Additional File Methods

In addition to the basic file reading and writing methods, Python's `open()` function provides several additional methods that offer more flexibility and control when working with files. Here are some of the additional file methods, along with examples:

1. **`seek(offset, whence)` Method:**
   The `seek()` method is used to change the file cursor position.

   ```python
   with open('example.txt', 'r') as file:
       file.seek(5)  # Move the cursor to the 6th byte
       content = file.read()
       print(content)
   ```

   This example moves the cursor to the 6th byte in the file before reading its content.

2. **`tell()` Method:**
   The `tell()` method returns the current position of the file cursor.

   ```python
   with open('example.txt', 'r') as file:
       content = file.read(7)
       position = file.tell()  # Get the current position
       print(f'Content: {content}, Position: {position}')
   ```

   After reading 7 bytes, this example prints both the content and the current cursor position.

3. **`truncate(size=None)` Method:**
   The `truncate()` method is used to resize the file to the given size. If no size is specified, it truncates the file at the current position.

   ```python
   with open('example.txt', 'r+') as file:
       file.truncate(10)  # Truncate the file to 10 bytes
       content = file.read()
       print(content)
   ```

   This example truncates 'example.txt' to 10 bytes and then reads and prints the content.

4. **`flush()` Method:**
   The `flush()` method flushes the internal buffer, ensuring that all data is written to the file.

   ```python
   with open('example.txt', 'w') as file:
   ```

```
    file.write('Hello, World!')
    file.flush()  # Flush the buffer to ensure data is written immediately
```

The `flush()` method is useful when you want to make sure that data is written to the file immediately.

5. **`writelines(lines)` Method:**
   The `writelines()` method writes a list of lines to the file.

   ```python
   lines = ['Line 1\n', 'Line 2\n', 'Line 3\n']

   with open('multiline.txt', 'w') as file:
       file.writelines(lines)
   ```

   This example writes each line from the list to 'multiline.txt'.

6. **`readinto(buffer)` Method:**
   The `readinto()` method reads data from the file and stores it in a pre-allocated buffer.

   ```python
   buffer = bytearray(5)

   with open('example.txt', 'rb') as file:
       bytes_read = file.readinto(buffer)
       print(f'Bytes Read: {bytes_read}, Content: {buffer}')
   ```

   This example reads 5 bytes from 'example.txt' into a pre-allocated buffer.

These additional file methods provide more control and options when working with files in Python. Depending on your use case, you may find these methods useful for tasks such as positioning the cursor, truncating files, flushing buffers, or reading data into pre-allocated buffers.