Functions and Methods

Note that all file methods should be performed on **file objects**, as defined using open().

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Example** |
| open() | Open a file as a python object | my\_file = open("file.txt", "r") (See below for opening modes) |
| .close() | Close a file | my\_file.close() |
| .seek() | Place the file iterator at a certain line. *Needed* to loop over a file contents multiple times! | my\_file.seek(0) |
| .read() | Read entire contents of a file into a string | contents = my\_file.read() |
| .readlines() | Read file contents line-by-line. Commonly used in loops. | lines = my\_file.readlines() |
| .write() | Write to a file | my\_file.write("I'm being written to a file!") |
| with open() as f: | Opens a file, allows manipulation of contents, and closes file automatically once outside the statement | with open("myfile.txt", "r") as file\_handle:  # Perform operations  # Once the with statement is exited, the file is automatically closed |

csv Module functions and methods

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Example** |
| csv.reader() | Setup a csv reader on an **opened** file handle. Rows are parsed into **lists**. | my\_file = open("file.csv", "r")  reader = csv.reader(my\_file)  Note that an argument delimiter=... may be provided for other separators, like \t for tabs. |
| csv.DictReader() | Setup a csv reader on an opened file handle. Rows are parsed into **dictionaries** | my\_file = open("file.csv", "r")  reader = csv.DictReader(my\_file) |
| <file>.writerow() | Write a row to csv | reader.writerow([a,b,c,d]) |
| x=csv.writer(<file>)  x.writerows() | Write the contents of a list (or list of lists) to a csv file | my\_file = open("file.csv", "w")  x=csv.writer("output.csv")  x.writerows(somelist)  see parse\_hyphy.py for more examples |

File modes

|  |  |
| --- | --- |
| **Mode** | **Meaning** |
| r | Read-only |
| w | Write-only (CAUTION: will overwrite file contents!) |
| a | Append (Will *not* overwrite file contents, but append content to the bottom of the file) |
| r+ | Read and write (Mac and Linux) |
| rw | Read and write (PC) |

os Module

Allows you to access command line arguments from python

|  |  |  |  |
| --- | --- | --- | --- |
| **Command** | **Description** | **Unix equivalent** | **Example** |
| os.listdir | List all items in a given directory | ls | os.listdir("/directory/of/interest/") |
| os.remove | Remove a file | rm | os.remove("i\_hate\_this\_file.txt") |
| os.rmdir | Remove a directory | rm -r | os.rmdir("/i/hate/this/directory/") |
| os.mkdir | Create a new directory | mkdir | os.mkdir("/path/to/brand/new/directory/") |
| os.mkdirs | Create many new directories | mkdir | os.mkdir("/path/to/a/brand/new/directory/", "/path/to/another/brand/new/directory/") |
| os.chdir | Change directory where python is running | cd | os.chdir("/another/directory/where/i/want/to/be/") |

sys Module

Allows you to specify arguments for python functions in the command line

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Example** |
| print sys.argv | Prints command line input to file | print sys.argv[0] will print the name of the python script  print len(sys.argv) will print the number of arguments given to python, including the script |
| sys.argv[<number>] | Allows you to save command line input as a variable in python | infile = sys.argv[1]  argument1 = sys.argv[2]  argument2 = sys.argv[3] |

re Module

All functions shown here take two arguments: re.<function>(pattern, string)

|  |  |  |
| --- | --- | --- |
| **Function** | **Meaning** | **Example** |
| re.search | Find instances of a regular expression in a string | re.search(pattern, string) |
| re.split | Split string by occurrences of pattern (similar to .split(), but with regex! | re.split(pattern, string) |
| re.findall | Return all non-overlapping matches of pattern in string, as a list of strings | re.findall(pattern, string) |

Regular Expressions

|  |  |
| --- | --- |
| **Symbol** | **Meaning** |
| \s | space character |
| \t | tab character |
| \n | newline character (Mac and Linux! PCs may or may recognize this) |
| \r | newline character (PCs! Mac and Linux may or may recognize this) |
| . | wildcard |
| \d | digit (numbers only!) |
| \w | letter or number (case insensitive) |
| + | Symbol to append after a regular expression indicating “one or more of these” <br> E.g., \d+ means match 1 or more numbers |
| \* | Symbol to append after a regular expression indicating “zero or more of these” <br> E.g., \d\* means match 0 or more numbers |
| ^ | Beginning of line character |
| $ | End of line character |
| () | Use parentheses to capture matched patterns into variables |
| \ | Escape regular expression characters. E.g. \. means match an actual period |
| [] | Use brackets to define a custom regular expression <br> E.g. [A-Z] matches a capital letter only. [123] matches the number 1,2, or 3 only. |

(Note that many, **many** more exist!!)