

## 11.9. `linecache` — Random access to text lines

Source code: [Lib/linecache.py](#)

The `linecache` module allows one to get any line from a Python source file, while attempting to optimize internally, using a cache, the common case where many lines are read from a single file. This is used by the `traceback` module to retrieve source lines for inclusion in the formatted traceback.

The `tokenize.open()` function is used to open files. This function uses `tokenize.detect_encoding()` to get the encoding of the file; in the absence of an encoding token, the file encoding defaults to UTF-8.

The `linecache` module defines the following functions:

`linecache.getline(filename, lineno, module_globals=None)`

Get line *lineno* from file named *filename*. This function will never raise an exception — it will return `''` on errors (the terminating newline character will be included for lines that are found).

If a file named *filename* is not found, the function will look for it in the module search path, `sys.path`, after first checking for a [PEP 302](#) `__loader__` in *module\_globals*, in case the module was imported from a zipfile or other non-file-system import source.

`linecache.clearcache()`

Clear the cache. Use this function if you no longer need lines from files previously read using `getline()`.

`linecache.checkcache(filename=None)`

Check the cache for validity. Use this function if files in the cache may have changed on disk, and you require the updated version. If *filename* is omitted, it will check all the entries in the cache.

`linecache.lazycache(filename, module_globals)`

Capture enough detail about a non-file-based module to permit getting its lines later via `getline()` even if *module\_globals* is `None` in the later call. This avoids doing I/O until a line is actually needed, without having to carry the module globals around indefinitely.

*New in version 3.5.*

Example:

```
>>> import linecache
>>> linecache.getline(linecache.__file__, 8)
'import sys\n'
```

>>>