Python File I/O Cheat Sheet

Learn computer science with Python:
www.k0nze.dev

open()

Opens a file and creates a file object.

Syntax

```
open(file, mode='r', buffering=-1,
     encoding=None, errors=None,
     newline=None, closefd=True,
     opener=None)
```

Parameters

`file:` path-like object/string (absolute or relative) of the file to be opened

`mode:` see open() modes

`buffering:` **0** switch off buffering (only in binary mode)
1 enable line buffering (only in text mode)
>1 buffer size in bytes
default:
binary mode automatically determined buffer size (usually 4096-8192 bytes)
text mode uses line buffering

`encoding:` name of the encoding (e.g. `'utf-8'`) should only be used in text mode

`errors:` Error handling modes ('strict', 'ignore', 'replace', 'surrogateescape', 'xmlcharrefreplace', 'backslashreplace', 'namereplace')

`newline:` sets the newline mode
None universal newline mode all newline characters are converted to `'\n'`
other modes:`'\n'`,`'\r'`,`'\r\n'`

`closefd:` **`False`** a file descriptor was passed to as file and the file descriptor will be kept open when the file is closed.
True if a file name was passed to file otherwise an error is raised

`opener:` A custom file opener can be passed that must return an open file descriptor

Return Value

corresponding file object. If the file can not be opened an `OSError` is raised

```
>>> file = open("foobar.txt" ,'a+')
>>> file.write("foo")
>>> file.close()
>>> with open("foobar.txt", 'a+') as file:
...     file.write("bar")
>>> with open("foobar.txt", 'r') as file:
...     file.read()
  foobar
>>> with open("foobar.txt", 'rb') as file:
...     file.read()
...     file.read()
  b'foobar\n'
  ''
```

.close()

Closes an open file object.

Syntax

```
file.close()
```

Parameters

None

Return Value

None

```
>>> file = open("text.txt")
>>> file.close()
```

with

The keyword `with` is used when dealing with unmanaged resources where the code does not have full control over it, such as files.

`with` makes sure that after using a resource, everything is cleaned up correctly to prevent unwanted side effects, even when exceptions are raised.

`with` can replace `try/finally` blocks.

Syntax

```
with expression [as variable]:
    with-block
```

Parameters

`variable:` object that supports the context manage protocol having __enter__() and __exit__() methods

```
>>> with open("foobar.txt", 'r') as file
...     file.read()
  foobar
```

.isatty()

Returns if the file is interactive (i.e. a terminal/tty device).

Syntax

```
file.isatty()
```

Parameters

None

Return Value

`True` if the file is interactive

```
>>> with open("/dev/tty1") as tty:
...     tty.isatty()
  True
```

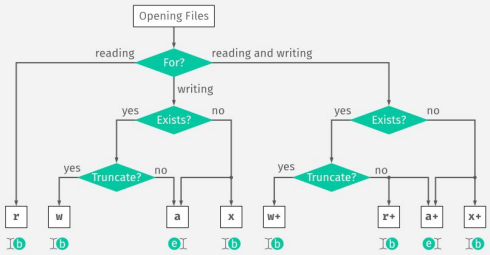
open() modes

Explaination

r	open for reading (default)
w	open for writing, truncate the file first
x	create a new file and open it for writing
a	open for writing, append to the of the file if it exists
b	binary mode
t	text mode (default)
+	open a disk file for updating (reading and writing)

Ib place file pointer at beginning of the file

eI place file pointer at end of the file



.readable()

Returns if an open file supports reading.

Syntax

```
file.readable()
```

Parameters

None

Return Value

`True` if the file supports reading. If `False`, `read()` will raise `OSError`

```
>>> d = {'a': 0, 'b': 1, 'c': 2, 'd': 3}
>>> d.values()
  dict_values([0, 1, 2, 3])
```

.writable()

Returns if an open file supports writing.

Syntax

```
file.writable()
```

Parameters

None

Return Value

`True` if the file supports writing. If `False`, `write()` and `truncate()` will raise `OSError`

```
>>> with open("foobar.txt", 'w') as file:
...     file.writable()
  True
```

.read()

Reads characters from a file and moves the file postion forward by the amount of read characters.

Syntax

```
file.read(size=-1)
```

Parameters

size: amount of characters to be read, if -1 characters until the end of file are read

Return Value

the characters read

```
>>> with open("foobar.txt", 'r') as file:
...     file.read(3)
  foo
```

.write()

Writes a string to a file and moves the file postion forward by the amount of written characters.

Syntax

```
file.write(s)
```

Parameters

s: string to write to the file

Return Value

amount of characters written

```
>>> with open("foobar.txt", 'w') as file:
...     file.write("barfoo")
  6
```

.readline()

Reads lines from a file terminated with a newline character or the end of file.

Syntax

```
file.readline(size=-1)
```

Parameters

size: at most size `characters` will be read, if -1 characters until the end of line are read

Return Value

the characters read

```
>>> with open("foobar.txt", 'r') as file:
...     file.readline()
  foobar
```

.writelines()

Writes lines to a file. No line separators are added to the end.

Syntax

```
file.writelines(lines)
```

Parameters

lines: list of lines to write to the file

Return Value

amount of characters written

```
>>> with open("foobar.txt", 'w') as file:
...     file.writelines(
          ["foobar", "barfoo"])
```

.flush()

Writes the buffer contents to the file.

Syntax

```
file.flush()
```

Parameters

None

Return Value

None

```
>>> with open("foobar.txt", mode='ab',
              buffering=128) as file:
...     file.write(b'foobar')
...     file.flush()
  6
```

.fileno()

Returns the underlying file descriptor.

Syntax

```
file.fileno()
```

Parameters

None

Return Value

Integer of the the file descriptor if it exists. Raises OSError if no file descriptor is used.

```
>>> with open("foobar.txt") as file:
...     file.fileno()
  3
```

.detach()

Separate the underlying raw stream from the buffer and return it.

Syntax

```
file.detach()
```

Parameters

None

Return Value

iterable containing all values of the dictionary

```
>>> file = open("foobar.txt", 'rb')
>>> file.detach()
  <_io.FileIO name='zen.txt' mode='rb'
  closefd=True>
```

.truncate()

Can extend or reduce the file size. In case of extension, the contents of the new file area depends on the operating system (usually zero bytes). Does not change the file postion.

Syntax

```
file.truncate(size=None)
```

Parameters

size: new file size in bytes, if None the new size is the current position

Return Value

if the new file size

```
>>> with open("foobar.txt", 'r+') as file:
...     file.truncate(2)
  2
```

Reading a File line by line

A while loop can be used to read a file line by line.

while loop over lines

```
>>> lines = []
>>> with open("foobar.txt") as file:
>>>     while True:
>>>         line = file.readline()
>>>         if not line:
>>>             break
>>>         lines.append(line)
>>> lines
  ['foo\n','bar']
```

Reading a JSON file as a dict

The json packages allows to convert a json file into a Python dictionary.

JSON File

```
{
  "agents": [
    {"name": "Laura Barton", "age": 34},...
  ],
  "avengers": [
    {"name": "Hawkeye", "age": 37}, ...
  ]
}
```

Example Code

```
>>> import json
>>> with open("mcu.json") as file:
...     data = json.load(file)
  data
  {'agents': [...],
   'avengers': [...]}
```

.seekable()

Returns if an open file supports random access.

Syntax

```
file.seekable()
```

Parameters

None

Return Value

`True` if the file supports random access. If `False`, `seek()`, `tell()`, and `truncate()` will raise `OSError`

```
>>> with open("foobar.txt", 'r') as file:
...     file.seekable()
  True
```

.seek()

Changes the file position by a given offset in bytes/ characters.

Syntax

```
file.seek(offset, whence=SEEK_SET)
```

Parameters

`offset:` is interpreted relative to the position indicated by whence

`whence:` **`SEEK_SET`** or **0** start of the file
`SEEK_CUR` or **1** current postion of the file
`SEEK_END` or **2** end of the file

Return Value

the new file position

```
>>> with open("foobar.txt", 'r') as file:
...     file.read(1)
...     file.seek(0, SEEK_END)
  'f'
  6
```

.tell()

Returns the current postion in a file.

Syntax

```
file.tell()
```

Parameters

None

Return Value

current position in a file as an integer.

```
>>> with open("foobar.txt", 'r') as file:
...     file.read(1)
...     file.tell()
  'a'
  1
```

k0nze builds
k0nze_
discord.k0nze.dev
patreon.com/k0nze

www.k0nze.dev