Ярошевич В.А.

090302-ИСТа-о22

ФИТУ 3 курс

**Лабораторная работа №4**

**РАБОТА С ОСНОВНЫМИ МОДУЛЯМИ**

**Цель работы:** изучить основные модули стандартной библиотеки Python 3, рассмотреть модули *os* и *datetime*.

**Задание №6:** написать функцию, которая принимает два объекта *datetime* и возвращает число по модулю дней между ними.

**Краткая теория**

Модули в *Python* – это файлы с расширением .*py*, которые содержат исполняемый код на Python. Именем модуля является имя файла без расширения. Система модулей позволяет логически организовывать код программы. Группирование кода в модули значительно облегчает процесс написания, а что более важно – дальнейшее понимание написанной программы. Для того, чтобы использовать файл .*py* как модуль в другом файле, требуется использовать ключевое слово **import.**

Стандартная библиотека Python содержит достаточно большое количество встроенных модулей. Рассмотрим некоторые из них:

− модуль **os**. Представляет множество функций для работы с операционной системой. При этом стоит учитывать, что их поведение, как правило, не зависит от конкретной операционной системы, что позволяет разрабатывать кроссплатформенные решения с использованием данного модуля;

− модуль **datetime**. Содержит классы для работы с датой и временем. В отличие от других языков программирования данный модуль достаточно прост в освоении и использовании;

− модуль **array**. Представляет собой реализацию массивов на Python. Они похожи на списки, но их различает то, что массивы ограничены одним определенным типом данных и размером каждого элемента;

− модуль **itertools**. Содержит сборник полезных итераторов;

− модуль **sys**. Модуль, позволяющий обращаться к некоторым переменным и функциям, которые взаимодействуют с интерпретатором Python;

− модуль **random**. Данный модуль, как следует из названия, предназначен для генерации случайной последовательности чисел, букв и случайного выбора элементов последовательности;

− модуль **math**. Предоставляет набор функций для выполнения математических операций;

− модуль **json**. Содержит функции для работы с форматом передачи данных *json*. Предоставляет возможности как парсинга данных, так и создания объектов в данном формате;

− модуль **gzip** и **zlib**. Предоставляет средства для работы со сжатыми файлами;

− модуль **tkinter**. Данный модуль позволяет создавать кроссплатформенные программы с графическим интерфейсом.

Данные модули всего лишь небольшая часть стандартной библиотеки Python. Существуют модули для работы со звуком, базами данных, шифрованием, хешированием, логгированием и многим другим. Остановимся более подробно на некоторых модулях, которые понадобятся в дальнейших работах.

**Модуль os**

Как было сказано ранее, данный модуль предоставляет функции для работы с операционной системой:

− **os.listdir(path)** – функция позволяет получить список файлов и директорий по указанному пути аргумента параметра *path*;

− **os.mkdir(path)** – функция позволяет создать новую директорию. В случае, если такая существует, пробрасывает *OSError*;

− **os.remove(path)** – функция удаляет файл;

− **os.rename(src, dst)** – функция переименовывает файл или директорию из *src* в *dst*;

− **os.rmdir(path)** – удаляет пустую директорию. Стоит выделить также вложенный модуль *os.path*, который предоставляет средства для работы с путями:

− **os.path.exists(path)** – функция возвращает *True*, в случае если *path* указывает на существующий путь или дескриптор открытого файла;

− **os.path.isfile(path)** – функция возвращает *True*, если *path* указывает на файл; 48

− **os.path.isdir(path)** – функция возвращает *True*, если *path* указывает на директорию;

− **os.path.join(path1[, path2[, ...]])** – функция соединяет пути с учетом особенностей операционной системы;

− **os.path.samefile(path1, path2)** – функция возвращает *True*, если *path1* и *path2* указывают на один и тот же файл или директорию.

**Модуль datetime**

− **datetime.date(year, month, day)** – класс *date* создает объект даты по значениям аргументов указанных параметров. В случае, если один из аргументов будет некорректным, сгенерируется исключение *ValueError*;

− **datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None)** – класс *time* создает объект времени по значениям аргументов указанных параметров;

− **datetime.today()** – возвращает объект *datetime* по текущей дате и времени;

− **datetime.strptime(date\_string, format)** – преобразует строку в *datetime*;

− **datetime.strftime(datetime\_obj, format)** – формирует строку из объекта *datetime* в указанном формате.

**Ход работы**

Ознакомившись с материалом, можно приступать к выполнению задания. Данную лабораторную следует разбить на следующие пункты:

1) Импортировать модуль

2) Написать функцию *days\_difference*

3) Абсолютное значение

1. Мы импортируем класс *datetime* из модуля *datetime*.

2. В этой функции мы вычисляем разницу между двумя датами, используя вычитание. Это дает объект *timedelta*, из которого мы можем получить количество дней с помощью атрибута days*.*

3. Мы используем функцию *abs(),* чтобы гарантировать, что возвращаемое значение всегда будет положительным, вне зависимости от порядка переданных дат.

Листинг кода:

from datetime import datetime

def days\_difference(date1, date2):

difference = abs((date2 - date1).days)

return difference

date1 = datetime(2024, 9, 25)

date2 = datetime(2024, 10, 5)

result = days\_difference(date1, date2)

print(result)

**Результаты выполнения кода:**

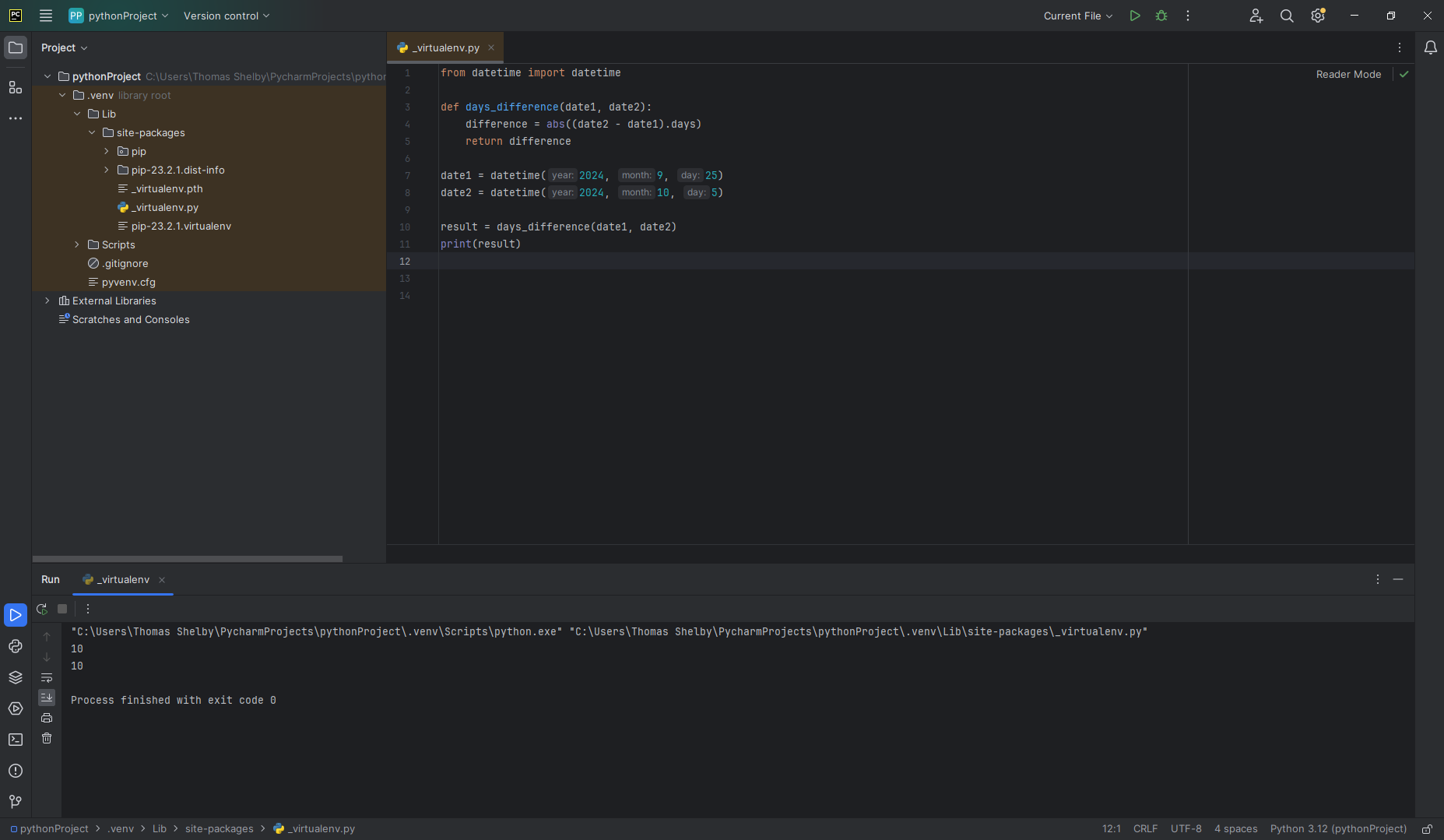


Рис. 1 - результаты работы кода

**Вывод**: в результате работы мы изучили основные модули стандартной библиотеки Python 3, а также рассмотрели модули *os* и *datetime*.