

Практическое занятие № 16

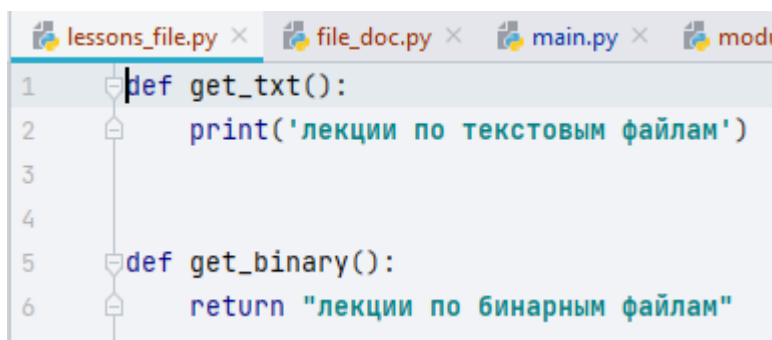
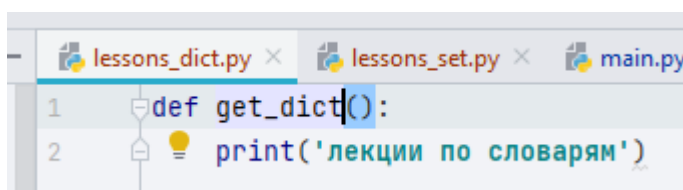
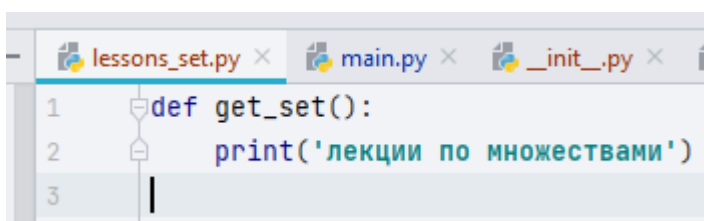
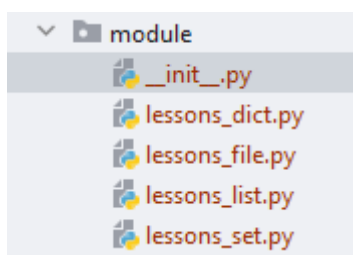
1. Наименование практического занятия: составление программ с использованием пакетов в IDE PyCharm Community.
2. Количество часов: 4
3. Место проведения: главный корпус РКСИ, ауд. 420.
4. Цели практического занятия: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, **приобрести навыки** составление программ с использованием пакетов в IDE PyCharm Community.

Инструкция к практическому заданию № 16

1. Выполнить все указанные действия, в конспекте отразить основные теоретические вопросы.

Пакетом называется папка с модулями, как правило, решающими сходные задачи, в которой расположен файл инициализации `__init__.py`. Файл инициализации может быть пустым или содержать код, который будет выполнен при первой операции импортирования любого модуля, входящего в состав пакета. В любом случае он обязательно должен присутствовать внутри папки с модулями.

В рамках текущего проекта создадим пакет `module`



!!! При написании скриптов следить за кодировкой (UTF-8) !!!

В файле `main.py` выполним импортирование пакета `module`. Пакеты импортируются также, как и модули.

```
main.py x lessons_file.py x
1 import module
2
3 print(dir(module))
```

Dir() покажет что импортировалось в пространство имен.

```
main x
C:\Users\OLGA\AppData\Local\Programs\Python\Python38-32\python.exe C:/Python
['NAME', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__',
Process finished with exit code 0
```

Видим переменную NAME и служебные переменные, т.е. при импорте пакета автоматически выполнится `__init__.py` и все что в нем определено импортируется при импорте всего пакета. Но пока без других скриптов пакета. Чтобы импортировать скрипт из модуля необходимо

```
_init_.py x main.py x lessons_file.py x
1 import module.lessons_set
2
3 NAME = "курс лекций по Python"
```

Конструкция `module.lessons_set` указывает из какого пакета (module) импортируется нужный модуль (lessons_set).

После запуска `main.py` появилось пространство имен `lessons_set`, в котором содержатся все объекты скрипта `lessons_set`, в частности функция `get_set()`.

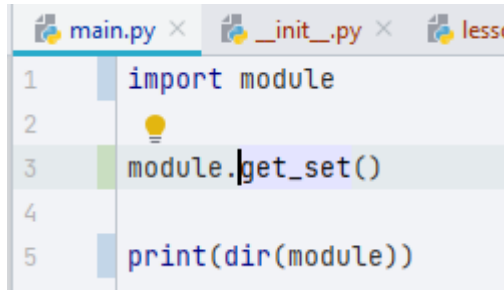
```
main.py
['__package__', '__path__', '__spec__', 'lessons_set', 'module']
```

Что бы обратиться к функции `get_set()` необходимо

```
_init_.py x main.py x lessons_file.py x
1 from .lessons_set import *
2
3 NAME = "курс лекций по Python"
```

Конструкция `.lessons_set` является относительным импортом. Относительный импорт является предпочтительнее, чем абсолютный (`module.lessons_set`), поскольку имя пакета в будущем может меняться и придется изменять название пакета у каждого `import`.

Т.к. при использовании `import *` происходит автоматическое импортирование пространства имен модуля в пространство имен пакета, то в `main.py` теперь достаточно сразу записать `get_set()`, минуя `lessons_set`.



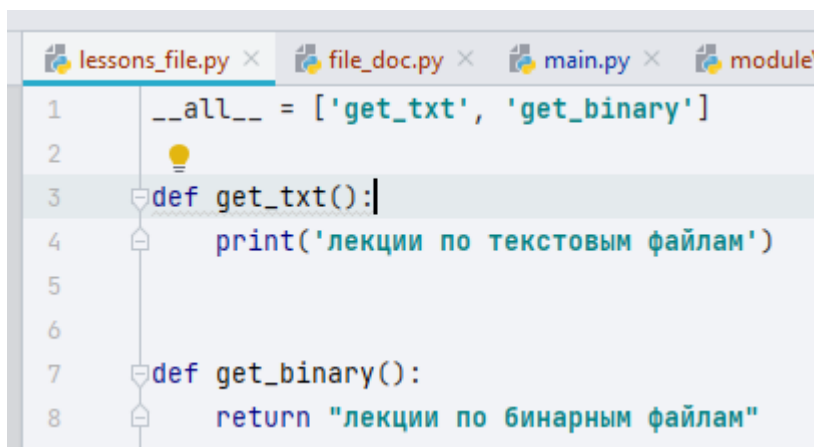
```
1 import module
2
3 module.get_set()
4
5 print(dir(module))
```

Кроме этого удобно использовать и такой вариант `import`, при котором все модули импортируются в программу, но имя пакета не указывается:

```
from . import lessons_set, lessons_dict, lessons_file, lessons_list
```

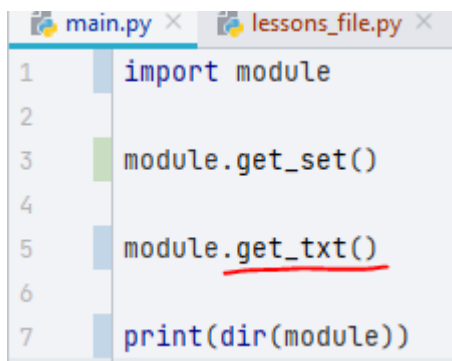
Символ «*» не рекомендуется использовать в основных программах (что бы избежать конфликта имен), но для модулей сделано исключение.

Что бы избежать конфликта имен при импортировании (через `import *`) можно контролировать импортируемые данные с помощью переменной `__all__`. Переменная `__all__` должна ссылаться на список, в котором указываются импортируемые объекты.



```
1 __all__ = ['get_txt', 'get_binary']
2
3 def get_txt():
4     print('лекции по текстовым файлам')
5
6
7 def get_binary():
8     return "лекции по бинарным файлам"
```

В `main.py` добавим и выполним

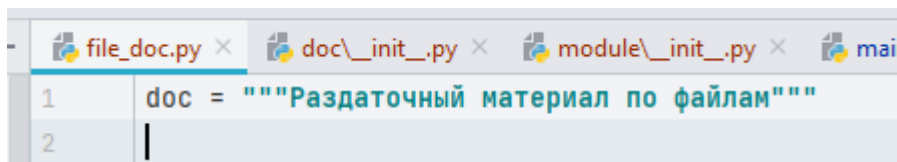
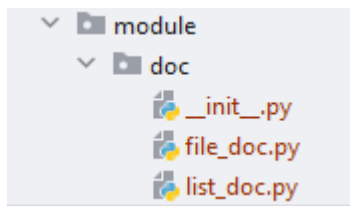


```
1 import module
2
3 module.get_set()
4
5 module.get_txt()
6
7 print(dir(module))
```

```
C:\Users\OLGA\AppData\Local\Pro
лекции по множествам
лекции по текстовым файлам
['NAME', '__builtins__', '__casl
```

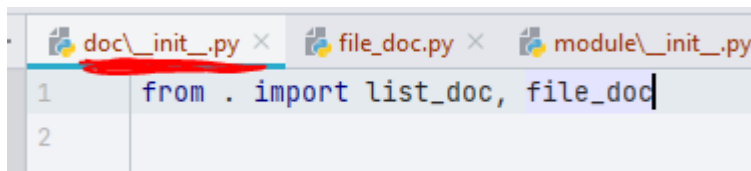
Вложенные пакеты

Внутри пакета `module` создадим пакет `doc`, в котором тоже будет свой файл инициализации. В пакете `doc` создадим еще два файла:



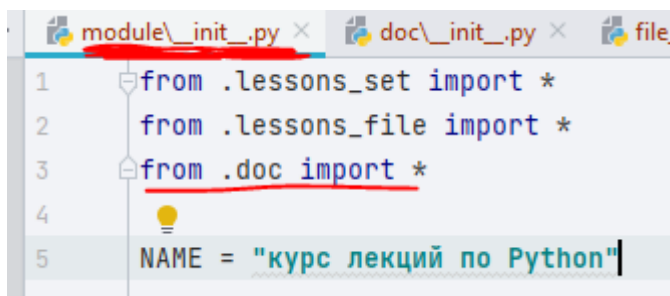
```
1 doc = """Раздаточный материал по файлам"""
2
```

В файле `init__.py` пакета `doc` выполним импорт двух модулей



```
1 from . import list_doc, file_doc
2
```

В файле `init__.py` пакета `module` импортируем пакет `doc`



```
1 from .lessons_set import *
2 from .lessons_file import *
3 from .doc import *
4
5 NAME = "курс лекций по Python"
```

А в файле `main.py` обратимся к строке «Раздаточный материал по спискам»

```
main.py x doc\_init_.py x module\_
1 import module
2
3 module.get_set()
4
5 module.get_txt()
6
7 print(module.list_doc.doc)
8
9 print(module.file_doc.doc)
10
11 print(module.get_binary())
12
13 print(dir(module))
14
```

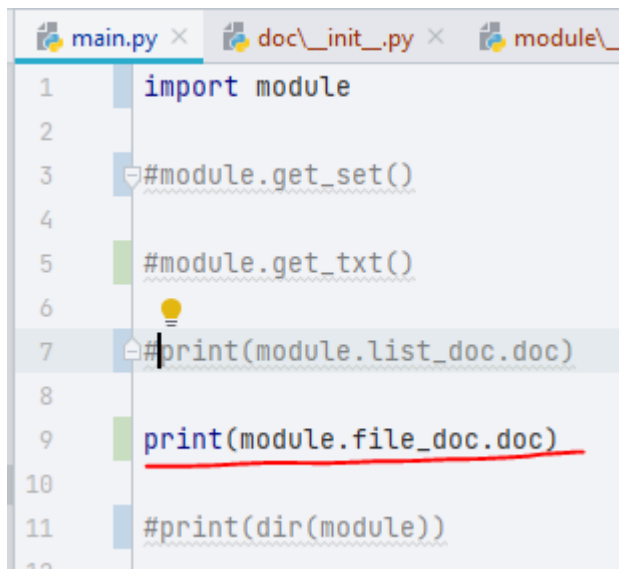
```
main x
C:\Users\OLGA\AppData\Local\Programs\Python\
лекции по множествами
лекции по текстовым файлам
Раздаточный материал по спискам
['NAME', '__builtins__', '__cached__', '__doc__']
```

Из модулей вложенных пакетов можно обращаться к модулям внешнего пакета

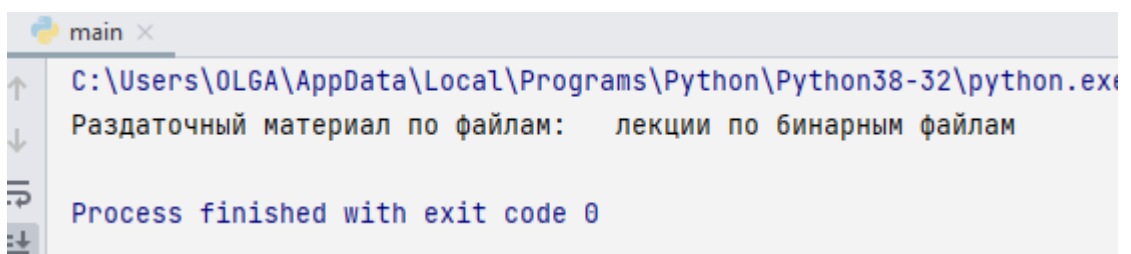
```
file_doc.py x main.py x module\_init_.py x doc\_init_.py x lessons_file.
1 from ..lessons_file import get_binary
2
3 doc = """Раздаточный материал по файлам: """ + get_binary()
4
```

Конструкция `..lessons_file` означает, что необходимо перейти на уровень выше, т.е. к пакету `module` и от туда взять файл `lessons_file`.

В `main.py` добавим и выполним



```
1 import module
2
3 #module.get_set()
4
5 #module.get_txt()
6
7 #print(module.list_doc.doc)
8
9 print(module.file_doc.doc)
10
11 #print(dir(module))
12
```



```
main x
C:\Users\OLGA\AppData\Local\Programs\Python\Python38-32\python.exe
Раздаточный материал по файлам: лекции по бинарным файлам
Process finished with exit code 0
```

Если необходимо обратиться уровнем еще выше, то указываются три точки. Чем выше уровень, тем больше точек необходимо указать.

2. Выполнить на оценку.

Создайте пакет 'figures', состоящий из трех подпакетов: 'triangle', 'circle', 'square'. В каждом подпакете будем иметь файл code.py, где создадим ряд функций:

- для пакета 'circle': функции circle_perimeter() – вычисляет длину окружности, circle_area() – вычисляет площадь окружности.

Еще заведем переменную default_radius = 5, которая будет скрыта при импорте модуля. Ее назначение – дефолтный радиус для окружности, если пользователь не введет свой.

Обе функции принимают на вход только радиус.

- для пакета 'triangle': функции triangle_perimeter() – вычисляет периметр треугольника, triangle_area() – вычисляет площадь фигуры.

Дополнительно создадим три переменные (длины сторон треугольника): a = 7, b = 2, c = 8, которые также не будут видны при импорте.

На вход функциям передается длина трех сторон (если пользователь ничего не введет, то используются значения по умолчанию).

- для пакета 'square': функции square_perimeter() – вычисляет периметр квадрата, square_area() – вычисляет площадь фигуры.

Дополнительная переменная a = 15 не доступна при импорте и принимается функциями, если пользователь не предоставил свои размеры стороны квадрата.

Ваша итоговая задача — позволить человеку, загрузившему ваш пакет, иметь возможность напрямую импортировать все функции из подпакетов.

Например, он может написать так: `'from figures import circle_area'`.

Также вы, как разработчик, после написания всей библиотеки решили поменять ее имя на `'figures'`.

Постарайтесь сделать код таким, чтобы это не заставило вас переписывать все внутренние импорты с учетом нового именования.