



open()

open a file

file: path to file (required)

mode: read/write/append, binary/text

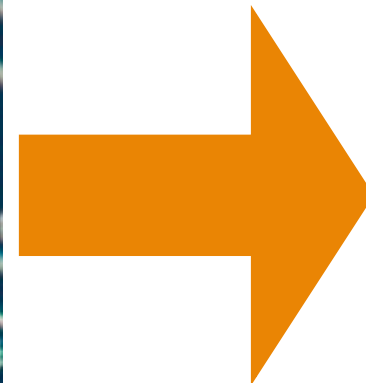
encoding: text encoding



Binary File Access

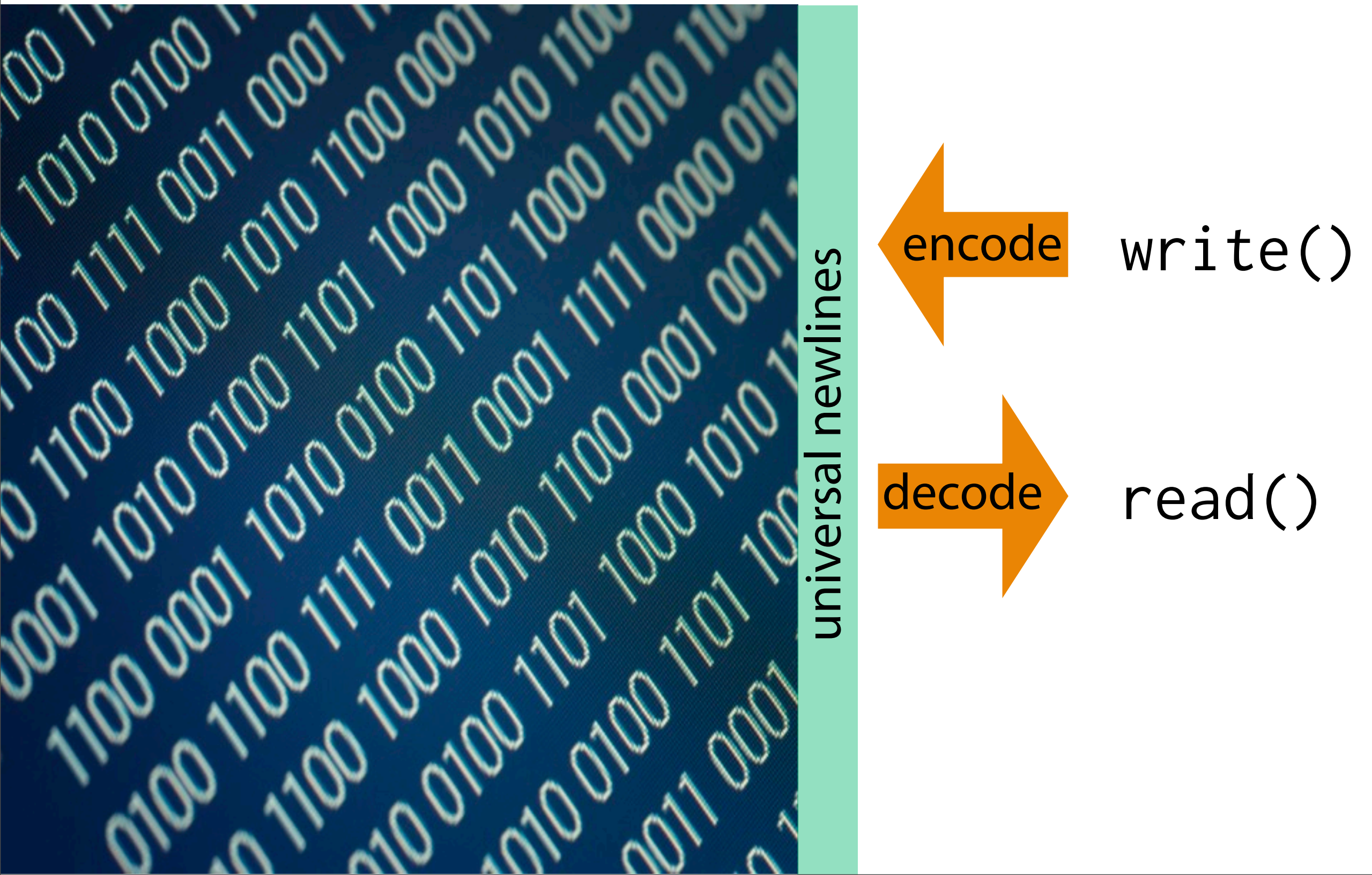


`write()`



`read()`

Text File Access



open() modes

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)
'U'	universal newlines mode (for backwards compatibility; should not be used in new code)

`write()` returns the
number of **codepoints**,
not the number of
characters.

Typical File Use

```
f = open()  
# work work work  
f.close()
```

*close() is required
to actually write the
data!*



with-block

resource cleanup with context-managers

`open()` returns a
context-manager!

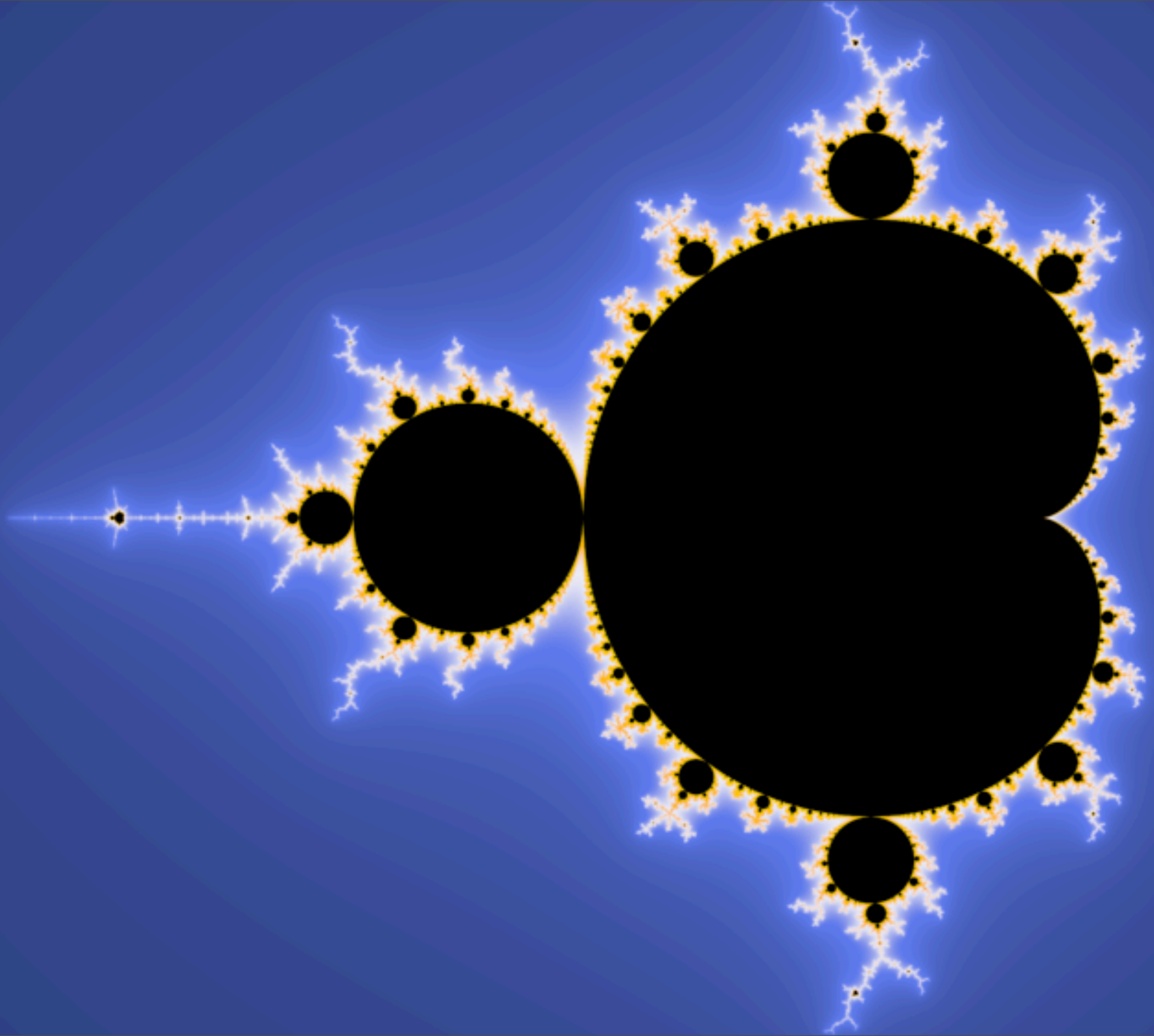
Moment of Zen

```
mgr = (EXPR)
exit = type(mgr).__exit__ # Not calling it yet
value = type(mgr).__enter__(mgr)
exc = True
try:
    try:
        VAR = value # Only if "as VAR" is present
        BLOCK
    except:
        # The exceptional case is handled here
        exc = False
        if not exit(mgr, *sys.exc_info()):
            raise
        # The exception is swallowed if exit() returns true
finally:
    # The normal and non-local-goto cases are handled here
    if exc:
        exit(mgr, None, None, None)
```



Binary files

Device independent bitmaps

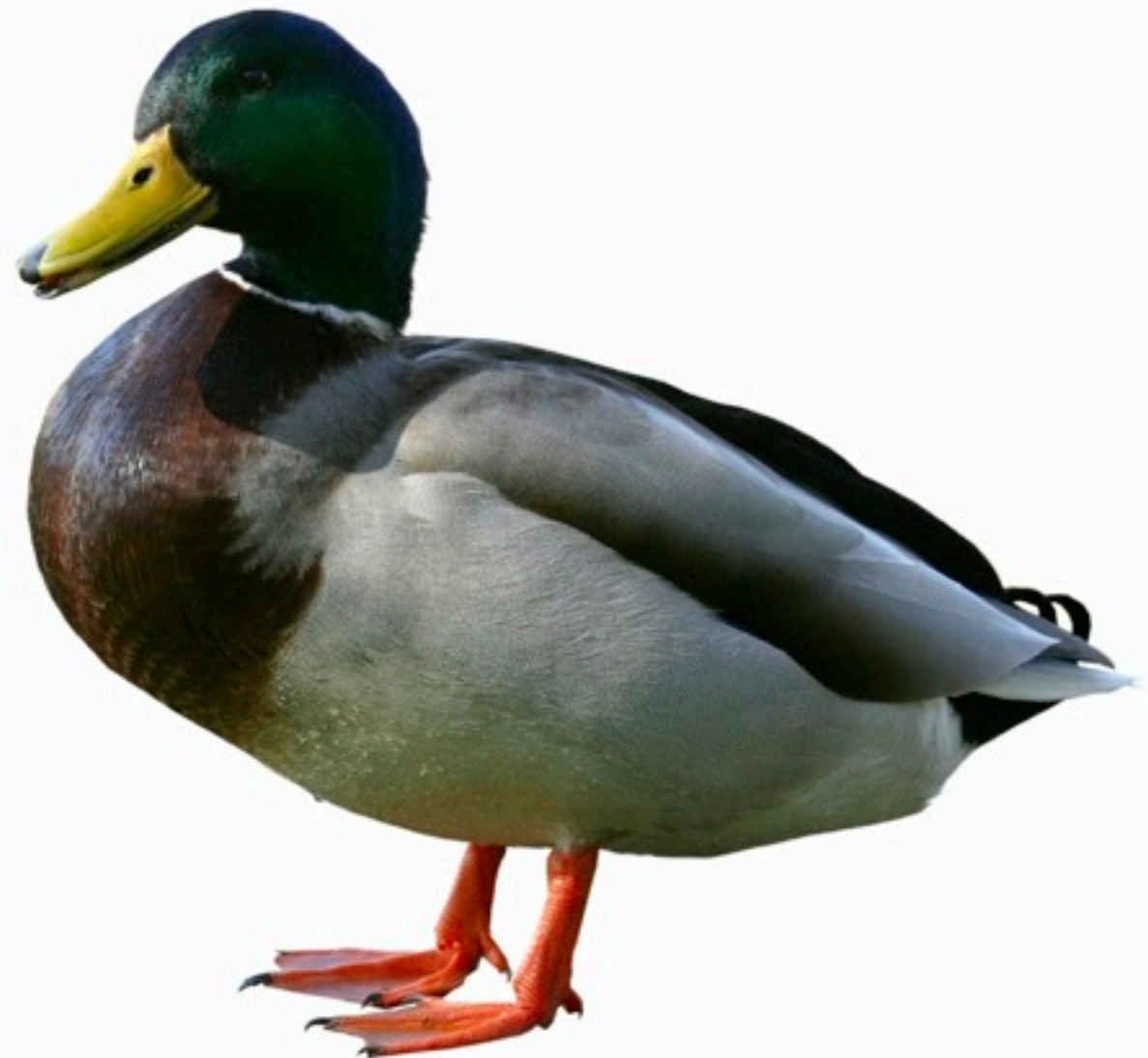




file-like objects

loosely-define set of behaviors for things that act like files

If it looks like
a file and reads
like a file, then it
is a file.





Files and resource management Summary

- Files are opened using the built-in `open()` function which accepts a file mode to control read/write/append behavior and whether the file is to be treated as raw binary or encoded text data
- For text data you should specify a text encoding
- Text files deal with string objects and perform universal newline translation and string encoding
- Binary files deal with `bytes` objects with no newline translation or encoding
- When writing files, it's our responsibility to provide newline characters for line breaks
- Files should always be closed after use
- Files provide various line-oriented methods for reading, and are also iterators which yield line by line
- Files are context managers and the `with`-statement can be used with context managers to ensure that clean up operations, such as closing files, are performed
- The notion of file-like-objects is loosely defined, but very useful in practice
 - Exercise EAFP to make the most of them
- Context managers aren't restricted to file-like-objects. We can use tools in the `contextlib` standard library module, such as the `closing()` wrapper to create our own context managers



Files and resource management Summary

- **help() can be used on instance objects, not just types**
- **Python supports bitwise operators &, | and left- and right-shifts**