

Лабораторная работа 2.13

Тема: Модули и пакеты

Цель работы: приобретение навыков по работе с модулями и пакетами языка программирования Python версии 3.x.

Порядок выполнения работы


1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 RomanGorchakov ▾

Repository name *

/ Test

✔ Test is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

Description (optional)

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

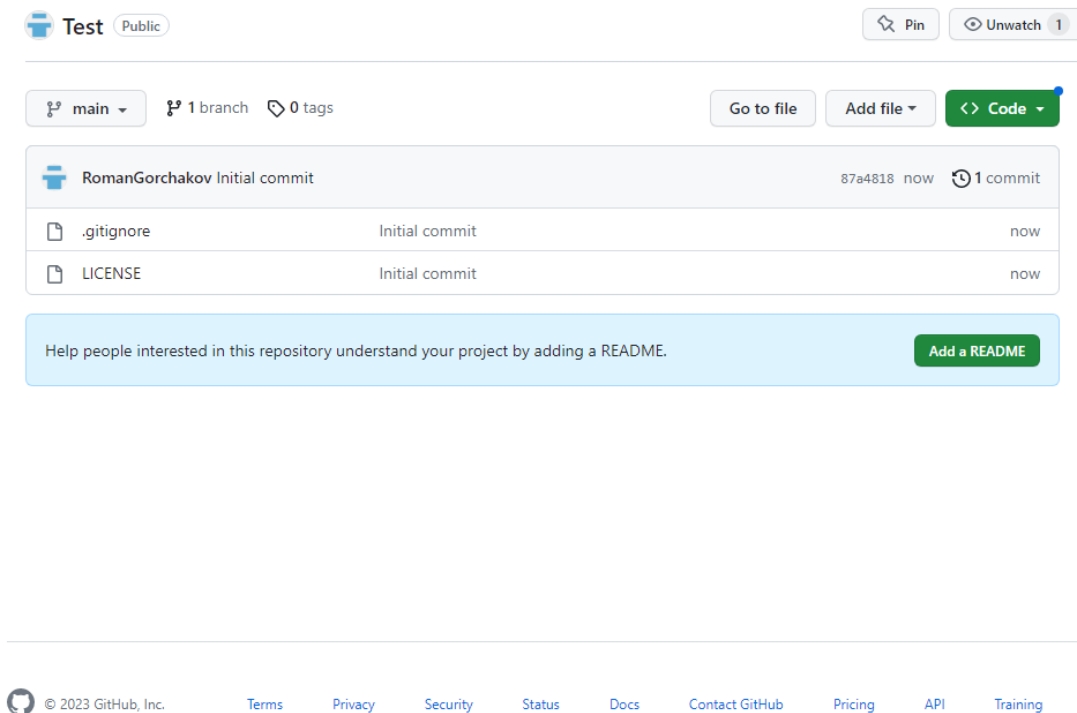
Choose a license

License: MIT License ▾


















A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository



2. Теперь необходимо дополнить файл `.gitignore` с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «<https://github.com/github/gitignore>» и скачиваем оттуда файл «Python.gitignore».

	Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
	PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
	Plone.gitignore	Covered by global vim template	10 years ago
	Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
	Processing.gitignore	Ignore transpiled .java and .class files (#3016)	4 years ago
	PureScript.gitignore	Update PureScript adding .spago (#3278)	3 years ago
	Python.gitignore	Update Python.gitignore	last year
	Qooxdoo.gitignore	Add gitignore for qooxdoo apps	13 years ago
	Qt.gitignore	Remove trailing whitespace	2 years ago
	R.gitignore	Merge pull request #3792 from jl5000/patch-1	2 years ago
	README.md	Merge pull request #3854 from AnilSeervi/patch-1	2 years ago
	ROS.gitignore	Added ignore for files created by <code>catkin_make_isolated</code>	6 years ago
	Racket.gitignore	Update Racket.gitignore	2 years ago
	Rails.gitignore	Ignore Rails .env according recommendations	2 years ago
	Raku.gitignore	Changes the name of Perl 6 to Raku (#3312)	3 years ago
	RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
	Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files (#3197)	4 years ago

```
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28
```

3. Теперь создаём файл «README.md», где вносим информацию о своей группе и ФИО. Сохраняем набранный текст через кнопку «Commit changes».



4. После этого нужно организовать репозиторий в соответствии с моделью ветвления Git-flow. Для этого В окне «Codespace» выбираем опцию «Create codespace on main», где введём команды: «git branch develop» и «git push -u origin develop» для создания ветки разработки; «git branch feature_branch» для создания ветки функций; «git branch release/1.0.0» для создания ветки релиза; «git checkout main» и «git branch hotfix» для создания веток hotfix.

```
@RomanGorchakov →/workspaces/Py13 (main) $ git checkout -b develop
Switched to a new branch 'develop'
● @RomanGorchakov →/workspaces/Py13 (develop) $ git branch feature_branch
● @RomanGorchakov →/workspaces/Py13 (develop) $ git branch release/1.0.0
● @RomanGorchakov →/workspaces/Py13 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● @RomanGorchakov →/workspaces/Py13 (main) $ git branch hotfix
● @RomanGorchakov →/workspaces/Py13 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/Py13 (develop) $
```

5. Создаём файл «individual.py», в котором программа объявляет внутреннюю функцию, которая из переданного ей списка строк формирует многострочную строку вида:

Строка 1

...

Строка N

и возвращает её, где строка1, строка2, ... – строки из переданного функции списка. Оформить все функции программы в виде отдельного модуля. Разработанный модуль должен быть подключен в основную программу с помощью одного из вариантов команды import.

```
C:\Users\student-09-523>python "H:\Основы программной инженерии\2.13\individual1
.py"
Введите строку: w
Введите строку: a
Введите строку: s
Введите строку: d
Введите строку: space
Введите строку: enter
Введите строку:
1 w
2 a
3 s
4 d
5 space
6 enter
C:\Users\student-09-523>
```

6. Создаём файл «individual2.py», в котором нужно использовать словарь, содержащий следующие ключи: название пункта назначения рейса; номер рейса; тип самолета. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по возрастанию номера рейса; вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение.

```

C:\Users\student-09-523>python "H:\Основы программной инженерии\2.13\individual12.py"
>>> add
Название пункта назначения рейса: novoaleksandrovs
>>> add
Название пункта назначения рейса: stavoropol
>>> add
Название пункта назначения рейса: moscow
>>> add
Название пункта назначения рейса: saint-petersburg
>>> add
Название пункта назначения рейса: kazan
>>> add
Название пункта назначения рейса: kaliningrad
>>> list
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолёта |
+-----+-----+-----+-----+
| 1 | kaliningrad | 1545 | 76 |
| 2 | kazan | 4861 | 41 |
| 3 | moscow | 5299 | 57 |
| 4 | novoaleksandrovs | 2251 | 29 |
| 5 | saint-petersburg | 2728 | 1 |
| 6 | stavoropol | 6630 | 44 |
+-----+-----+-----+-----+
>>> help
Список команд:
add - добавить рейс;
list - вывести список рейсов;
select <товар> - информация о рейсе;
help - отобразить справку;
exit - завершить работу с программой.
>>> exit


```



7. Выполняем коммит файлов в репозиторий Git в ветку разработки, сливаем её с веткой main и отправляем изменения на сервер GitHub.

```

● @RomanGorchakov →/workspaces/Py13 (develop) $ git add .
● @RomanGorchakov →/workspaces/Py13 (develop) $ git commit -m "Packages and modules"
[develop 04474e6] Packages and modules
7 files changed, 166 insertions(+)
create mode 100644 individual1.py
create mode 100644 individual2.py
create mode 100644 mymodule.py
create mode 100644 packet/__init__.py
create mode 100644 packet/display_plane.py
create mode 100644 packet/get_plane.py
create mode 100644 packet/show_plane.py
● @RomanGorchakov →/workspaces/Py13 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● @RomanGorchakov →/workspaces/Py13 (main) $ git merge develop
Updating fa803ce..04474e6
Fast-forward
 individual1.py      | 9 ++++++++
 individual2.py      | 10 ++++++++
 mymodule.py         | 14 ++++++++
 packet/__init__.py  | 1 +
 packet/display_plane.py | 30 ++++++++
 packet/get_plane.py  | 10 ++++++++
 packet/show_plane.py | 92 ++++++++
7 files changed, 166 insertions(+)
create mode 100644 individual1.py
create mode 100644 individual2.py
create mode 100644 mymodule.py
create mode 100644 packet/__init__.py
create mode 100644 packet/display_plane.py
create mode 100644 packet/get_plane.py
create mode 100644 packet/show_plane.py
● @RomanGorchakov →/workspaces/Py13 (main) $ git push -u
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 2 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 2.09 KiB | 2.09 MiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/RomanGorchakov/Py13
fa803ce..04474e6 main -> main
branch 'main' set up to track 'origin/main'.
● @RomanGorchakov →/workspaces/Py13 (main) $

```

 Py13 Public

 Pin  Unwatch 1

 main  1 Branch  0 Tags



 Add file  Code

 RomanGorchakov Add files via upload a24d8b7 · now  5 Commits

 packet	Packages and modules	9 minutes ago
 .gitignore	Create .gitignore	last week
 LICENSE	Create LICENSE	last week
 README.md	Create README.md	last week
 individual1.py	Packages and modules	9 minutes ago
 individual2.py	Packages and modules	9 minutes ago
 mymodule.py	Packages and modules	9 minutes ago
 ЛР2.13_ГорчаковРВ.pdf	Add files via upload	now

 README  MIT license 

Информация обо мне

Имя: Роман.

Фамилия: Горчаков.

Отчество: Владимирович.

Название группы: ПИЖ-6-о-22-1.

Контрольные вопросы

1. Что является модулем языка Python?

Под модулем в Python понимается файл с расширением `.py`. Модули предназначены для того, чтобы в них хранить часто используемые функции, классы, константы и т. п. Можно условно разделить модули и программы: программы предназначены для непосредственного запуска, а модули для импортирования их в другие программы.

2. Какие существуют способы подключения модулей в языке Python?

Самый простой способ импортировать модуль в Python это воспользоваться конструкцией `import имя_модуля`. За один раз можно импортировать сразу несколько модулей, для этого их нужно перечислить через запятую после слова

`import`. Если вы хотите задать псевдоним для модуля в вашей программе, можно воспользоваться синтаксисом «`import имя_модуля as новое_имя_модуля`».

Используя любой из вышеперечисленных подходов, при вызове функции из импортированного модуля, вам всегда придется указывать имя модуля (или псевдоним). Для того, чтобы этого избежать делайте импорт через конструкцию «`from имя_модуля import имя_объекта`». При этом импортируется только конкретный объект, остальные функции недоступны, даже если при их вызове указать имя модуля. Для импортирования нескольких функций из модуля, можно перечислить их имена через запятую. Импортируемому объекту можно задать псевдоним. Если необходимо импортировать все функции, классы и т. п. из модуля, то воспользуйтесь следующей формой оператора «`from имя_модуля import *`».

3. Что является пакетом языка Python?

Пакет в Python – это каталог, включающий в себя другие каталоги и модули, но при этом дополнительно содержащий файл `__init__.py`. Пакеты используются для формирования пространства имен, что позволяет работать с модулями через указание уровня вложенности (через точку). Для импортирования пакетов используется тот же синтаксис, что и для работы с модулями.

4. Каково назначение файла `__init__.py`?

Файл `__init__.py` выполняет две основные функции:

- инициализация пакета. Код, который нужно выполнить при импортировании модулей из пакета, можно поместить в `__init__.py`;
- определение пакета. Наличие `__init__.py` говорит Python, что директория является пакетом, даже если он пуст.

5. Каково назначение переменной `__all__` файла `__init__.py`?

Переменную `__all__` хранит список модулей, который импортируется при загрузке через конструкцию «`from имя_пакета import *`».