# Урок 18.01. Модуль os. Генерация путей и метод listdir

Если посмотреть на тему нашего сегодняшнего занятия, то можно догадаться, что мы сегодня начнем выходить за рамки одного нашего одного скрипта и начнем работать с операционной системой (ОС) и её файлами. Нам нужно узнать много интересного, так что сразу перейдем к делу.

Для начала вкратце вспомним как формируются пути в файловой системе и рассмотрим, что такое абсолютный и относительный путь.

Относительный путь к файлу – это такой путь к файлу, который определяется относительно текущей папки, документа, программы и т.д. Например, «Lessons\Module 01. Введение\Урок 01.01. Первая программа. Переменные».

Абсолютный путь – полный путь к файлу, начиная с корня диска. Например, «C:\IvanovAA\Lessons\Module 01. Введение\Урок 01.01. Первая программа. Переменные».

И, как всегда, рассмотрим небольшую задачу «Путь к файлу».

Нам нужно сформировать относительный путь к файлу, т.к. этот путь нужно будет использовать в других скриптах проекта. Сам путь состоит из последовательности двух папок и имени файла, причем одна из папок и имя файла вводятся пользователем с клавиатуры. И нам нужно вывести на экран этот относительный путь. Программа будет выглядеть следующим образом:

folder = 'project'

file\_name = 'resources.txt'

path = f'doc/{folder}/{file\_name}'

print(path)

Здесь, конечно, мы немного забыли (забили) на ввод данных с клавиатуры, чтобы не вводить сейчас одно и то же несколько раз. Дальше с помощью f-string мы сформировали строку и сохранили её в переменной. Но здесь есть одна не большая проблема, и состоит она в универсальности. Возможно, вы не знали, но в разных ОС разделители папок и фала в путях производится разными слешами (косые черты – «/» и «\»). Сейчас в нашей программе использован слеш для Windows, и если запускать такой скрипт на linux и попытаться обратиться по этому пути, то мы получим ошибку. А наша программа должна быть универсальной, и уметь запускаться на любой системе. А значит мы должны избавиться от вручную прописанных разделителей, и, как вы уже могли догадаться, за нас уже всё придумали. Для решения нам потребуется использовать модуль os, который отвечает за работу с ОС пользователя. Импортируем его в нашу программу «import os»и сформируем относительный путь правильно.

import os

folder = 'project'

file\_name = 'resources.txt'

real\_path = os.path.join('docs', folder, file\_name)

print(real\_path)

Запустим и получим правильный относительный путь к нашему файлу. Если мы хотим получить абсолютный путь к нашему файлу, то это можно получить с помощью команды «abspath»:

import os

folder = 'project'

file\_name = 'resources.txt'

real\_path = os.path.join('docs', folder, file\_name)

print(real\_path)

abs\_path = os.path.abspath(file\_name)

print(abs\_path)

И вот наш абсолютный путь.

С модулем os есть еще пару интересных фокусов. И для начала узнаем как можно перемещаться между папками в относительных путях: чтобы обратиться на папку «выше» (выйти из текущей папки на уровень выше) в путях нужно прописать двоеточие «..». Например, «..\folder\resources.txt». Но тут мы использовали слеш, что не очень правильно, и если мы хотим получить абсолютный путь то нужно написать вот так:

os.path.abspath(os.path.join('..', 'folder', 'resources.txt'))

Если нам нужно обратиться в корневой каталог (на диске C или D), то можно использовать такую конструкцию «\folder\resources.txt», и опять мы использовали слеши, однако перед первым слешем ничего не стоит, в таком случае к корню диска можно обратиться следующим образом:

os.path.abspath(os.path.join(os.path.sep, 'folder', 'resources.txt'))

Здесь мы использовали такой метод, как os.path.sep, этот метод и обозначает разделитель в пути именах файловой системы.

Для закрепления давайте рассмотрим еще одну задачу – «Проекты».

Как мы знаем, для каждого проекта всегда есть своя папка, и, конечно же, проектов бывает несколько. Представим, что у нас их пока что два. И наша программа должна вывести на экран список файлов каждого проекта. И тут нам понадобится знание о такой функции библиотеки os как listdir (выводит список всех объектов, находящихся в директории). И вот так будет выглядеть наша программа:

import os

def print\_dirs(project):

print(f'\nСодержимое директории: {project}')

for i\_elem in os.listdir(project):

path = os.path.join(project, i\_elem)

print(f' {path}')

projects\_list = ['Lessons', 'My\_project']

for i\_project in projects\_list:

path\_to\_project = os.path.realpath(os.path.join('..', '..', '..', i\_project))

print\_dirs(path\_to\_project)

Для корректной работы, надо обращать внимание из какой папки мы запускаем нашу программу.

Мы уже с вами познакомились, как генерировать пути к файлам и папкам, однако, нам еще нужно проверить, а существует ли такой путь вообще. Для этого возьмем нашу последнюю программу с проектами, допишем в список проектов не существующую папку, например, вот так:

import os

def print\_dirs(project):

print(f'\nСодержимое директории: {project}')

for i\_elem in os.listdir(project):

path = os.path.join(project, i\_elem)

print(f' {path}')

projects\_list = ['new\_project', 'Lessons', 'My\_project']

for i\_project in projects\_list:

path\_to\_project = os.path.realpath(os.path.join('..', '..', '..', i\_project))

print\_dirs(path\_to\_project)

И теперь если запустить такую программу, у нас будет выведена ошибка, что системе не удается найти указанный путь. А значит, нам нужно проверять существует ли папка, чтобы наша программа всё-таки работала, а не «падала» при малейшем отклонении. Для этого в функции которая занимается выводом списка папки нужно прежде этого проверить а существует ли вообще такой. Как мы уже знаем, всё что связанно с путями находится в os.path, так давайте посмотрим какие методы существуют в этой библиотеке. И если вы пролистали все методы и не смогли найти подходящий, то я вам подскажу это метод exists. А программа будет выглядеть вот так:

import os

def print\_dirs(project):

print(f'\nСодержимое директории: {project}')

if not os.path.exists(project):

print('\tНет такого проекта')

return

for i\_elem in os.listdir(project):

path = os.path.join(project, i\_elem)

print(f' {path}')

projects\_list = ['new\_project', 'Lessons', 'My\_project']

for i\_project in projects\_list:

path\_to\_project = os.path.realpath(os.path.join('..', '..', '..', i\_project))

print\_dirs(path\_to\_project)

Запускаем и всё работает.

Давайте рассмотрим ещё одну стандартную задачу.

Задача «Поиск файла».

Нам нужно написать программу, которая ищет файл, заданный по имени.

Очень короткая задача. Для её решения нам понадобится использование рекурсивных функций, с помощью которых мы будет заглядывать внутрь папок и уже там искать данный файл. Вот так будет выглядеть решение:

import os

def search\_file(dir\_name, file\_name):

if file\_name in os.listdir(dir\_name):

return os.path.join(dir\_name, file\_name)

for cur\_dir in os.listdir(dir\_name):

cur\_dir = os.path.realpath(os.path.join(dir\_name, cur\_dir))

if os.path.isdir(cur\_dir):

result\_search = search\_file(cur\_dir, file\_name)

if result\_search:

return result\_search

return None

path = os.path.realpath(os.path.join('..', '..', '..'))

print(search\_file(path, 'search.py'))

Вот такая программа у нас получилась.