ШКОЛА ОБЛАЧНЫХ ТЕХНОЛОГИЙ

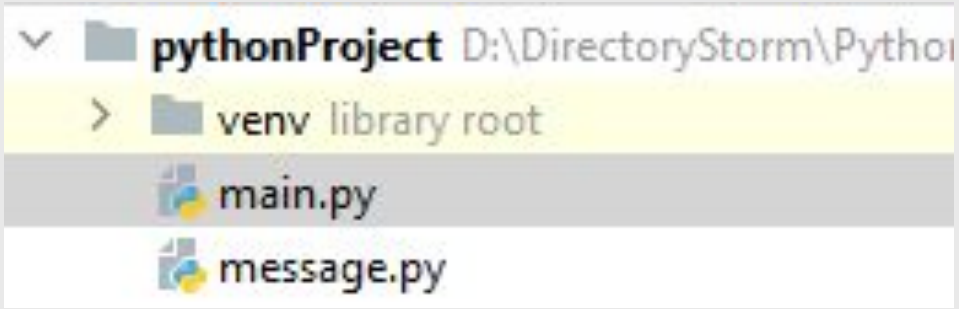
Модули. Import модулей. Разделение кода в Python.

Модуль

Модуль в языке Python представляет отдельный файл с кодом, который можно повторно использовать в других программах. Для создания модуля необходимо создать собственно файл с расширением *. py, который будет представлять модуль. Название файла будет представлять название модуля.

Допустим, основной файл программы называется main.py. И мы хотим подключить к нему внешние модули.

Для этого сначала определим новый модуль: создадим в той же папке, где находится main.py, новый файл, который назовем message.py.



Соответственно модуль будет называться message. Определим в нем следующий код:

```
hello = "Hello all"


def print_message(text):
    print(f"Message: {text}")
```

Import

Оператор import в Python применяется для того, чтобы сделать код в одном модуле доступным для работы в другом. Импорт в Python важен для эффективного структурирования кода. Правильное применение импорта повысит вашу продуктивность: вы сможете повторно использовать код и при этом продолжать осуществлять поддержку своих проектов.

```
import message

print(message.hello)

message.print_message("Hello work")
```

Подключение функциональности модуля в глобальное пространство имен

Другой вариант настройки предполагает импорт функциональности модуля в глобальное пространство имен текущего модуля с помощью ключевого слова from:

```
from message import print_message

# обращаемся к функии print_message из модуля message
print_message("Hello work")  # Message: Hello work

# переменная hello из модуля message не доступна, так как она не импортирована
# print(message.hello)
# print(hello)
```

Если необходимо импортировать в глобальное пространство имен весь функционал, то вместо названий отдельных функций и переменных можно использовать символ зводочки *:

```
from message import *

# обращаемся к функции print_message из модуля message
print_message("Hello work")  # Message: Hello work

# обращаемся к переменной hello из модуля message
print(hello)  # Hello all
```

Стоит отметить, что импорт в глобальное пространство имен чреват коллизиями имен функций. Например, если у нас том же файле определена функция с тем же именем до ее вызова, то будет вызываться функция, которая определена последней:

```
from message import *

print_message("Hello work")  # Message: Hello work - применяется функция из модуля message


def print_message(some_text):
    print(f"Text: {some_text}")


print_message("Hello work")  # Text: Hello work - применяется функция из текущего файла
```

Установка псевдонимов

При импорте модуля и его функциональности мы можем установить для них псевдонимы. Для этого применяется ключевое слово as, после которого указывается псевдоним. Например, установим псевдоним для модуля:

```
import message as mes  # модуль message проецируется на псевдоним mes

# выводим значение переменной hello
print(mes.hello)  # Hello all
# обращаемся к функии print_message
mes.print_message("Hello work")  # Message: Hello work
```

В данном случае пространство имен будет называться mes, и через этот псевдоним можно обращаться к функциональности модуля.

Подобным образом можно установить псевдонимы для отдельной функциональности модуля:

```
from message import print_message as display
from message import hello as welcome

print(welcome)
display("Hello work")
```

Задания для выполнения

1) Написать калькулятор при этом все математические действия проводить в другом файле calc.py.

2) Написать программу которая будет возводить число от пользователя в квадрат. Все действия должны производиться в другом файле. (Дополнительно: Реализовать проверку на то ввёл ли пользователь число или другой символ)

3) С помощью модуля Random сгенерировать массив чисел с случайным значением при этом длинна массива должна быть от 7 до 14 элементов.

Домашняя работа

- Повторить все предыдущие лекции.