Федеральное государственное образовательное бюджетное   
учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

**ОТЧЕТ   
по практической (самостоятельной) работе**

**Лабораторная работа № 6:** Python

**Студента:** Ерминой С.Д.

**Дисциплина /Профессиональный модуль:** Основы алгоритмизации и программирования

|  |  |  |
| --- | --- | --- |
| **Группа: 2ИСИП-319** |  | **Преподаватель:** |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_/И.В. Сибирев/ |
|  |  | **Дата выполнения:** |
|  |  | 13.04.2021 г. |
|  |  | **Оценка за работу: \_\_\_\_\_\_\_\_\_\_\_\_** |

Москва   
2021

Содержание

[Лекция 3](#_Toc69238949)

[Получение текущего рабочего каталога в Python 3](#_Toc69238950)

[Изменение текущего рабочего каталога в Python 3](#_Toc69238951)

[Получение списка файлов в директории на Python 4](#_Toc69238952)

[Применение os.walk() 4](#_Toc69238953)

[Использование командной строки, через subprocess 4](#_Toc69238954)

[Комбинация os и fnmatch 5](#_Toc69238955)

[Использование os.listdir() и генераторов 6](#_Toc69238956)

[Использование pathlib 6](#_Toc69238957)

[Использование os.scandir() 7](#_Toc69238958)

[Пункты модификации: 10](#_Toc69238959)

[Измененный скрипт: 10](#_Toc69238960)

[Создание папки 10](#_Toc69238961)

[Проверить наличие файла 11](#_Toc69238962)

[Создаем новый файл в Python 14](#_Toc69238963)

[Чтение из файла 14](#_Toc69238964)

[Запись в файл 15](#_Toc69238965)

[Дозапись информации в файл 17](#_Toc69238966)

[Как запустить файл Python с помощью Командной строки Windows 18](#_Toc69238967)

[Найдите путь к файлу Python 18](#_Toc69238968)

[Практическое задание 19](#_Toc69238969)

[Список использованной литературы 19](#_Toc69238970)

Лекция

Получение текущего рабочего каталога в Python

Метод getcwd() модуля os в Python возвращает строку, содержащую абсолютный путь к текущему рабочему каталогу. Возвращенная строка не включает завершающий символ косой черты.

os.getcwd()

Чтобы использовать методы модуля os, вы должны импортировать модуль в верхней части файла.

Ниже приведен пример, показывающий, как распечатать текущий рабочий каталог:

*# Import the os module*

**import** **os**

*# Get the current working directory*

cwd = os.getcwd()

*# Print the current working directory*

**print**("Current working directory: {0}".format(cwd))

*# Print the type of the returned object*

**print**("os.getcwd() returns an object of type: {0}".format(type(cwd)))

Результат будет выглядеть примерно так:

Current working directory: /home/linuxize/Desktop

os.getcwd() returns an object of type: <class 'str'>

Если вы хотите найти каталог, в котором находится скрипт, используйте os.path.realpath(\_\_file\_\_) . Он вернет строку, содержащую абсолютный путь к запущенному скрипту.

Изменение текущего рабочего каталога в Python

Чтобы изменить текущий рабочий каталог в Python, используйте метод chdir() .

os.getcwd(path)

Метод принимает один аргумент — путь к каталогу, в который вы хотите перейти. Аргумент path может быть абсолютным или относительным.

Вот пример:

*# Import the os module*

**import** **os**

*# Print the current working directory*

**print**("Current working directory: {0}".format(os.getcwd()))

*# Change the current working directory*

os.chdir('/tmp')

*# Print the current working directory*

**print**("Current working directory: {0}".format(os.getcwd()))

Результат будет выглядеть примерно так:

Current working directory: /home/linuxize/Desktop

Current working directory: /tmp

Аргумент, передаваемый методу chdir() должен быть каталогом, в противном случае NotADirectoryError исключение NotADirectoryError . Если указанный каталог не существует, возникает исключение FileNotFoundError . Если у пользователя, от имени которого выполняется сценарий, нет необходимых разрешений, возникает исключение PermissionError .

*# Import the os module*

**import** **os**

path = '/var/www'

**try**:

os.chdir(path)

**print**("Current working directory: {0}".format(os.getcwd()))

**except** FileNotFoundError:

**print**("Directory: {0} does not exist".format(path))

**except** NotADirectoryError:

**print**("{0} is not a directory".format(path))

**except** PermissionError:

**print**("You do not have permissions to change to {0}".format(path))

Получение списка файлов в директории на Python

Применение os.walk()

Модуль os содержит длинный список методов, которые касаются работы с файловой системой и операционной системой. Один из них walk(), возвращающий имена файлов в дереве каталогов, двигаясь по дереву сверху вниз или снизу вверх (сверху вниз по умолчанию).

os.walk() возвращает список из трех элементов: имя корневого каталога, список имен вложенных папок и список файлов в текущем каталоге. Он одинаково хорошо работает с интерпретаторами Python 2 и 3.

import os

for root, dirs, files in os.walk("."):

for filename in files:

print(filename)

Использование командной строки, через subprocess

Модуль subprocess позволяет выполнить системную команду и собрать её результат. В нашем случае вызываемая системная команда выглядит следующим образом:

$ ls -p . | grep -v /$

Инструкция ls -p . распечатывает список файлов текущего каталога, добавляя разделитель / в конце имени каждого подкаталога, которые нам понадобится на следующем шаге. Вывод этого вызова передается команде grep, которая отфильтровывает данные по мере поступления. Параметры -v / $ исключают все имена записей, которые заканчиваются разделителем /. Фактически / $ -- регулярное выражение, которое соответствует всем строкам, содержащим символ / самым последним символом в строке, который определяется символом $.

Модуль subprocess позволяет строить настоящие конвейеры, а также соединять входные и выходные потоки, как это делается в командной строке. Вызов метода subprocess.Popen() открывает соответствующий процесс и определяет два параметра stdin и stdout.

Первая переменная ls определяет процесс выполнения ls –p для захвата stdout в конвейере. Поэтому поток stdout определяется как subprocess.PIPE. Вторая переменная grep также определяется как процесс, но вместо этого выполняет инструкцию grep –v /$.

Чтобы прочитать вывод команды ls из конвейера, поток stdin grep присваиваивается в ls.stdout. В заключение, переменная endOfPipe считывает вывод команды grep из grep.stdout, затем распечатывается в stdout циклом for.

import subprocess

# определение команды ls

ls = subprocess.Popen(["ls", "-p", "."],

stdout=subprocess.PIPE,

)

# определение команды grep

grep = subprocess.Popen(["grep", "-v", "/$"],

stdin=ls.stdout,

stdout=subprocess.PIPE,

)

# чтение из данных из потока stdout

endOfPipe = grep.stdout

# распечатка файлов в строку

for line in endOfPipe:

print(line)

Запуск файла

$ python find-files3.py

find-files2.py

find-files3.py

find-files4.py

...

Данное решение работает достаточно хорошо с Python 2 и 3, но его можно улучшить. Рассмотрим другие варианты.

Комбинация os и fnmatch

Решение, использующее подпроцессы, элегантно, но требует большого количества кода. Вместо этого, давайте объединим методы из двух модулей os и fnmatch. Этот вариант также работает с Python 2 и 3.

В качестве первого шага, импортируем модули os и fnmatch. Далее определим каталог, в котором нужно перечислить файлы, используя os.listdir(), а также шаблон для фильтрации файлов. В цикле for выполняется итерация списка записей, хранящихся в переменной listOfFiles.

В завершение, с помощью fnmatch отфильтровываются искомые записи и распечатываются соответствующие записи в stdout.

import os, fnmatch

listOfFiles = os.listdir('.')

pattern = "\*.py"

for entry in listOfFiles:

if fnmatch.fnmatch(entry, pattern):

print(entry)

Результат выполнения

$ python find-files.py

find-files.py

find-files2.py

find-files3.py

...

Использование os.listdir() и генераторов

Следующий вариант объединяет метод os.listdir() с функцией генератором. Код работает как с версиями 2, так и с 3 Python.

Как уже было сказано ранее, listdir() возвращает список записей для данного каталога. Метод os.path.isfile() возвращает True, если данная запись является файлом. Оператор yield завершает работу функции, но сохраняя текущее состояние и возвращает только имя записи являющейся файлом.

import os

def files(path):

for file in os.listdir(path):

if os.path.isfile(os.path.join(path, file)):

yield file

for file in files("."):

print(file)

Использование pathlib

Модуль pathlib предназначен для парсинга, сборки, тестирования и иной работы с именами файлов и их путями, используя объектно-ориентированный API вместо низкоуровневых строковых операций. Начиная с Python 3 модуль находится в стандартной библиотеке.

В следующем листинге определяется текущий каталог точкой («.»). Затем метод iterdir() возвращает итератор, который возвращает имена всех файлов. Далее циклом for распечатываются имена файлов друг за другом.

import pathlib

# определение пути

currentDirectory = pathlib.Path('.')

for currentFile in currentDirectory.iterdir():

print(currentFile)

В качестве альтернативы, можно отфильтровать файлы по именам с помощью метода glob. Таким образом, получаем требуемые файлы. Например, в приведенном ниже коде перечисляются Python файлы в выбранном каталоге, указав шаблон «\*.py» в glob.

import pathlib

# определение пути

currentDirectory = pathlib.Path('.')

# определение шаблона

currentPattern = "\*.py"

for currentFile in currentDirectory.glob(currentPattern):

print(currentFile)

Использование os.scandir()

В Python 3.6 добавлен новый метод scandir(), доступный из модуля os. Как понятно из названия он значительно упрощает получение списка файлов в каталоге.

Чтобы определить текущий рабочий каталог и сохранить его, инициализируем значение переменной path, для этого импортируем модуль os и вызовем функцию getcwd(). Далее, scandir() возвращает список записей для выбранного пути, которые проверяются на принадлежность файлу, используя метод is\_file().

import os

# определение текущей рабочей директории

path = os.getcwd()

# чтение записей

with os.scandir(path) as listOfEntries:

for entry in listOfEntries:

# печать всех записей, являющихся файлами

if entry.is\_file():

print(entry.name)

Получение списка файлов в каталоге и подкаталогах в Python

Давайте перечислим папки / файлы в текущем каталоге, используя **listdir**:

print(os.listdir())

# ['Data', 'Population\_Data', 'README.md', 'tutorial.py', 'tutorial\_v2.py']

Как видите, у меня есть 2 папки: Data и Population\_Data. У меня также есть 3 файла: файл разметки README.md и два файла Python, а именно, tutorial.py и tutorial\_v2.py.

Чтобы получить всю древовидную структуру папки моего проекта, давайте напишем функцию и затем используем **os.walk()**, чтобы перебрать все файлы в каждой папке текущего каталога.

def list\_files(startpath):

for root, dirs, files in os.walk(startpath):

if dir!= '.git':

level = root.replace(startpath, '').count(os.sep)

indent = ' ' \* 4 \* (level)

print('{}{}/'.format(indent, os.path.basename(root)))

subindent = ' ' \* 4 \* (level + 1)

for f in files:

print('{}{}'.format(subindent, f))

Вызовите эту функцию, используя текущий путь к рабочей директории, который мы видели как это сделать ранее:

startpath = os.getcwd()

list\_files(startpath)

Мы получим:

Project/

README.md

tutorial.py

tutorial\_v2.py

Data/

uscitiesv1.4.csv

Population\_Data/

Alabama/

Alabama\_population.csv

Alaska/

Alaska\_population.csv

Arizona/

Arizona\_population.csv

Arkansas/

Arkansas\_population.csv

California/

California\_population.csv

Colorado/

Colorado\_population.csv

Connecticut/

Connecticut\_population.csv

Delaware/

Delaware\_population.csv

...

Вывод был усечен для краткости.

Как видно из вывода, имена папок оканчиваются на /, а файлы внутри папок имеют отступ с четырьмя пробелами справа. В папке Data есть один CSV-файл с именем uscitiesv1.4.csv. Этот файл содержит данные о населении для каждого города в Соединенных Штатах. Папка Population\_Data содержит папки для состояний, содержащие отдельные csv-файлы для данных о населении для каждого состояния, извлеченные из uscitiesv1.4.csv.

* Вы хотите узнать, существует ли папка на Google Диске, используя имя папки.
* Вы хотите добиться этого, используя токен доступа и requests.get().

Если мое понимание верно, как насчет этой модификации? Пожалуйста, подумайте об этом как об одном из нескольких ответов.

Пункты модификации:

* Вы можете выполнить поиск в папке, используя запрос на фильтрацию файла drive.files.list.
  + В вашем случае запрос выглядит следующим образом. < UL> < Литий> name='filename' and mimeType='application/vnd.google-apps.folder'
* Если вы не хотите искать в корзине для мусора, добавьте к запросу and trashed=false.

 Чтобы подтвердить, существует ли папка, в этом случае проверяется свойство files. Это свойство представляет собой массив. Если папка существует, в массиве есть элементы.

Измененный скрипт:

import requests

import json

foldername = '#####' # Put folder name here.

access\_token = 'token'

url = 'https://www.googleapis.com/drive/v3/files'

headers = {'Authorization': 'Bearer ' + access\_token} # Modified

query = {'q': "name='" + foldername + "' and mimeType='application/vnd.google-apps.folder'"} # Added

response = requests.get(url, headers=headers, params=query) # Modified

obj = response.json() # Added

if obj['files']: # Added

print('Existing.') # Folder is existing.

else:

print('Not existing.') # Folder is not existing.

**Создание папки**

Для создания папки/каталога в любой операционной системе нужна следующая команда:

# создать пустой каталог (папку)

os.mkdir("folder")

После ее выполнения в текущем рабочем каталоге тут же появится новая папка с названием «folder».

Если запустить ее еще раз, будет [**вызвана ошибка**](https://pythonru.com/osnovy/obrabotka-iskljuchenij-python-blok-try-except-blok-finally) FileExistsError, потому что такая папка уже есть. Для решения проблемы нужно запускать команду только в том случае, если каталога с таким же именем нет. Этого можно добиться следующим образом:

# повторный запуск mkdir с тем же именем вызывает FileExistsError,

# вместо этого запустите:

if not os.path.isdir("folder"):

os.mkdir("folder")

Функция os.path.isdir() вернет True, если переданное имя ссылается на существующий каталог.

Проверить наличие файла

*Path.exists()*:

Метод Path.exists() проверяет существование пути path файловой системы. Другими словами, указывает ли путь к существующему файлу или каталогу:

>>> **from** **pathlib** **import** Path

>>> Path('.').exists()

*# True*

>>> Path('setup.py').exists()

*# True*

>>> Path('/etc').exists()

*# True*

>>> Path('nonexistentfile').exists()

*# False*

**Примечание**. Если путь path указывает на символическую ссылку, то [метод Path.exists()](https://docs-python.ru/standart-library/modul-pathlib-python/opredelit-tip-puti-fajlovoj-sistemy-fajl-katalog-ssylka/#Path.exists) возвращает информацию о том, указывает ли символическая ссылка на существующий файл или каталог.

*Path.is\_dir()*:

Метод Path.is\_dir() вернет True, если путь path указывает на каталог, или символическую ссылку, указывающую на каталог. Метод вернет False если путь указывает на файл другого типа.

Значение False также возвращается, если путь не существует или является неработающей символической ссылкой. Метод может вернуть False при отсутствии доступа к директории или символической ссылке, указывающей на директорию.

>>> **from** **pathlib** **import** Path

>>> Path('/some/to/directory').is\_dir()

*# True*

>>> Path('/some/to/file.ext').is\_dir()

*# False*

>>> Path('/some/to/symlink').is\_dir()

*# False*

*Path.is\_file()*:

Метод Path.is\_file() вернет True, если путь path указывает на файл или символическую ссылку, указывающую на обычный файл. Метод вернет False, если путь указывает на файл другого типа.

Значение False также возвращается, если путь не существует или является неработающей символической ссылкой. Метод может вернуть False при отсутствии доступа к файлу или символической ссылке, указывающей на файл.

>>> **from** **pathlib** **import** Path

>>> Path('/some/to/directory').is\_file()

*# False*

>>> Path('/some/to/file.ext').is\_file()

*# True*

>>> Path('/some/to/symlink').is\_file()

*# False*

*Path.is\_mount()*:

Метод Path.is\_mount() вернет True, если путь path является точкой монтирования в файловой системе, где смонтирована другая файловая система. В POSIX функция проверяет, находится ли родительский путь path/.. на устройстве, отличном от path или path/.. и path указывают на однy и ту же ноду i-node на одном устройстве - это должно определять точки монтирования для всех вариантов Unix и POSIX.

Не реализовано в Windows.

>>> **from** **pathlib** **import** Path

>>> Path('/some/to/directory').is\_mount()

*# False*

>>> Path('/some/to/mount').is\_mount()

*# True*

>>> Path('/some/to/file.ext').is\_mount()

*# False*

*Path.is\_symlink()*:

Метод Path.is\_symlink() вернет True, если путь path указывает на символическую ссылку, иначе вернет False.

Значение False также возвращается, если путь path не существует. Метод может вернуть False при отсутствии доступа к символической ссылке.

>>> **from** **pathlib** **import** Path

>>> Path('/some/to/directory').is\_symlink()

*# False*

>>> Path('/some/to/file.ext').is\_symlink()

*# False*

>>> Path('/some/to/symlink').is\_symlink()

*# True*

*Path.is\_socket()*:

Метод Path.is\_socket() вернет True, если путь path указывает на сокет Unix или символическую ссылку, указывающую на сокет Unix. Метод вернет False, если путь указывает на файл другого типа.

Значение False также возвращается, если путь не существует или является неработающей символической ссылкой. Метод может вернуть False при отсутствии доступа к сокету Unix или символической ссылке, указывающей на сокет.

*Path.is\_fifo()*:

Метод Path.is\_fifo() вернет True, если путь path указывает на FIFO или символическую ссылку, указывающую на FIFO. Метод вернет False, если путь указывает на файл другого типа.

Значение False также возвращается, если путь не существует или является неработающей символической ссылкой. Метод может вернуть False при отсутствии доступа к FIFO или символической ссылке, указывающей на FIFO.

*Path.is\_block\_device()*:

Метод Path.is\_block\_device() вернет True, если путь path указывает на блочное устройство или символическую ссылку, указывающую на блочное устройство. Метод вернет False, если путь указывает на файл другого типа.

Значение False также возвращается, если путь не существует или является неработающей символической ссылкой. Метод может вернуть False при отсутствии доступа к блочному устройству или символической ссылке, указывающей на блочное устройство.

*Path.is\_char\_device()*:

Метод Path.is\_char\_device() вернет True, если путь path указывает на символьное устройство или символическую ссылку, указывающую на символьное устройство. Метод вернет False, если путь указывает на файл другого типа.

Значение False также возвращается, если путь не существует или является неработающей символической ссылкой. Метод может вернуть False при отсутствии доступа к символьному устройству или символической ссылке, указывающей на символьное устройство.

Создаем новый файл в Python

# -\*- coding: utf-8 -\*-

my\_file = open('snake.txt', 'w')

# Закрываем файл

my\_file.close()

После выполнения данного скрипта, по указанному нами пути будет создан новый файл snake.txt если файл не был создан то нужно проверить права доступа на папку. Стоит учесть, что если файл snake.txt существовал ранее, то после открытия его функцией open() и мод W, тогда содержимое файла удаляется. Если вы хотите добавить что-то в конце, то следует использовать мод 'a+'.

**Чтение из файла**

Открыли мы файл, а теперь мы хотим прочитать из него информацию. Для этого есть несколько способов, но большого интереса заслуживают лишь два из них.

Первый - метод read, читающий весь файл целиком, если был вызван без аргументов, и n символов, если был вызван с аргументом (целым числом n).

>>>

**>>>** f = open('text.txt')

**>>>** f.read(1)

'H'

**>>>** f.read()

'ello world!\nThe end.\n\n'

Ещё один способ сделать это - прочитать файл построчно, воспользовавшись [циклом for](https://pythonworld.ru/osnovy/cikly-for-i-while-operatory-break-i-continue-volshebnoe-slovo-else.html):

>>>

**>>>** f = open('text.txt')

**>>> for** line **in** f:

**...**  line

**...**

'Hello world!\n'

'\n'

'The end.\n'

'\n'

**Запись в файл**

Теперь рассмотрим запись в файл. Попробуем записать в файл вот такой вот список:

>>>

**>>>** l = [str(i)+str(i-1) **for** i **in** range(20)]

**>>>** l

['0-1', '10', '21', '32', '43', '54', '65', '76', '87', '98', '109', '1110', '1211', '1312', '1413', '1514', '1615', '1716', '1817', '1918']

Откроем файл на запись:

>>>

**>>>** f = open('text.txt', 'w')

Запись в файл осуществляется с помощью метода write:

>>>

**>>> for** index **in** l:

**...**  f.write(index + '**\n**')

**...**

4

3

3

3

3

Для тех, кто не понял, что это за цифры, поясню: метод write возвращает число записанных символов.

После окончания работы с файлом его **обязательно нужно закрыть** с помощью метода close:

>>>

**>>>** f.close()

Теперь попробуем воссоздать этот список из получившегося файла. Откроем файл на чтение (надеюсь, вы поняли, как это сделать?), и прочитаем строки.

>>>

**>>>** f = open('text.txt', 'r')

**>>>** l = [line.strip() **for** line **in** f]

**>>>** l

['0-1', '10', '21', '32', '43', '54', '65', '76', '87', '98', '109', '1110', '1211', '1312', '1413', '1514', '1615', '1716', '1817', '1918']

**>>>** f.close()

Мы получили тот же список, что и был. В более сложных случаях (словарях, вложенных кортежей и т. д.) алгоритм записи придумать сложнее. Но это и не нужно. В python уже давно придумали средства, такие как [pickle](https://pythonworld.ru/moduli/modul-pickle.html" \t "_blank) или [json](https://pythonworld.ru/moduli/modul-json.html" \t "_blank), позволяющие сохранять в файле сложные структуры.

**Дозапись информации в файл**

Для дозаписи информации в файл, то есть, записи с сохранением предыдущего содержимого, файл следует открыть в режиме ‘a’:

file = open("out.txt", "a")

Тогда, выполняя эту программу, мы в файле увидим уже шесть строчек. И смотрите, в зависимости от режима доступа к файлу, мы должны использовать или методы для записи, или методы для чтения. Например, если вот здесь попытаться прочитать информацию с помощью метода read:

file.read()

то возникнет ошибка доступа. Если же мы хотим и записывать и считывать информацию, то можно воспользоваться режимом a+:

file = open("out.txt", "a+")

Так как здесь файловый указатель стоит на последней позиции, то для считывания информации, поставим его в самое начало:

   file.seek(0)

**print**( file.read() )

А вот запись данных всегда осуществляется в конец файла.

Следующий полезный метод для записи информации – это writelines:

file.writelines(["Hello1**\n**", "Hello2**\n**"])

Он записывает несколько строк, указанных в коллекции. Иногда это бывает удобно, если в процессе обработки текста мы имеем список и его требуется целиком поместить в файл.

[**Как запустить файл Python с помощью Командной строки Windows**](https://ru.wikihow.com/%D0%B7%D0%B0%D0%BF%D1%83%D1%81%D1%82%D0%B8%D1%82%D1%8C-%D1%84%D0%B0%D0%B9%D0%BB-Python-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9-%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8-Windows)

**Найдите путь к файлу Python**

**1**

**Перейдите к папке с файлом Python.** Найдите файл Python, который хотите открыть в Командной строке.

* + Если вам уже известен путь к файлу Python, который вы хотите открыть, перейдите к разделу об [открытии файла в Командной строке.](https://ru.wikihow.com/%D0%B7%D0%B0%D0%BF%D1%83%D1%81%D1%82%D0%B8%D1%82%D1%8C-%D1%84%D0%B0%D0%B9%D0%BB-Python-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9-%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8-Windows#step_2_1).

**2**

**Выберите файл Python.** Нажмите один раз на файл Python, путь к которому вы хотите узнать.

**3**

**Щелкните правой кнопкой мыши по файлу Python.** Появится выпадающее меню.

**4**

**Выберите Свойства из выпадающего меню.** Откроется окно свойств.

**5**

**Обратите внимание на значение в строке «Расположение».** Адрес папки (или «путь») справа от пункта «Расположение» — это именно то, что вам нужно ввести в Командную строку, чтобы перейти к каталогу, в котором хранится файл Python.

* + Чтобы скопировать расположение, его необходимо выделить (зажмите и перетащите указатель мыши по значению в строке «Расположение»), а затем нажать Ctrl+C.

Практическое задание

Разобраться с HTTP. Написать программу сервер,

что автоматически запускает свой сайт (HTML сайт визитка) в браузере через cmd.

***В ЭЛЖУРЕ ПРИКРЕПИЛА***

Список использованной литературы

<https://www.severcart.ru/blog/all/list_direcory_content_with_python/>

<http://python-3.ru/page/python-wite-writelines>

<https://pythonworld.ru/tipy-dannyx-v-python/fajly-rabota-s-fajlami.html>

<https://ru.wikihow.com/%D0%B7%D0%B0%D0%BF%D1%83%D1%81%D1%82%D0%B8%D1%82%D1%8C-%D1%84%D0%B0%D0%B9%D0%BB-Python-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9-%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8-Windows#/%D0%A4%D0%B0%D0%B9%D0%BB:Use-Windows-Command-Prompt-to-Run-a-Python-File-Step-5.jpg>