

Лабораторная работа 4.3

Тема: Наследование и полиморфизм в языке Python

Цель работы: приобретение навыков по созданию иерархии классов при написании программ с помощью языка программирования Python версии 3.x

Ссылка на GitHub: https://github.com/RomanGorchakov/Py3_4


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 *
 RomanGorchakov ▾	/ 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.](#)

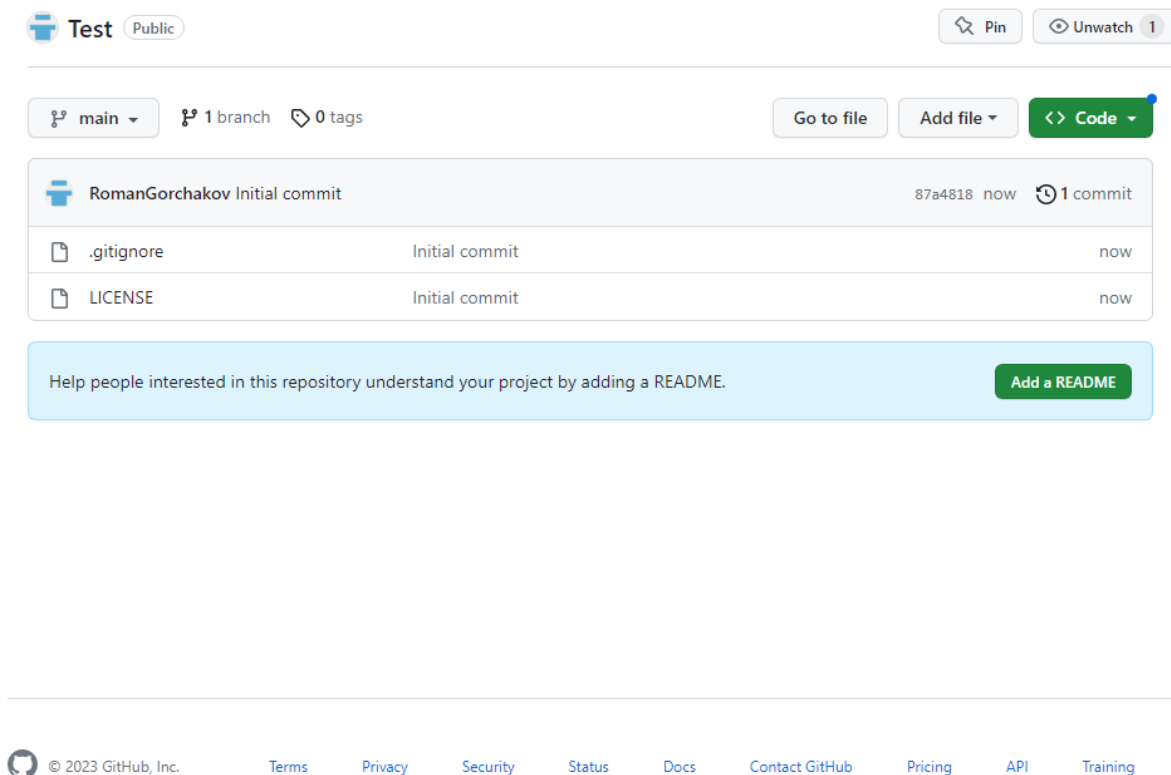
Choose a license

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`».

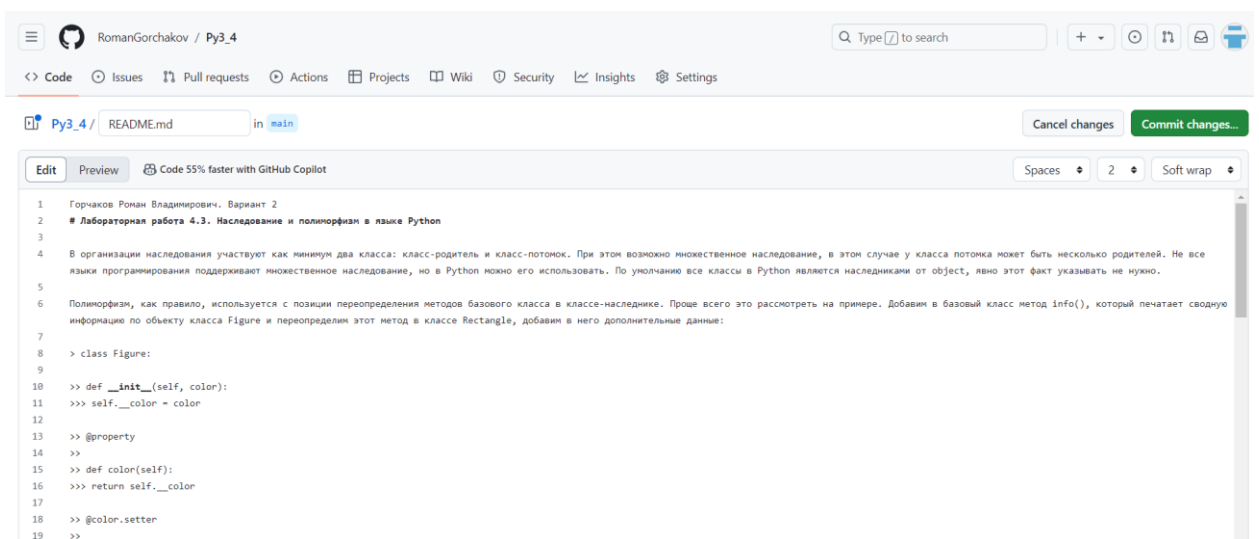
	Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
	PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
	Plone.gitignore	Covered by global vim template	10 years ago
	Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
	Processing.gitignore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



The screenshot shows the GitHub web interface for a repository named 'Py3_4' by user 'RomanGorchakov'. The 'README.md' file is open in the 'main' branch. The editor shows the following content:

```

1 Горчаков Роман Владимирович. Вариант 2
2 # Лабораторная работа 4.3. Наследование и полиморфизм в языке Python
3
4 В организации наследования участвуют как минимум два класса: класс-родитель и класс-потомок. При этом возможно множественное наследование, в этом случае у класса потомка может быть несколько родителей. Не все
5 языки программирования поддерживают множественное наследование, но в Python можно его использовать. По умолчанию все классы в Python являются наследниками от object, явно этот факт указывать не нужно.
6
7 Полиморфизм, как правило, используется с позиции переопределения методов базового класса в классе-наследнике. Проще всего это рассмотреть на примере. Добавим в базовый класс метод info(), который печатает сводную
8 информацию по объекту класса Figure и переопределим этот метод в классе Rectangle, добавив в него дополнительные данные:
9
10 > class Figure:
11 >> def __init__(self, color):
12 >>> self.__color = color
13
14 >> @property
15 >> def color(self):
16 >>> return self.__color
17
18 >> @color.setter
19 >>

```

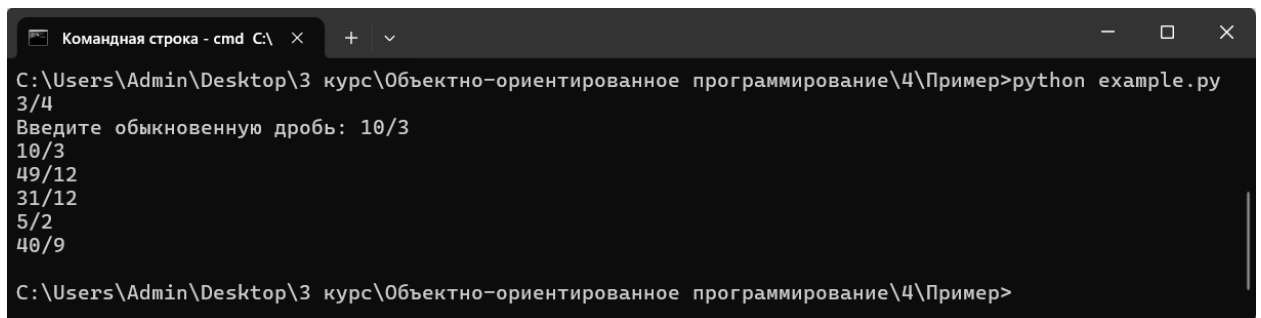
At the top right of the editor, there are buttons for 'Cancel changes' and 'Commit changes...'. The interface also shows a search bar, navigation tabs for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings', and a status bar at the bottom indicating 'Code 55% faster with GitHub Copilot'.

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_4 (main) $ git checkout -b develop
Switched to a new branch 'develop'
• @RomanGorchakov → /workspaces/Py3_4 (develop) $ git branch feature_branch
• @RomanGorchakov → /workspaces/Py3_4 (develop) $ git branch release/1.0.0
• @RomanGorchakov → /workspaces/Py3_4 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
• @RomanGorchakov → /workspaces/Py3_4 (main) $ git branch hotfix
• @RomanGorchakov → /workspaces/Py3_4 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov → /workspaces/Py3_4 (develop) $
```

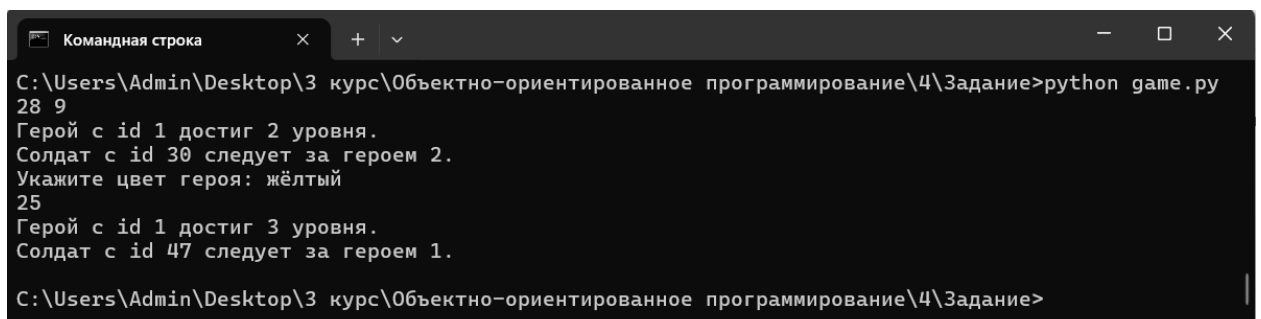
```
Collecting black
  Downloading black-24.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata (79 kB)
Collecting click>=8.0.0 (from black)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting mypy_extensions>=0.4.3 (from black)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: packaging>=22.0 in /home/codespace/.local/lib/python3.12/site-packages (from black) (24.1)
Collecting pathspec>=0.9.0 (from black)
  Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: platformdirs>=2 in /home/codespace/.local/lib/python3.12/site-packages (from black) (4.3.6)
Downloading black-24.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (1.8 MB)
1.8/1.8 MB 25.7 MB/s eta 0:00:00
Downloading click-8.1.7-py3-none-any.whl (97 kB)
Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Installing collected packages: pathspec, mypy_extensions, click, black
Successfully installed black-24.10.0 click-8.1.7 mypy_extensions-1.0.0 pathspec-0.12.1
• @RomanGorchakov → /workspaces/Py3_4 (develop) $ pip install flake8
Collecting flake8
  Downloading flake8-7.1.1-py2.py3-none-any.whl.metadata (3.8 kB)
Collecting mccabe<0.8.0,>=0.7.0 (from flake8)
  Downloading mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
Collecting pycodestyle<2.13.0,>=2.12.0 (from flake8)
  Downloading pycodestyle-2.12.1-py2.py3-none-any.whl.metadata (4.5 kB)
Collecting pyflakes<3.3.0,>=3.2.0 (from flake8)
  Downloading pyflakes-3.2.0-py2.py3-none-any.whl.metadata (3.5 kB)
Downloading flake8-7.1.1-py2.py3-none-any.whl (57 kB)
Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Downloading pycodestyle-2.12.1-py2.py3-none-any.whl (31 kB)
Downloading pyflakes-3.2.0-py2.py3-none-any.whl (62 kB)
Installing collected packages: pyflakes, pycodestyle, mccabe, flake8
Successfully installed flake8-7.1.1 mccabe-0.7.0 pycodestyle-2.12.1 pyflakes-3.2.0
• @RomanGorchakov → /workspaces/Py3_4 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
• @RomanGorchakov → /workspaces/Py3_4 (develop) $ conda env export > environment.yml
○ @RomanGorchakov → /workspaces/Py3_4 (develop) $
```

5. Создаём файл «example.py», в котором нужно создать класс Rational для работы с рациональными дробями. Обязательно должны быть реализованы операции: сложения, вычитания, умножения, деления и сравнения. Должна быть реализована приватная функция сокращения дроби reduce, которая обязательно вызывается при выполнении арифметических операций.



```
Командная строка - cmd C:\ x + v
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Пример>python example.py
3/4
Введите обыкновенную дробь: 10/3
10/3
49/12
31/12
5/2
40/9
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Пример>
```

6. Создаём файл «game.py». В основной ветке программы создаётся по одному герою для каждой команды. В цикле генерируются объекты-солдаты. Их принадлежность команде определяется случайно. Солдаты разных команд добавляются в разные списки. Измеряется длина списков солдат противоборствующих команд и выводится на экран. У героя, принадлежащего команде с более длинным списком, увеличивается уровень. Отправьте одного из солдат первого героя следовать за ним. Выведите на экран идентификационные номера этих двух юнитов.



```
Командная строка x + v
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Задание>python game.py
28 9
Герой с id 1 достиг 2 уровня.
Солдат с id 30 следует за героем 2.
Укажите цвет героя: жёлтый
25
Герой с id 1 достиг 3 уровня.
Солдат с id 47 следует за героем 1.
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Задание>
```

7. Создаём файл «individual1.py», в котором нужно создать класс Pair; определить методы изменения полей и сравнения пар: пара p1 больше пары p2,

если $(first.p1 > first.p2)$ или $(first.p1 = first.p2)$ и $(second.p1 > second.p2)$. Определить класс-наследник Fraction с полями: целая часть числа и дробная часть числа. Определить полный набор методов сравнения.

```
Командная строка - cmd C:\ x + v
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 1>python individual1.py
Вторая пара больше первой
0.1
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 1>
```

8. Создаём файл «individual2.py», в котором нужно создать абстрактный базовый класс Number с абстрактными методами – арифметическими операциями. Создать производные классы Integer (целое) и Real (действительное).

```
Командная строка - cmd C:\ x + v
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 2>python individual2.py
54
1892.25
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 2>
```

9. Проверяем правильность написания программ с помощью туру.

```
Командная строка x + v
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Задание>туру game.py
Success: no issues found in 1 source file
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Задание>cd C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания>cd C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 2
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 2>туру individual1.py
Success: no issues found in 1 source file
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 2>C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 3
"C:\Users\Admin\Desktop\3" не является внутренней или внешней командой, исполняемой программой или пакетным файлом.
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 3>cd C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 3
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 3>туру individual2.py
Success: no issues found in 1 source file
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\4\Индивидуальные задания\Задание 3>
```

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

```
● @RomanGorchakov →/workspaces/Py3_4 (main) $ git merge develop
Updating e6a528e..e204074
Fast-forward
 .pre-commit-config.yaml | 10 +++
 environment.yml          | 80 +++++
 .../320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/game.py" | 72 +++++
 .../320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/individual1.py" | 45 +++++
 .../320\227\320\260\320\264\320\260\320\275\320\270\320\265 3/individual2.py" | 69 +++++
 .../320\2404_2_320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf" | Bin 0 -> 803512 bytes
 ""\320\237\321\200\320\270\320\274\320\265\321\200\321\213\example.py" | 139 +++++
 ""\320\237\321\200\320\270\320\274\320\265\321\200\321\213\game.py" | 72 +++++
 8 files changed, 487 insertions(+)
 create mode 100644 .pre-commit-config.yaml
 create mode 100644 environment.yml
 create mode 100644 ""\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 320\267\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/game.py"
 create mode 100644 ""\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 320\267\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/individual1.py"
 create mode 100644 ""\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 320\267\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 3/individual2.py"
 create mode 100644 ""\320\236\321\202\321\221\321\202\320\233\320\2404_2_320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
 create mode 100644 ""\320\237\321\200\320\270\320\274\320\265\321\200\321\213\example.py"
 create mode 100644 ""\320\237\321\200\320\270\320\274\320\265\321\200\321\213\game.py"
● @RomanGorchakov →/workspaces/Py3_4 (main) $ git push -u
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 2 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (15/15), 733.19 KiB | 15.94 MiB/s, done.
Total 15 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/RomanGorchakov/Py3_4
 e6a528e..e204074 main -> main
branch 'main' set up to track 'origin/main'.
○ @RomanGorchakov →/workspaces/Py3_4 (main) $
```

Py3_4Public

PinUnwatch1Fork0Star0

main1 BranchTagsGo to fileAdd fileCode

RomanGorchakov мурр433f699 · now7 Commits

Индивидуальные задания	Classes	14 minutes ago
Отчёт	Polymorphism	10 minutes ago
Примеры	Classes	14 minutes ago
.gitignore	Create .gitignore	last week
.pre-commit-config.yaml	Classes	14 minutes ago
LICENSE	Create LICENSE	last week
README.md	Update README.md	10 hours ago
environment.yml	Classes	14 minutes ago
mypy.ini	mypy	now

READMEMIT license

Горчаков Роман Владимирович. Вариант 2

Лабораторная работа 4.3. Наследование и полиморфизм в языке Python

AboutNo description, website, or topics provided.

ReadmeMIT licenseActivity0 stars1 watching0 forks

ReleasesNo releases publishedCreate a new release

PackagesNo packages publishedPublish your first package

LanguagesPython 100.0%

Suggested workflowsBased on your tech stack

Контрольные вопросы

1. Что такое наследование и как оно реализовано в языке Python?

Наследование — это концепция объектно-ориентированного программирования, согласно которой абстрактный тип данных может наследовать

данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов программного обеспечения.

В организации наследования участвуют как минимум два класса: класс родитель и класс потомок. При этом возможно множественное наследование, в этом случае у класса потомка может быть несколько родителей. Синтаксически создание класса с указанием его родителя выглядит так: `class имя_класса(имя_родителя1, [имя_родителя2,..., имя_родителя_n])`.

2. Что такое полиморфизм и как он реализован в языке Python?

Полиморфизм в языках программирования – способность выполнять одно и то же действие над объектами разных типов. Как правило, он используется с позиции переопределения методов базового класса в классе наследнике.

В языке Python полиморфизм реализуется двумя способами: наследованием (от базового класса наследуются классы-потомки, которые реализуют у себя его методы) и с помощью утиной типизации (переменные связываются с типом не в момент объявления, а в момент присваивания значения).

3. Что такое «утиная» типизация в языке программирования Python?

Утиная типизация – концепция, характерная для языков программирования с динамической типизацией, согласно которой конкретный тип или класс объекта не важен, а важны лишь свойства и методы, которыми этот объект обладает. Другими словами, при работе с объектом его тип не проверяется, вместо этого проверяются свойства и методы этого объекта.

4. Каково назначение модуля abc языка программирования Python?

По умолчанию Python не предоставляет абстрактных классов. Python поставляется с модулем, который обеспечивает основу для определения абстрактных базовых классов (ABC), и имя этого модуля – ABC. ABC работает, декорируя методы базового класса как абстрактные, а затем регистрируя конкретные классы как реализации абстрактной базы.

5. Как сделать некоторый метод класса абстрактным?

Метод становится абстрактным, если он украшен ключевым словом `@abstractmethod`.

6. Как сделать некоторое свойство класса абстрактным?

Абстрактные классы включают в себя атрибуты в дополнение к методам, вы можете потребовать атрибуты в конкретных классах, определив их с помощью `@abstractproperty`.

7. 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`.