File IO

Method

open()

> `Opens` a file.
> takes two parameters; filename, and mode.

```
file = open("example.txt", "w")
print(file)
```

Modes

Mode	Description
`r`	Open a file for reading. (default)
`w`	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
`x`	Open a file for exclusive creation. If the file already exists, the operation fails.
`a`	Open a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.
`t`	Open in text mode. (default)
`b`	Open in binary mode.
`+`	Open a file for updating (reading and writing)

close()

> `Closes` an opened file. It has no effect if the file is already closed.

```
file = open("example.txt", "w")
file.close()
```

detach()

> `Separates the underlying binary buffer` from the TextIOBase and returns it.

```
file = open("example.txt", "w")
buffer = file.detach()
```

fileno()

> Returns an `integer number (file descriptor)` of the file.

```
file = open("example.txt", "r")
fd = file.fileno()
```

flush()

> `Flushes` the write buffer of the file stream.

```
file = open("example.txt", "w")
file.write("Hello, World!")
file.flush()
```

isatty()

> Returns True if the `file stream is interactive`.

```
file = open("example.txt", "r")
interactive = file.isatty()
```

read(`n`)

> Reads at `most n characters` from the file. Reads till the end of the file if n is negative or None.

```
file = open("example.txt", "r")
content = file.read(10)
```

readable()

> Returns True if the `file stream can be read from`.

```
file = open("example.txt", "r")
is_readable = file.readable()
```

readline(`n` =-1)

> Reads and returns `one line from the file`. Reads in at most n bytes if specified.

```python
file = open("example.txt", "r")
line = file.readline()
```

readlines(`n` =-1)

> Reads and returns a `list of lines from the file` . Reads in at most n bytes/characters if specified.

```python
file = open("example.txt", "r")
lines = file.readlines()
```

seek(`offset` , `from` = `SEEK_SET`)

> Changes the `file position to offset bytes` , in reference to from (start, current, end).

```python
file = open("example.txt", "r")
file.seek(5, 0)  # Seek to the 5th byte from the beginning of the file
```

seekable()

> Returns True `if the file stream supports random access` .

```python
file = open("example.txt", "r")
is_seekable = file.seekable()
```

tell()

> Returns an `integer that represents the current position of the file's object` .

```python
file = open("example.txt", "r")
position = file.tell()
```

truncate(`size` = `None`)

> `Resizes` the `file stream to size bytes` . If size is not specified, resizes to the current location.

```python
file = open("example.txt", "r+")
file.truncate(10)
```

writable()

> Returns True `if the file stream can be written to` .

```python
file = open("example.txt", "w")
is_writable = file.writable()
```

write(`s`)

> `Writes` the string s to the file and `returns the number of characters written` .

```python
file = open("example.txt", "w")
num_chars_written = file.write("Hello, World!")
```

writelines(`lines`)

> `Writes` a `list of lines` to the file.

```python
file = open("example.txt", "w")
lines = ["Line 1", "Line 2", "Line 3"]
file.writelines(lines)
```