**Файлы**

**.**– означает текущую директорию

**..** – означает на один уровень выше текущей директории

**/** - в начале строки означает(для windows???), что это путь относительно текущей директории выполняемого файла; если в начале строки такого символа нет, то путь воспринимается как абсолютный. Используется когда надо задать дальнейший путь, оттолкнувшись от текущего каталога.

C:\myfile.py – абсолютный путь

myfile.py – от текущей директории

/myfolder/myfile – от текущей директории

./myfolder/myfile – от текущей директории

../myfolder/myfile – от директории выше на 1 уровень оттекущей**unix**

../../myfolder/myfile – от директории выше на 2 уровняот текущей**unix**

Функциональный подход

from os import path

path.abspath(‘.’)

path – класс типа модуль

abspath – функция возвращает абсолютный путь текущего файла

ООПподход

from pathlib import Path

Path(‘C:/’) – новый объект класса Path с инициализированным путем

для **windows**

Path(‘.’) – новый объект класса Path с инициализированным путем к

текущей директории для **unix**

Path(‘.’).absolute()# абсолютный путь к **текущему каталогу**, где

#находится выполняемый файл

Или

Path.сwd()# абсолютный путь к выполняемому файлу (стат. метод)

Path – имя класса

**Класс Path**

Целевой каталог или файл указан в конце пути

/myfolder/**myfile**

Составление пути

**joinpath({строка с путем})** – создать новый объект с присоединением следующей части пути

StartFolder = Path(‘C:/’) – дляwindows

absPath = myFolder.joinpath(‘NextFolder’)

# ‘C:\NextFolder’#слеш заменяется для windows

**joinpath({строка с путем}) / ‘каталог2’ / ‘каталог N’** – используя магические методы, можно не вызывать метод для каждого раза

exists() - проверка присутствия директории или файла

Path(‘<путь>’).exists() – путь может быть абсолютным или относительным

Path(‘main.py’)

is\_file() – это файл?

is\_dir() –это директория?

iterdir() – возвращает список всех директорий и файлов по указанному пути в объекте Path

mkdir() – создать директорию, путь к которой указан, если его еще нет

rmdir() – удалить директорию, путь к которой указан

unlink() – удалить файл

**Чтение и запись в файл**

Встроенные функции python

open(‘<путь к файлу>) –открыть файл для чтения

open(‘<путь к файлу>’, ‘w’) – открыть файл для перезаписи

open(‘<путь к файлу>’, ‘a’) – открыть файл для до записи

Возвращает объект класса **TextIOWrapper**

Класс TextIOWrapper

write(‘текст’) – записать текст в файл

write(‘First string\n’) – ‘\n’ – переводкаретки

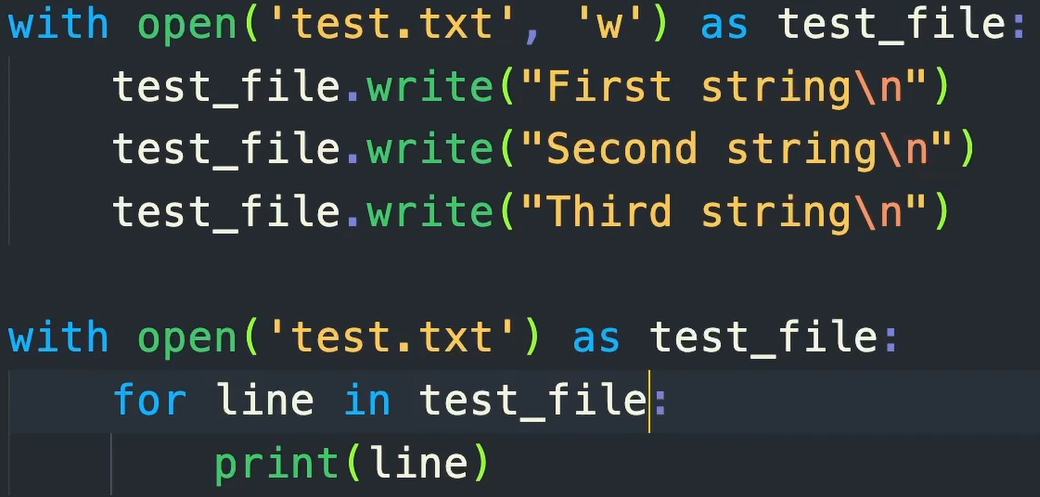
read() – считывает (загружает в себя) содержимое файла

close() – закрыть файл (необходимо использовать, в том числе,

когда нужно прочитать файл после **записи** или наоборот)

readline() – возвращает очередную строку по очереди

readlines() – возвращает список (list) строк файла

Это итерируемый объект сам по себе

Конструкция with позволяет не закрывать файл явно. После выхода из блока with у объекта test\_file вызовется метод \_\_exit\_\_. После этого удалится и сам объект. В методе \_\_exit\_\_ выполняется закрытие файла (отпуск дескриптора).