

Лабораторная работа 4.1

Тема: Элементы объектно-ориентированного программирования в языке Python

Цель работы: приобретение навыков по работе с классами и объектами при написании программ с помощью 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 *	Repository name *
<div> RomanGorchakov</div>	<div>Test</div>
	✔ 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

 **Test** Public

 Pin  Unwatch 1

 main  1 branch  0 tags

Go to file

Add file 

 Code 

 **RomanGorchakov** Initial commit 87a4818 now  1 commit

 .gitignore


















Initial commit now

 LICENSE

Initial commit now

Help people interested in this repository understand your project by adding a README. [Add a README](#) © 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#)

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».

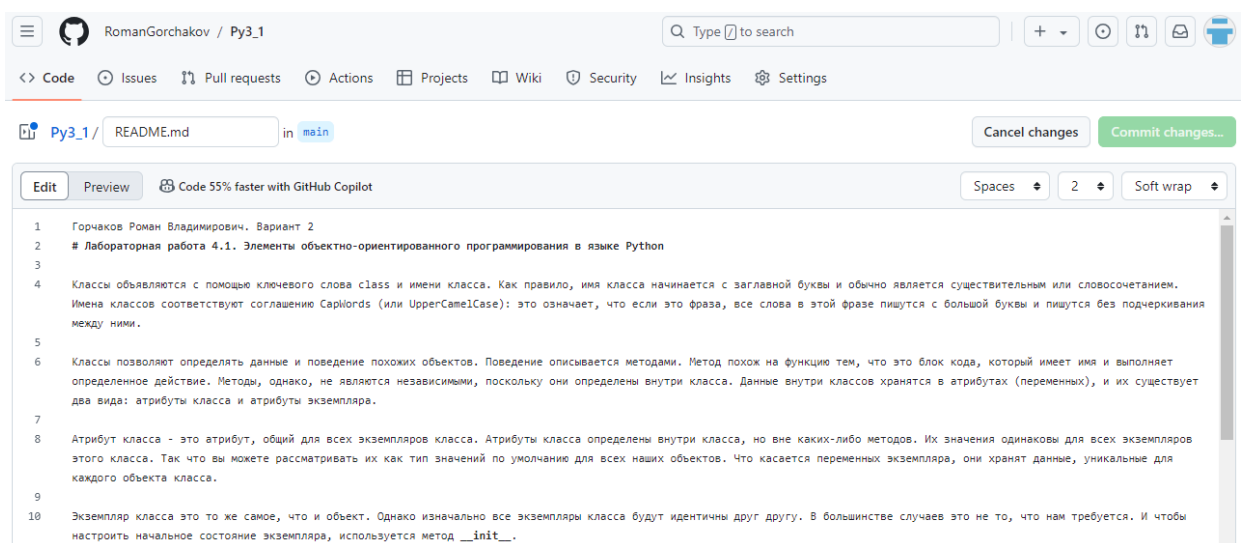
	Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
	PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
	Plone.gitignore	Covered by global vim template	10 years ago
	Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
	Processing.gitignore	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 ФИО и теоретический конспект лекции. Сохраняем набранный текст через кнопку «Commit changes».



The screenshot shows the GitHub web interface for a repository named 'Py3_1' by user 'RomanGorchakov'. The 'README.md' file is open in the 'main' branch. The file content is as follows:

```

1  Горчаков Роман Владимирович. Вариант 2
2  # Лабораторная работа 4.1. Элементы объектно-ориентированного программирования в языке Python
3
4  Классы объявляются с помощью ключевого слова class и имени класса. Как правило, имя класса начинается с заглавной буквы и обычно является существительным или словосочетанием.
   Имена классов соответствуют соглашению Caplords (или UpperCamelCase): это означает, что если это фраза, все слова в этой фразе пишутся с большой буквы и пишутся без подчеркивания
   между ними.
5
6  Классы позволяют определять данные и поведение похожих объектов. Поведение описывается методами. Метод похож на функцию тем, что это блок кода, который имеет имя и выполняет
   определенное действие. Методы, однако, не являются независимыми, поскольку они определены внутри класса. Данные внутри классов хранятся в атрибутах (переменных), и их существует
   два вида: атрибуты класса и атрибуты экземпляра.
7
8  Атрибут класса - это атрибут, общий для всех экземпляров класса. Атрибуты класса определены внутри класса, но вне каких-либо методов. Их значения одинаковы для всех экземпляров
   этого класса. Так что вы можете рассматривать их как тип значений по умолчанию для всех наших объектов. Что касается переменных экземпляра, они хранят данные, уникальные для
   каждого объекта класса.
9
10 Экземпляр класса это то же самое, что и объект. Однако изначально все экземпляры класса будут идентичны друг другу. В большинстве случаев это не то, что нам требуется. И чтобы
    настроить начальное состояние экземпляра, используется метод __init__.

```

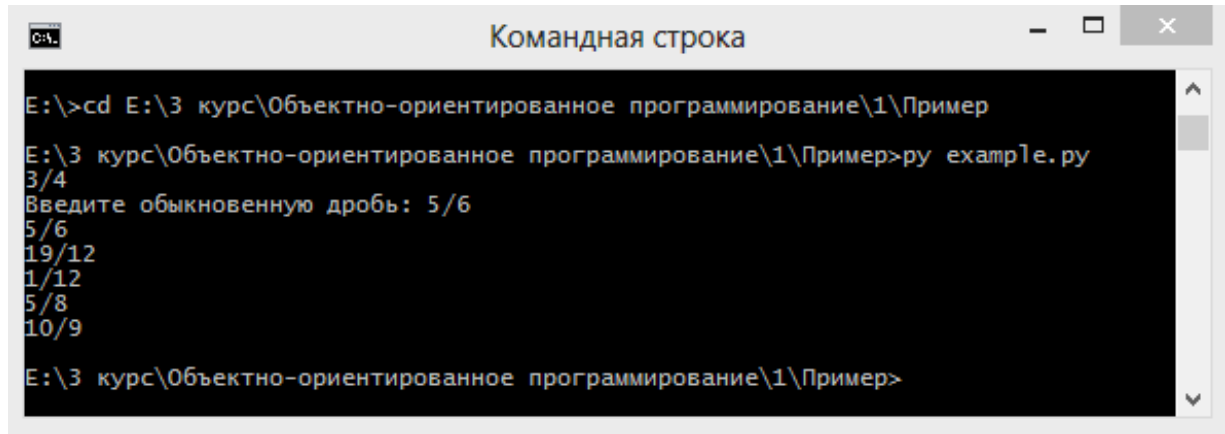
The interface includes a search bar, navigation tabs (Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings), and a bottom bar with 'Edit', 'Preview', and 'Code 55% faster with GitHub Copilot' options. The 'Commit changes...' button is visible in the top right of the editor area.

4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой репозиторий. После этого организуем репозиторий в соответствие с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout -b 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. Устанавливаем библиотеки isort, black и flake8 и создаём файлы .pre-commit-config.yaml и environment.yml.

```
@RomanGorchakov →/workspaces/Py3_1 (main) $ git checkout -b develop
Switched to a new branch 'develop'
● @RomanGorchakov →/workspaces/Py3_1 (develop) $ git branch feature_branch
● @RomanGorchakov →/workspaces/Py3_1 (develop) $ git branch release/1.0.0
● @RomanGorchakov →/workspaces/Py3_1 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● @RomanGorchakov →/workspaces/Py3_1 (main) $ git branch hotfix
● @RomanGorchakov →/workspaces/Py3_1 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/Py3_1 (develop) $
```

```
● @RomanGorchakov →/workspaces/Py3_1 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
bash: pre-commit: command not found
● @RomanGorchakov →/workspaces/Py3_1 (develop) $ pip install pre-commit
Collecting pre-commit
  Downloading pre_commit-3.8.0-py2.py3-none-any.whl.metadata (1.3 kB)
Collecting cfgv>=2.0.0 (from pre-commit)
  Downloading cfgv-3.4.0-py2.py3-none-any.whl.metadata (8.5 kB)
Collecting identify>=1.0.0 (from pre-commit)
  Downloading identify-2.6.0-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting nodeenv>=0.11.1 (from pre-commit)
  Downloading nodeenv-1.9.1-py2.py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: pyyaml>=5.1 in /home/codespace/.local/lib/python3.12/site-packages (from pre-commit) (6.0.2)
Collecting virtualenv>=20.10.0 (from pre-commit)
  Downloading virtualenv-20.26.4-py3-none-any.whl.metadata (4.5 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv>=20.10.0->pre-commit)
  Downloading distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: filelock<4,>=3.12.2 in /home/codespace/.local/lib/python3.12/site-packages (from virtualenv>=20.10.0->pre-commit) (3.13.1)
Requirement already satisfied: platformdirs<5,>=3.9.1 in /home/codespace/.local/lib/python3.12/site-packages (from virtualenv>=20.10.0->pre-commit) (4.2.2)
Downloading pre_commit-3.8.0-py2.py3-none-any.whl (204 kB)
Downloading cfgv-3.4.0-py2.py3-none-any.whl (7.2 kB)
Downloading identify-2.6.0-py2.py3-none-any.whl (98 kB)
Downloading nodeenv-1.9.1-py2.py3-none-any.whl (22 kB)
Downloading virtualenv-20.26.4-py3-none-any.whl (6.0 MB)
 6.0/6.0 MB 22.7 MB/s eta 0:00:00
Installing collected packages: distlib, virtualenv, nodeenv, identify, cfgv, pre-commit
Successfully installed cfgv-3.4.0 distlib-0.3.8 identify-2.6.0 nodeenv-1.9.1 pre-commit-3.8.0 virtualenv-20.26.4
● @RomanGorchakov →/workspaces/Py3_1 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
● @RomanGorchakov →/workspaces/Py3_1 (develop) $ conda env export > environment.yml
○ @RomanGorchakov →/workspaces/Py3_1 (develop) $
```

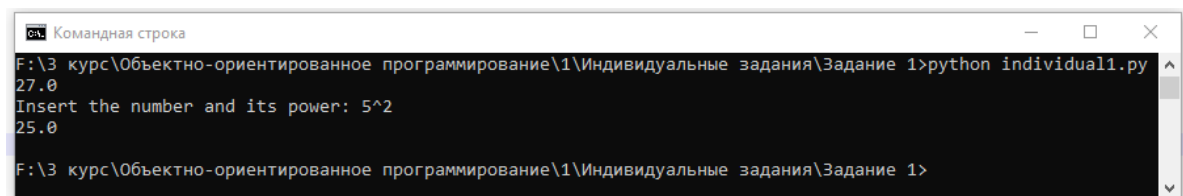
5. Создаём файл «example.py», в котором нужно создать класс Rational для работы с рациональными дробями с реализацией операций сложения, вычитания, умножения, деления, сравнения и сокращения дробей.



```
Командная строка

E:\>cd E:\3 курс\Объектно-ориентированное программирование\1\Пример
E:\3 курс\Объектно-ориентированное программирование\1\Пример>python example.py
3/4
Введите обыкновенную дробь: 5/6
5/6
19/12
1/12
5/8
10/9
E:\3 курс\Объектно-ориентированное программирование\1\Пример>
```

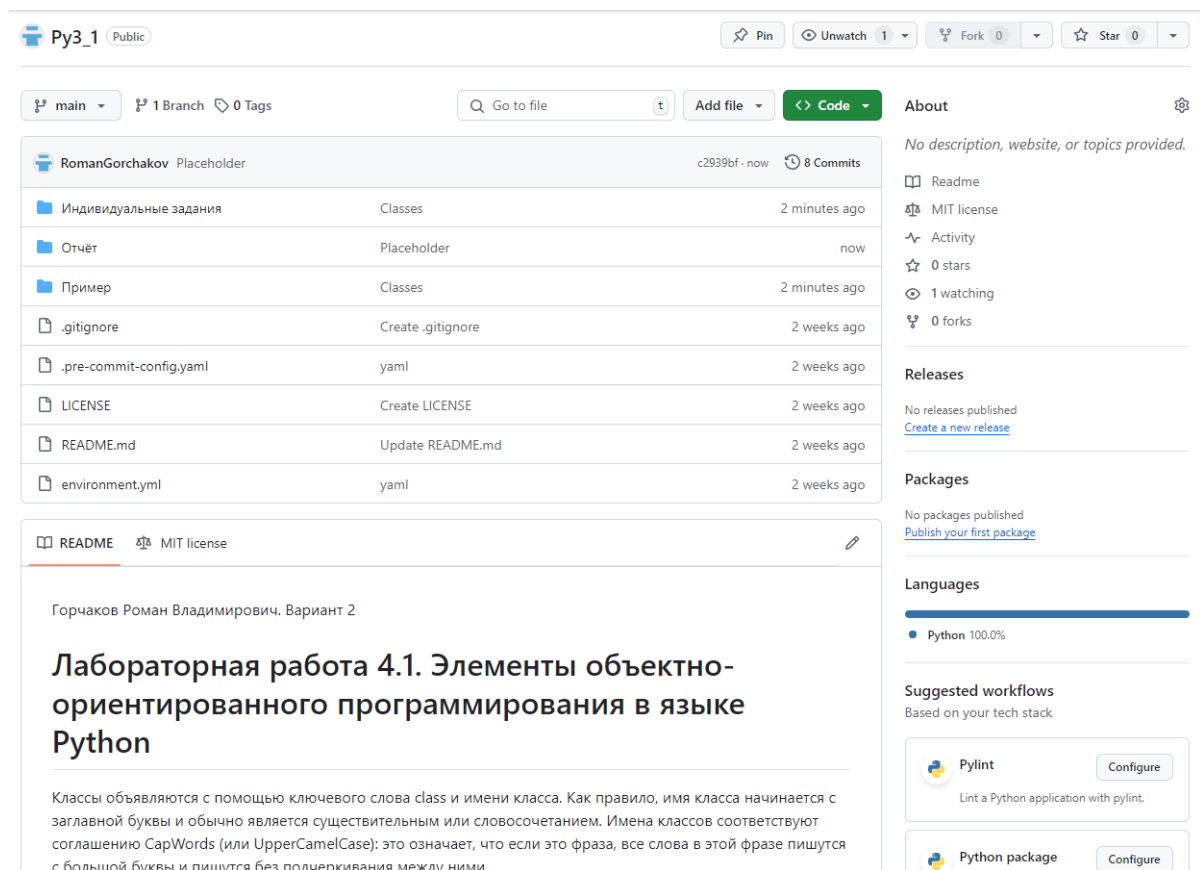
6. Создаём файл «individual1.py», в котором нужно создать класс с двумя дробными числами с реализацией возведения первого числа в степень, значение которой равно второму числу.



```
Командная строка

F:\3 курс\Объектно-ориентированное программирование\1\Индивидуальные задания\Задание 1>python individual1.py
27.0
Insert the number and its power: 5^2
25.0
F:\3 курс\Объектно-ориентированное программирование\1\Индивидуальные задания\Задание 1>
```

7. Создаём файл «individual2.py», в котором нужно создать класс для работы с моделями экранных окон с реализацией операций перемещения окна по горизонтали и вертикали, изменения высоты, ширины, цвета, состояния окна и опроса состояния.



Py3_1 Public

main 1 Branch 0 Tags

Go to file Add file Code

RomanGorchakov Placeholder c2939bf · now 8 Commits

File	Type	Time
Индивидуальные задания	Classes	2 minutes ago
Отчет	Placeholder	now
Пример	Classes	2 minutes ago
.gitignore	Create .gitignore	2 weeks ago
.pre-commit-config.yaml	yaml	2 weeks ago
LICENSE	Create LICENSE	2 weeks ago
README.md	Update README.md	2 weeks ago
environment.yml	yaml	2 weeks ago

README MIT license

Горчаков Роман Владимирович. Вариант 2

Лабораторная работа 4.1. Элементы объектно-ориентированного программирования в языке Python

Классы объявляются с помощью ключевого слова `class` и имени класса. Как правило, имя класса начинается с заглавной буквы и обычно является существительным или словосочетанием. Имена классов соответствуют соглашению CapWords (или UpperCamelCase): это означает, что, если это фраза, все слова в этой фразе пишутся с большой буквы и пишутся без подчеркивания между ними.

About: No description, website, or topics provided.

Releases: No releases published. [Create a new release](#)

Packages: No packages published. [Publish your first package](#)

Languages: Python 100.0%

Suggested workflows: Based on your tech stack

- Pylint: [Configure](#) (Lint a Python application with pylint.)
- Python package: [Configure](#)

Контрольные вопросы

1. Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова `class` и имени класса. Как правило, имя класса начинается с заглавной буквы и обычно является существительным или словосочетанием. Имена классов соответствуют соглашению CapWords (или UpperCamelCase): это означает, что, если это фраза, все слова в этой фразе пишутся с большой буквы и пишутся без подчеркивания между ними.

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса определены внутри класса, но вне каких-либо методов. Их значения одинаковы для всех экземпляров этого класса. Так что вы можете рассматривать их как тип значений по умолчанию для всех наших объектов. Что касается переменных экземпляра, они хранят данные, уникальные для каждого объекта класса.

3. Каково назначение методов класса?

Классы позволяют определять данные и поведение похожих объектов. Поведение описывается методами. Метод похож на функцию тем, что это блок кода, который имеет имя и выполняет определенное действие. Методы, однако, не являются независимыми, поскольку они определены внутри класса.

4. Для чего предназначен метод `__init__()` класса?

Метод `__init__` является конструктором. Конструкторы - это концепция объектноориентированного программирования. Класс может иметь один и только один конструктор. Если `__init__` определен внутри класса, он автоматически вызывается при создании нового экземпляра класса.

5. Каково назначение `self`?

Аргумент `self` представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам. Важно использовать параметр `self` внутри метода, если мы хотим сохранить значения экземпляра для последующего использования. В большинстве случаев нам также необходимо использовать параметр `self` в других методах, потому что при вызове метода первым аргументом, который ему передается, является сам объект.

6. Как добавить атрибуты в класс?

В дополнение к изменению атрибутов мы также можем создавать атрибуты для класса или конкретного экземпляра. Например, мы хотим видеть информацию о всех видах наших питомцев. Мы могли бы записать ее в самом классе с самого начала или создать переменную следующим образом:

```
Pet.all_specs = [tom.spec, avocado.spec, ben.spec]
```

```
tom.all_specs # ["cat", "dog", "goldfish"]  
avocado.all_specs # ["cat", "dog", "goldfish"]  
ben.all_specs
```

7. Как осуществляется управление доступом к методам и атрибутам в языке Python?

В Python таких возможностей нет, и любой может обратиться к атрибутам и методам вашего класса, если возникнет такая необходимость. Это существенный недостаток этого языка, т.к. нарушается один из ключевых принципов ООП –

инкапсуляция. Хорошим тоном считается, что для чтения/изменения какого-то атрибута должны использоваться специальные методы, которые называются `getter/setter`, их можно реализовать, но ничего не мешает изменить атрибут напрямую. При этом есть соглашение, что метод или атрибут, который начинается с нижнего подчеркивания, является скрытым, и снаружи класса трогать его не нужно (хотя сделать это можно).

8. Каково назначение функции `isinstance`?

Встроенная функция `isinstance(obj, Cls)`, используемая при реализации методов арифметических операций и операций отношения, позволяет узнать что некоторый объект `obj` является либо экземпляром класса `Cls` либо экземпляром одного из потомков класса `Cls`.