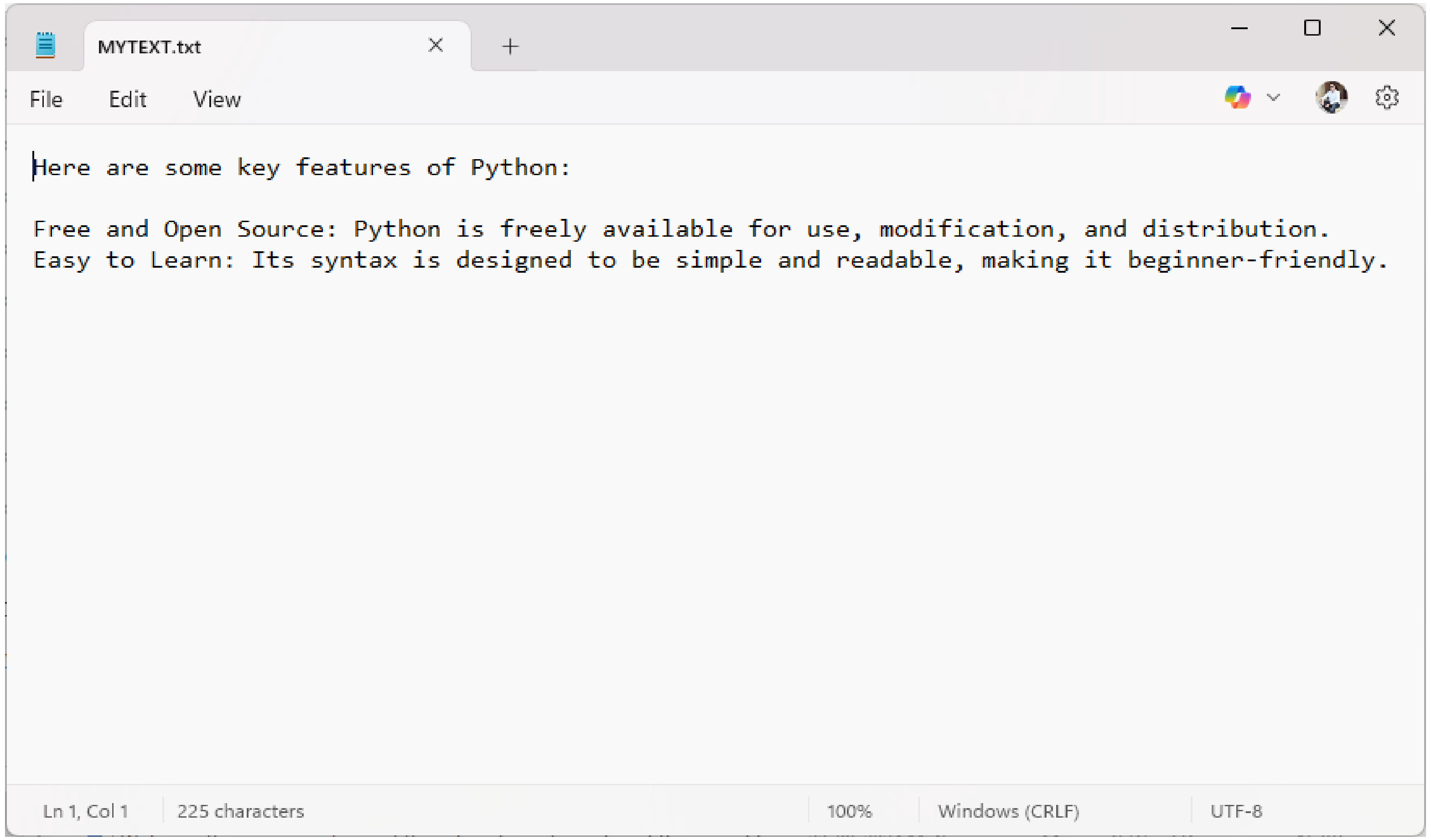
'a' (Append): Opens the file for writing, adding new data to the end. Creates a new file if it doesn't exist.

'r+' (Read and Write): Opens the file for both reading and writing. The file pointer is at the beginning.

'w+' (Write and Read): Opens the file for both writing and reading. Truncates the file if it exists, otherwise creates a new file. The file pointer is at the beginning.

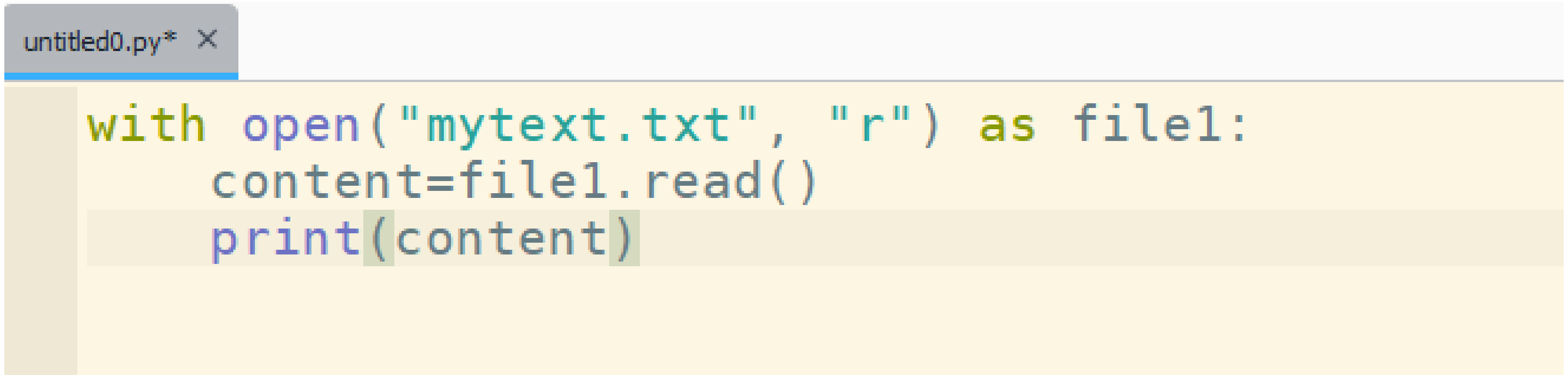
'a+' (Append and Read): Opens the file for both appending and reading. The file pointer is at the end if the file exists. Creates a new file if it doesn't exist.

Opening a file using ***“with”*** clause

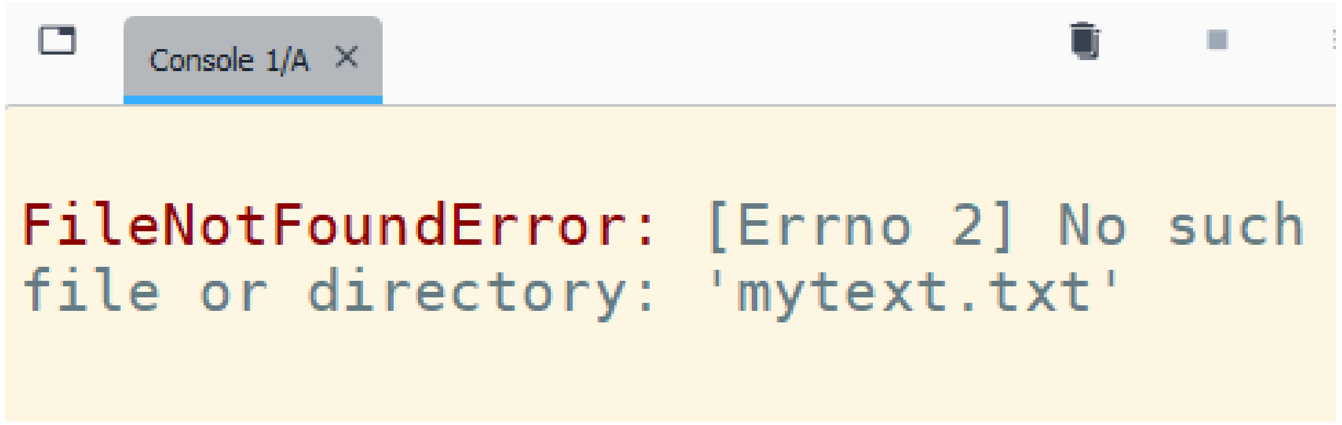


```
Here are some key features of Python:

Free and Open Source: Python is freely available for use, modification, and distribution.
Easy to Learn: Its syntax is designed to be simple and readable, making it beginner-friendly.
```

Opening a file using ***"with"*** clause

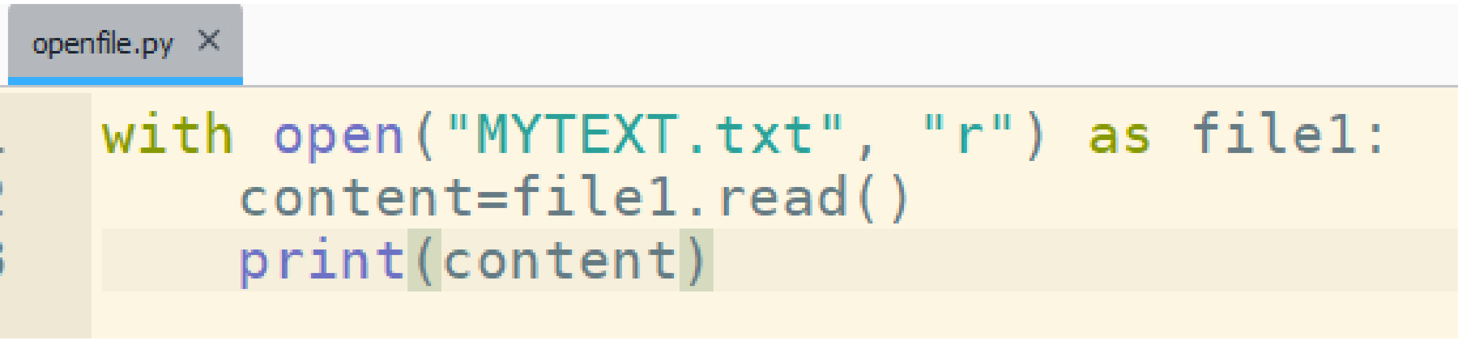


```
with open("mytext.txt", "r") as file1:
    content=file1.read()
    print(content)
```



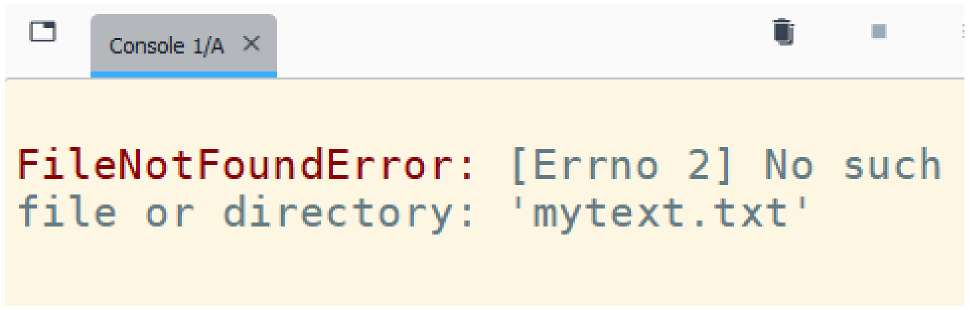
```
FileNotFoundError: [Errno 2] No such
file or directory: 'mytext.txt'
```

Opening a file using *“with”* clause



openfile.py

```python
with open("MYTEXT.txt", "r") as file1:
    content=file1.read()
    print(content)
```



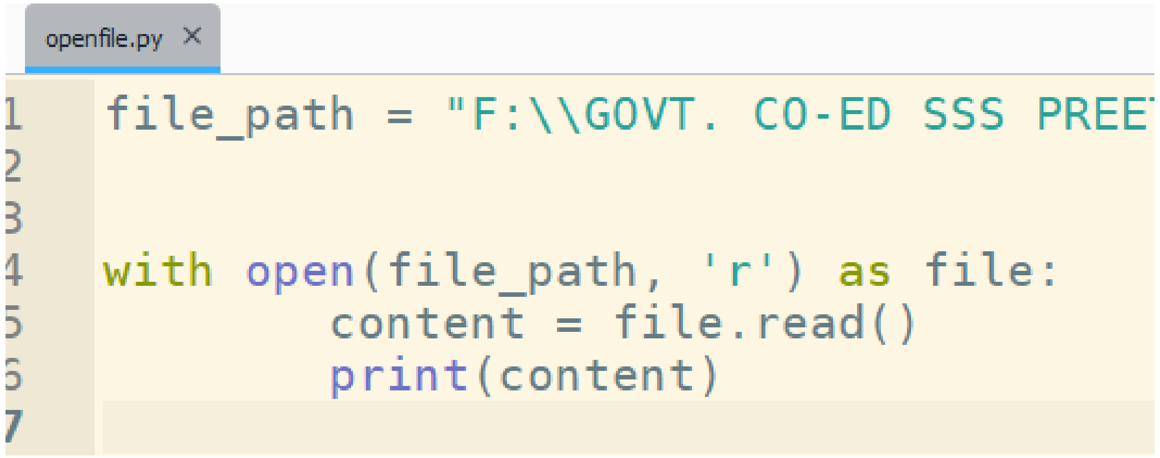
Console 1/A

```
FileNotFoundError: [Errno 2] No such
file or directory: 'mytext.txt'
```

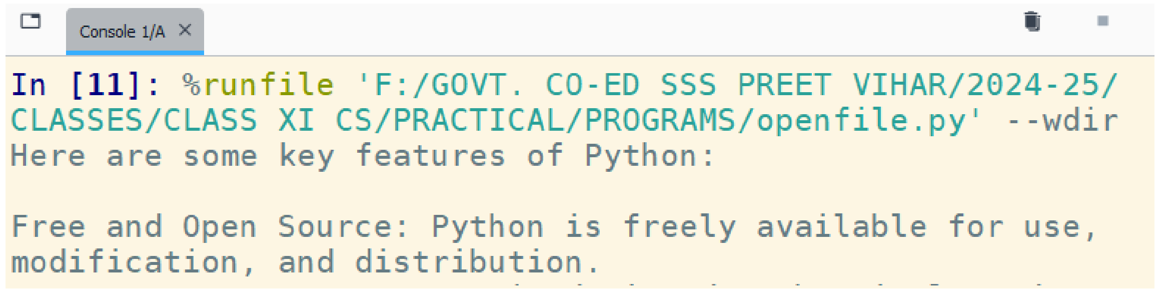
- Python file and file to be opened are stored on different location
- The Absolute path of the file is not given

Opening a file using *"with"* clause

- Absolute File Path
- file_path = "F:\\GOVT. CO-ED SSS PREET VIHAR\\2024-25\\CLASSES\\CLASS XI CS\\PRACTICAL\\PROGRAMS\\MYTEXT.txt"


openfile.py

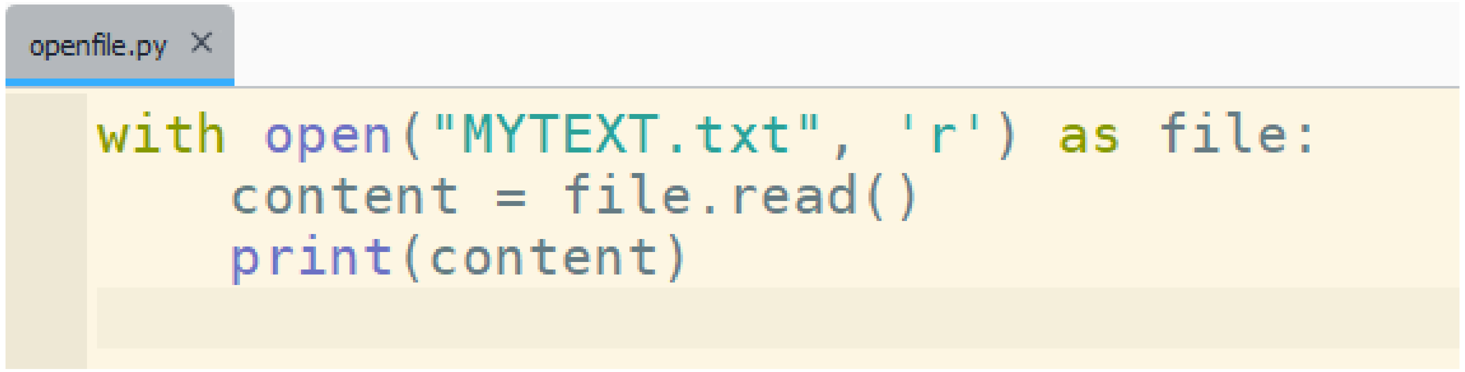
```python
file_path = "F:\\GOVT. CO-ED SSS PREE

with open(file_path, 'r') as file:
        content = file.read()
        print(content)

```


Console 1/A

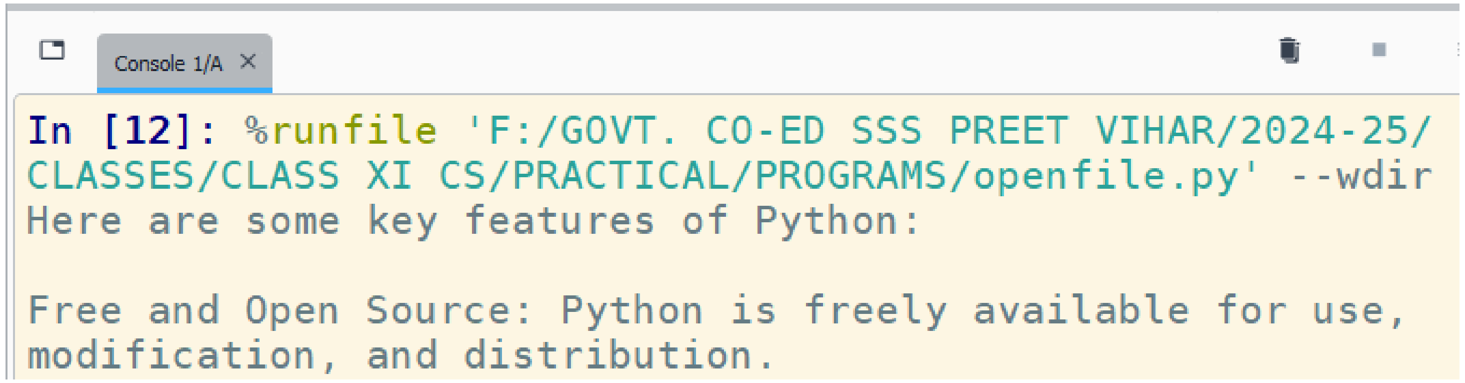
```
In [11]: %runfile 'F:/GOVT. CO-ED SSS PREET VIHAR/2024-25/
CLASSES/CLASS XI CS/PRACTICAL/PROGRAMS/openfile.py' --wdir
Here are some key features of Python:

Free and Open Source: Python is freely available for use,
modification, and distribution.
```

Opening a file using ***“with”*** clause (Relative Path)

Both the Files are in same folder

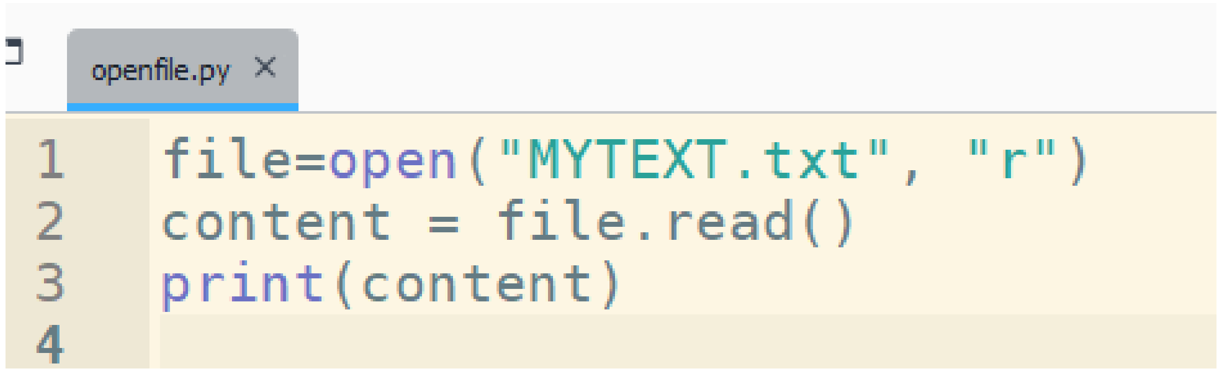

openfile.py

```
with open("MYTEXT.txt", 'r') as file:
    content = file.read()
    print(content)
```

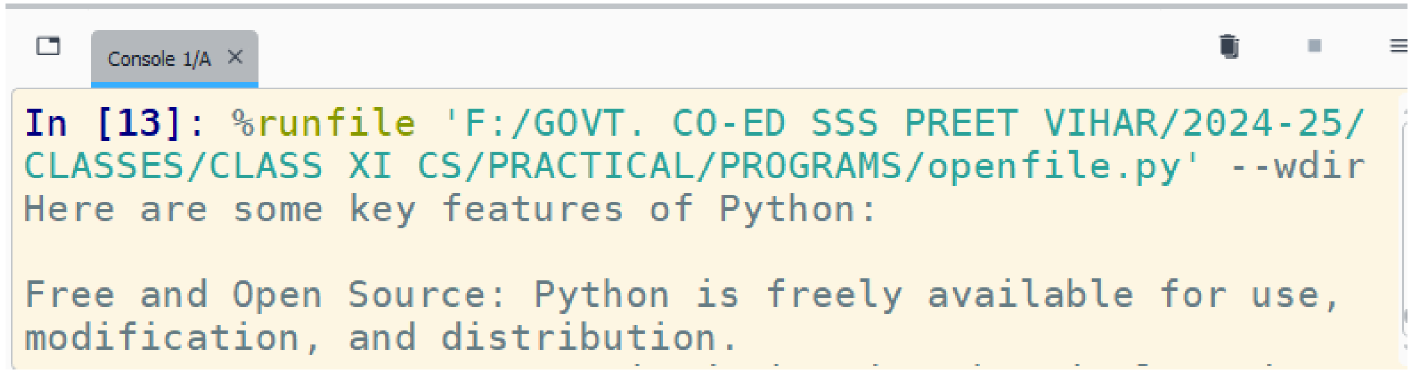

Console 1/A

```
In [12]: %runfile 'F:/GOVT. CO-ED SSS PREET VIHAR/2024-25/
CLASSES/CLASS XI CS/PRACTICAL/PROGRAMS/openfile.py' --wdir
Here are some key features of Python:

Free and Open Source: Python is freely available for use,
modification, and distribution.
```

Opening a file using ***"open()"*** clause



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



```
In [13]: %runfile 'F:/GOVT. CO-ED SSS PREET VIHAR/2024-25/
CLASSES/CLASS XI CS/PRACTICAL/PROGRAMS/openfile.py' --wdir
Here are some key features of Python:

Free and Open Source: Python is freely available for use,
modification, and distribution.
```

Read the Data

- Once the file is open, you use specific methods or functions to read the data from it. The way you read the data can vary:
  - **Read the entire file:** Read all the content at once into a single variable.
  - **Read line by line:** Read the file one line at a time.
  - **Read a specific number of characters or bytes:** Read a chunk of the file.

Read the Data (Methods to read data)

- **read():**
  - Reads the entire content of the file as a single string. Use this for smaller files.
- **readline():**
  - Reads one line from the file, including the trailing newline character (\n). Returns an empty string when the end of the file is reached. Useful for processing files line by line.
- **readlines():**
  - Reads all lines from the file and returns them as a list of strings, where each string represents a line (including the newline character). Suitable for smaller to medium-sized files where you need to access lines by index.

read() method

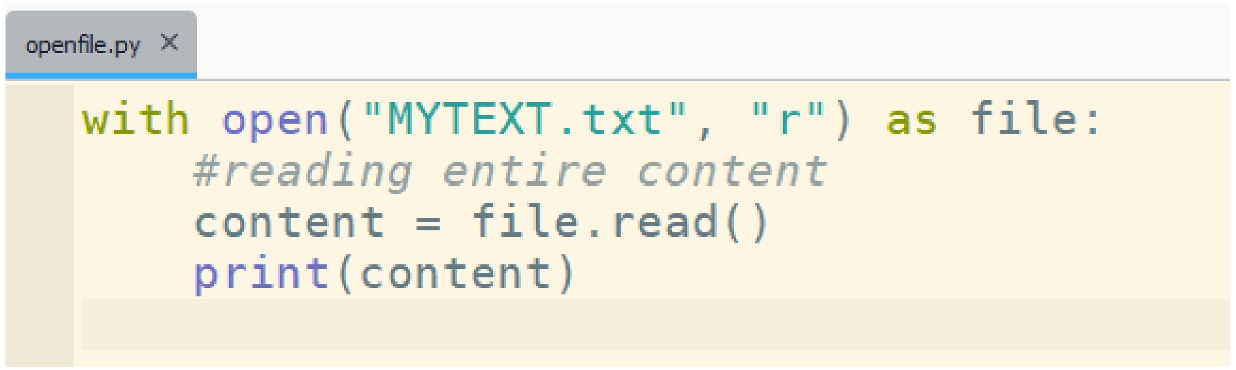
- The ***read()*** methods are used to retrieve data from a file object that has been opened for reading (usually with the mode 'r' or 'rb’)
- ***read([size]):***
- Purpose:
  - Reads the entire content of the file or a specified number of characters/bytes from the current file position.
- Return Value:
  - **In text mode ('r'):** Returns a single string containing the entire content read. If the end of the file (EOF) is reached, it returns an empty string (‘’).
  - **In binary mode ('rb'):** Returns a bytes object containing the entire content read. If EOF is reached, it returns an empty bytes object (b'').

read() method

- **Optional Argument**: ***size***:
  - If the size argument is provided (a non-negative integer), read() attempts to read at most size characters (in text mode) or size bytes (in binary mode) from the file.
  - If size is omitted or negative, it reads the entire file content until EOF.
- **Use Cases:**
  - Reading the entire content of a small to medium-sized file into a single string or bytes object.
  - Reading a specific number of characters or bytes at a time for processing in chunks.

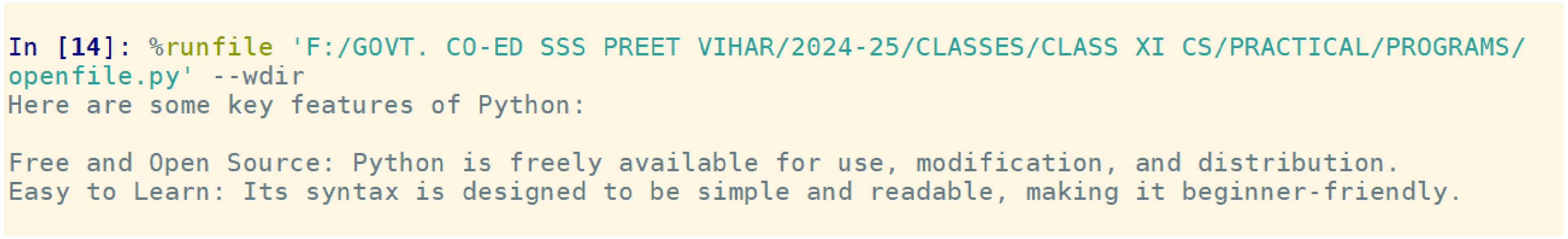
Example (Text Mode): Entire content

Not passing any argument to read() method for reading entire content of file



openfile.py

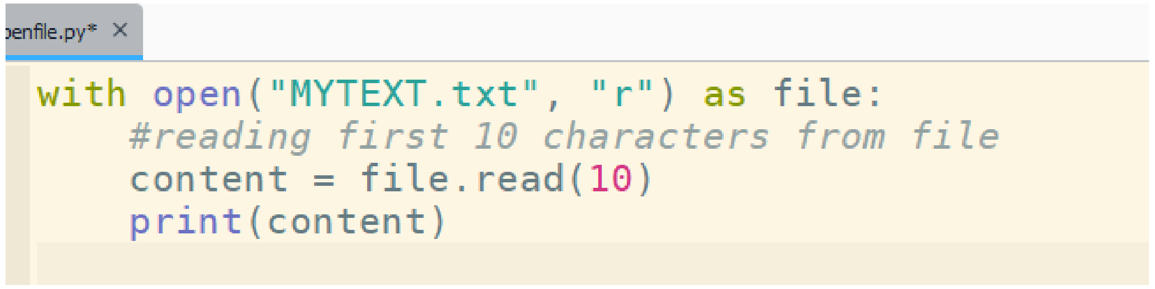
```python
with open("MYTEXT.txt", "r") as file:
    #reading entire content
    content = file.read()
    print(content)
```



```
In [14]: %runfile 'F:/GOVT. CO-ED SSS PREET VIHAR/2024-25/CLASSES/CLASS XI CS/PRACTICAL/PROGRAMS/
openfile.py' --wdir
Here are some key features of Python:

Free and Open Source: Python is freely available for use, modification, and distribution.
Easy to Learn: Its syntax is designed to be simple and readable, making it beginner-friendly.
```

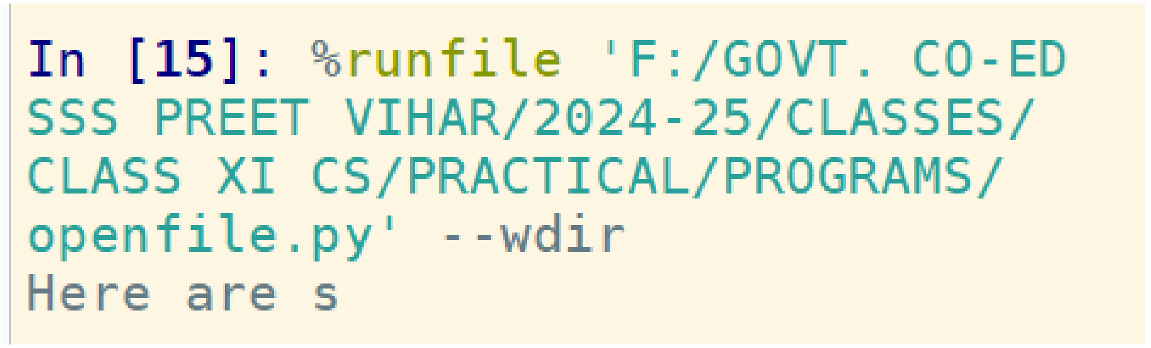
Example (Text Mode): First 10 chars.

Passing 10 as argument in read() method to read first 10 characters



penfile.py* ×

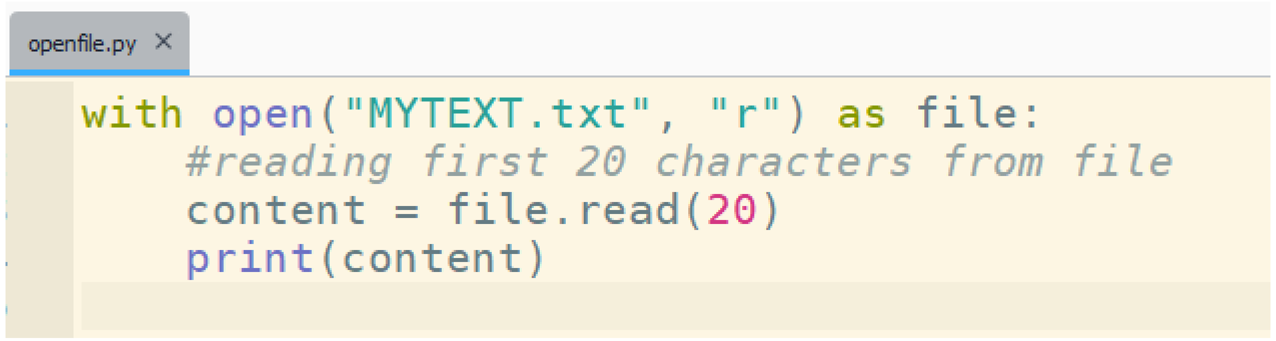
```
with open("MYTEXT.txt", "r") as file:
    #reading first 10 characters from file
    content = file.read(10)
    print(content)
```



```
In [15]: %runfile 'F:/GOVT. CO-ED SSS PREET VIHAR/2024-25/CLASSES/CLASS XI CS/PRACTICAL/PROGRAMS/openfile.py' --wdir
Here are s
```

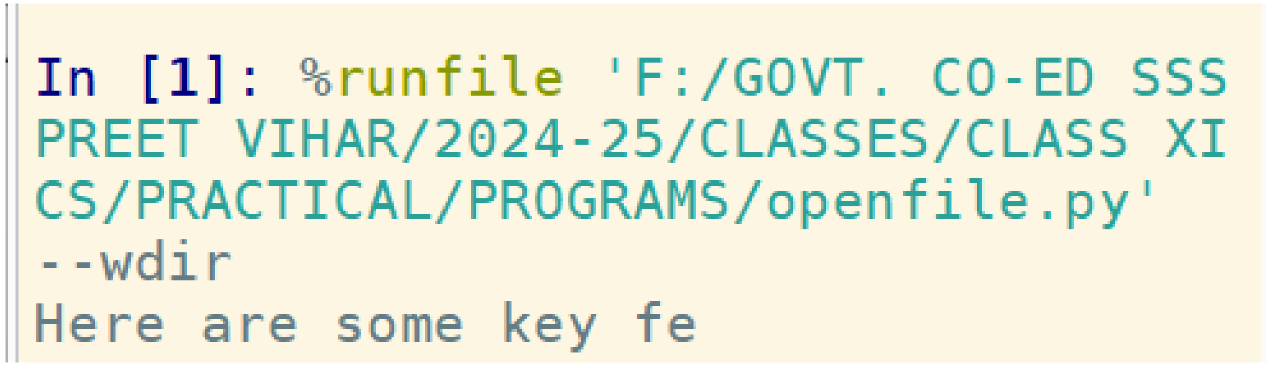
Example (Text Mode): First 20 chars.

Passing 20 as argument in read() method to read first 10 characters



openfile.py

```python
with open("MYTEXT.txt", "r") as file:
    #reading first 20 characters from file
    content = file.read(20)
    print(content)
```



```
In [1]: %runfile 'F:/GOVT. CO-ED SSS
PREET VIHAR/2024-25/CLASSES/CLASS XI
CS/PRACTICAL/PROGRAMS/openfile.py'
--wdir
Here are some key fe
```

readline(): Method

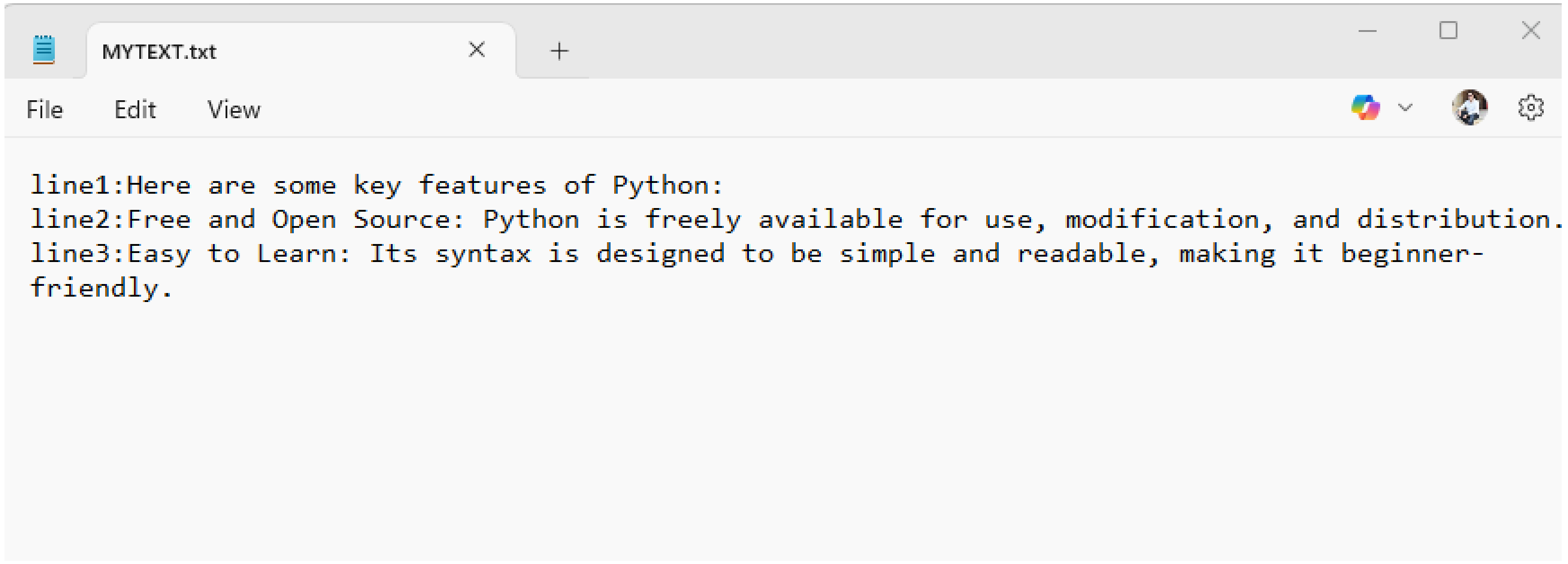
Purpose:

- Reads one entire line from the file, including the trailing newline character (\n) at the end of the line (if present).

Return Value:

- **In text mode ('r'):** Returns the line as a string. If EOF is reached, it returns an empty string (‘’).
- **In binary mode ('rb'):** Returns the line as a bytes object, including the trailing bytes representing the newline sequence (e.g., b'\r\n' or b'\n'). Returns an empty bytes object (b'') at EOF.

MYTEXT.txt file



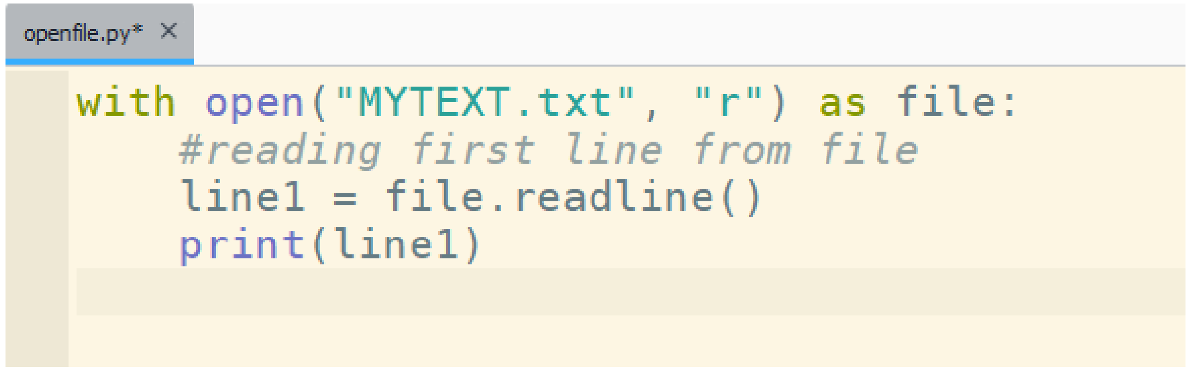
```
line1:Here are some key features of Python:
line2:Free and Open Source: Python is freely available for use, modification, and distribution.
line3:Easy to Learn: Its syntax is designed to be simple and readable, making it beginner-
friendly.
```

readline(): Method

- **No Argument:**
  - readline() does not take any size argument.
  - It reads until it encounters a newline or EOF.
- **Use Cases:**
  - Reading a file line by line, which is memory-efficient for large files.
  - Processing files where data is organized into lines.

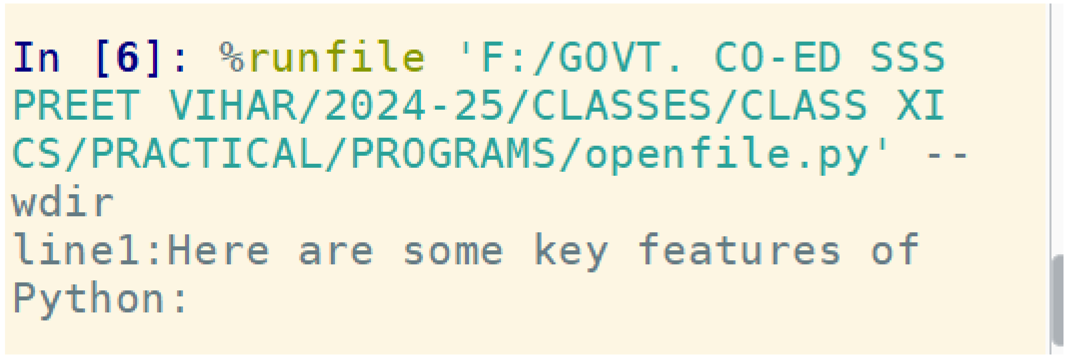
Example (Text Mode): first line

Reading First line from file



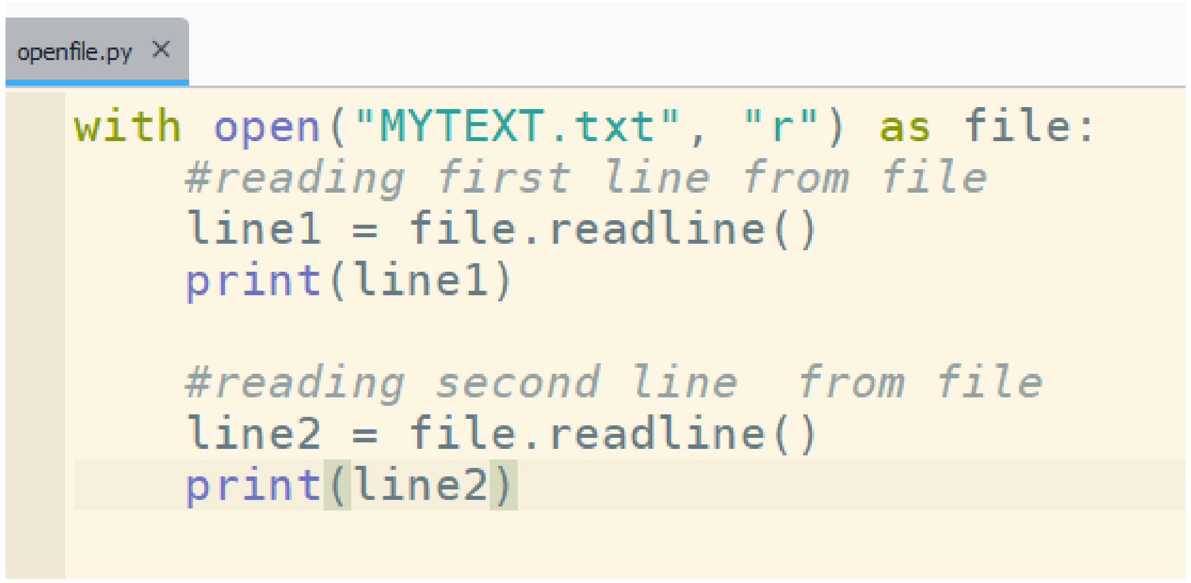
openfile.py*

```python
with open("MYTEXT.txt", "r") as file:
    #reading first line from file
    line1 = file.readline()
    print(line1)
```

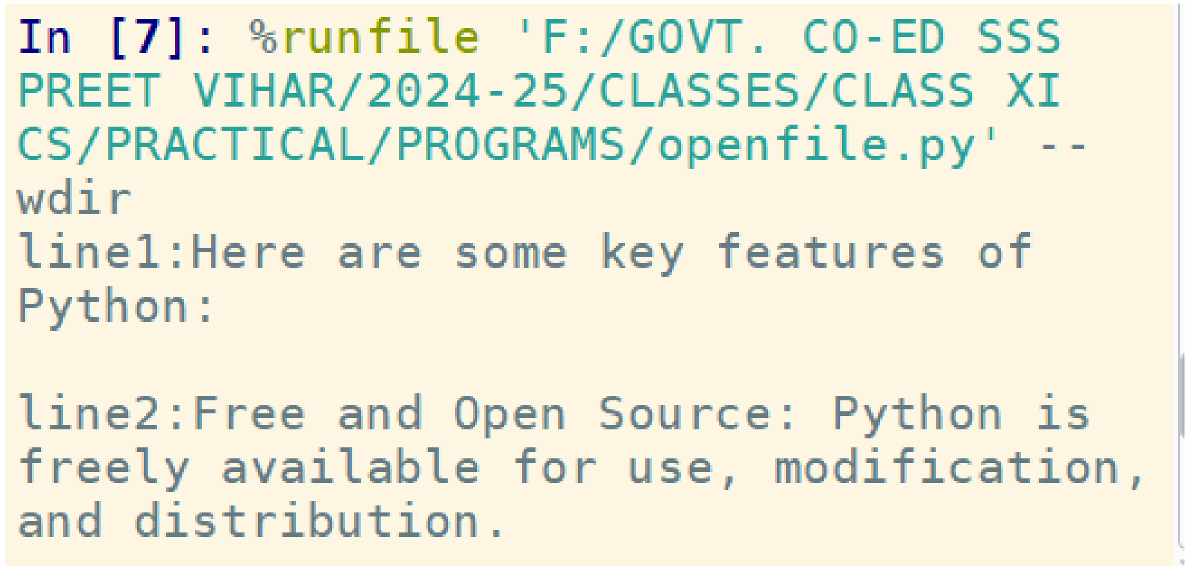


```
In [6]: %runfile 'F:/GOVT. CO-ED SSS
PREET VIHAR/2024-25/CLASSES/CLASS XI
CS/PRACTICAL/PROGRAMS/openfile.py' --
wdir
line1:Here are some key features of
Python:
```

Example (Text Mode): first two lines line

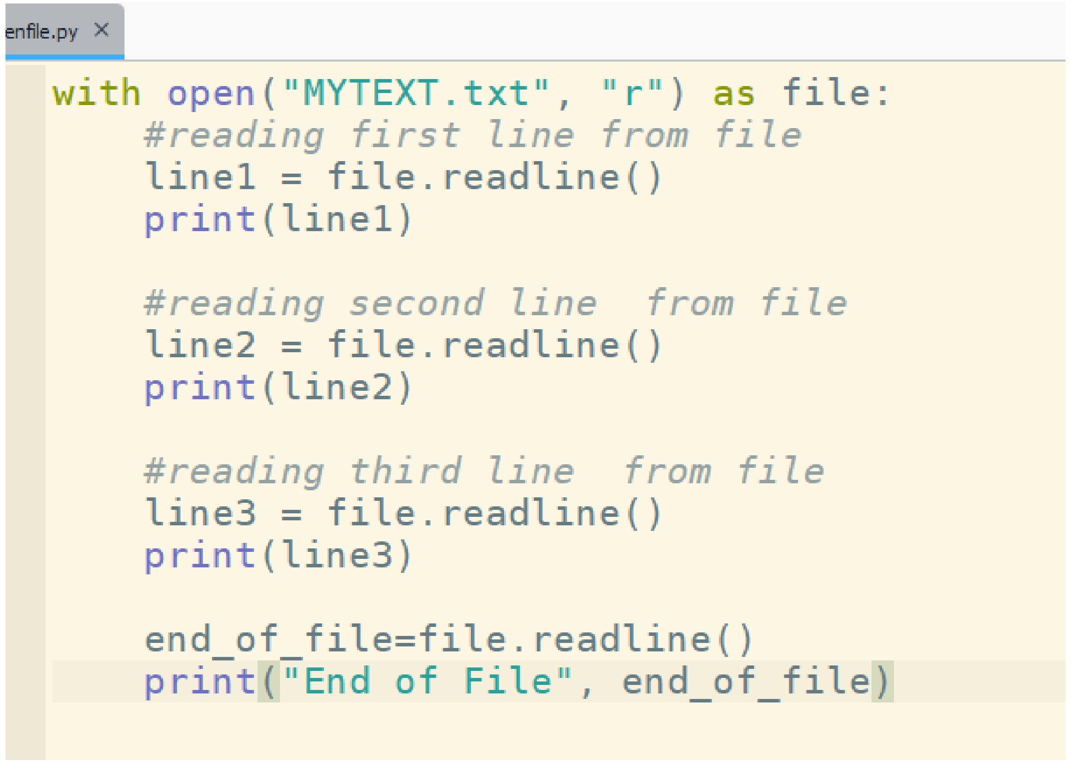

openfile.py

```python
with open("MYTEXT.txt", "r") as file:
    #reading first line from file
    line1 = file.readline()
    print(line1)

    #reading second line  from file
    line2 = file.readline()
    print(line2)
```

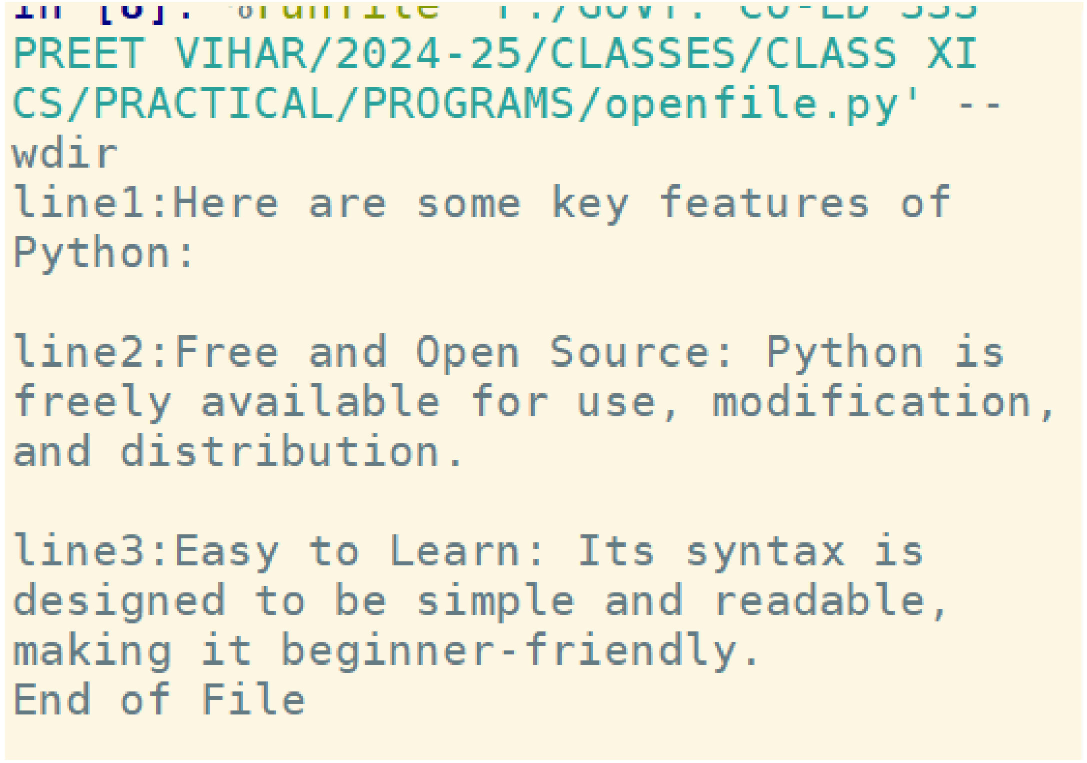


```
In [7]: %runfile 'F:/GOVT. CO-ED SSS
PREET VIHAR/2024-25/CLASSES/CLASS XI
CS/PRACTICAL/PROGRAMS/openfile.py' --
wdir
line1:Here are some key features of
Python:

line2:Free and Open Source: Python is
freely available for use, modification,
and distribution.
```

Example (Text Mode): All lines one by one line and end of file



```
with open("MYTEXT.txt", "r") as file:
    #reading first line from file
    line1 = file.readline()
    print(line1)

    #reading second line  from file
    line2 = file.readline()
    print(line2)

    #reading third line  from file
    line3 = file.readline()
    print(line3)

    end_of_file=file.readline()
    print("End of File", end_of_file)
```



```
PREET VIHAR/2024-25/CLASSES/CLASS XI
CS/PRACTICAL/PROGRAMS/openfile.py' --
wdir
line1:Here are some key features of
Python:

line2:Free and Open Source: Python is
freely available for use, modification,
and distribution.

line3:Easy to Learn: Its syntax is
designed to be simple and readable,
making it beginner-friendly.
End of File
```

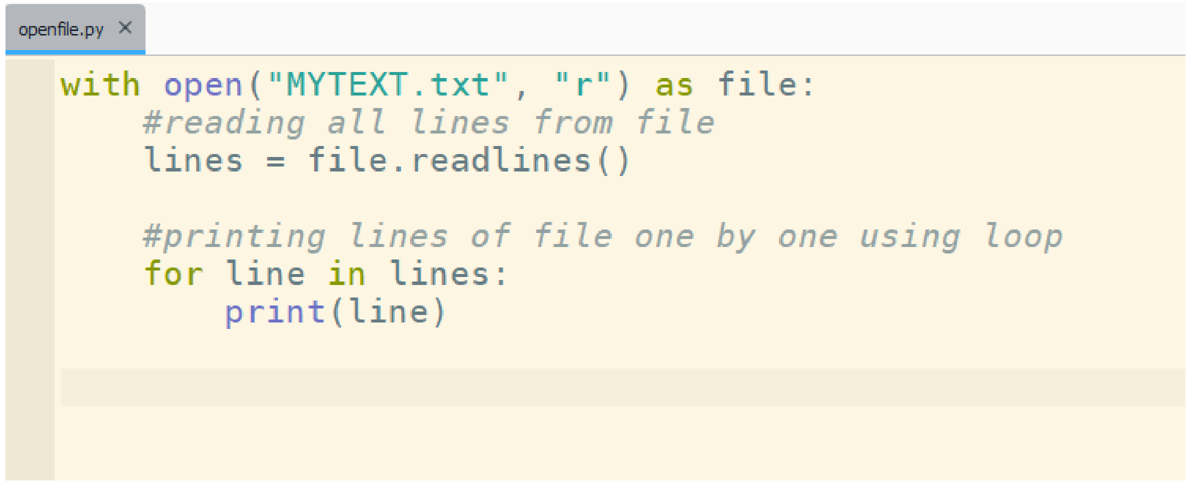
readlines(): Method

Purpose:	Reads all the lines in the file and returns them as a list of strings (in text mode) or a list of bytes objects (in binary mode). Each item in the list represents one line, including the trailing newline character (if present).
Return Value:	A list of strings (text mode) or bytes objects (binary mode), where each element is a line from the file. Returns an empty list ([ ]) if the file is empty or if the file pointer is already at EOF.

readlines(): Method

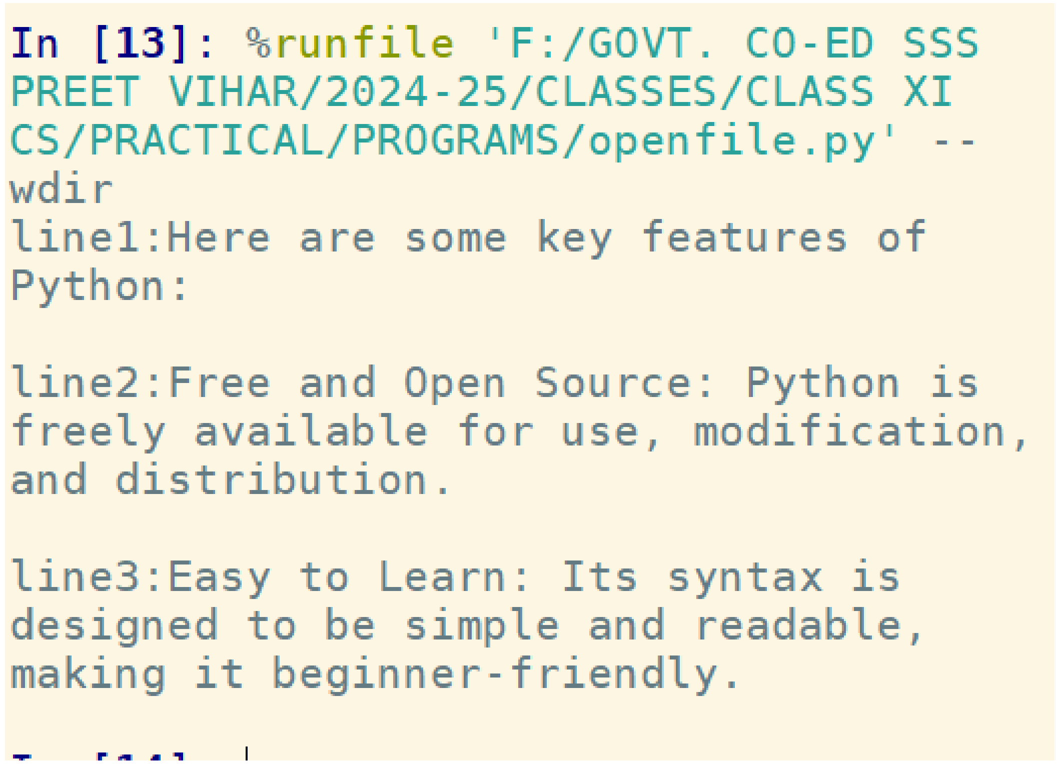
No Argument:	readlines() does not take any size argument.
Use Cases:	Reading the entire content of a file into a list of lines for further processing. Useful when you need to access specific lines by index or iterate over all lines multiple times (though iterating over the file object directly is often more memory-efficient for large files).

Example (Text Mode):



openfile.py

```
with open("MYTEXT.txt", "r") as file:
    #reading all lines from file
    lines = file.readlines()

    #printing lines of file one by one using loop
    for line in lines:
        print(line)
```



```
In [13]: %runfile 'F:/GOVT. CO-ED SSS
PREET VIHAR/2024-25/CLASSES/CLASS XI
CS/PRACTICAL/PROGRAMS/openfile.py' --
wdir
line1:Here are some key features of
Python:

line2:Free and Open Source: Python is
freely available for use, modification,
and distribution.

line3:Easy to Learn: Its syntax is
designed to be simple and readable,
making it beginner-friendly.
```

Choosing the Right *read()* Method:

read()	• when you need the entire file content as a single string or bytes object, or when you want to read a specific chunk of data. Be cautious with very large files as it can consume a lot of memory.
readline()	• when you need to process the file line by line and want explicit control over each line.
readlines()	• when you need to store all lines in a list for later access or manipulation. However, for very large files, iterating over the file object directly is generally preferred for memory efficiency.