

Лабораторная работа 4.5

Тема: Аннотация типов

Цель работы: приобретение навыков по работе с аннотациями типов при написании программ с помощью языка программирования Python версии 3.x. Рассмотрен вопрос контроля типов переменных и функций с использованием комментариев и аннотаций. Приведено описание PEP'ов, регламентирующих работу с аннотациями, и представлены примеры работы с инструментом туру для анализа Python кода.

Ссылка на GitHub: https://github.com/RomanGorchakov/Py3_6

Порядок выполнения работы

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

The screenshot shows the GitHub 'Start writing code' interface. It is divided into two main sections: 'Start a new repository for RomanGorchakov' and 'Introduce yourself with a profile README'.

Start a new repository for RomanGorchakov

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

Py3_6

Py3_6 is available.

☒ Public
Anyone on the internet can see this repository

☐ Private
You choose who can see and commit to this repository

Create a new repository

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

RomanGorchakov / README.md

Create

```
1 - 🙋 Hi, I'm @RomanGorchakov
2 - 📖 I'm interested in ...
3 - 🌱 I'm currently learning ...
4 - 💡 I'm looking to collaborate on ...
5 - 💬 How to reach me ...
6 - 🗨 Pronouns: ...
7 - ⚡ Fun fact: ...
8
```

Use tools of the trade

Simplify your development workflow with a GUI

Install GitHub Desktop to visualize, commit, and push changes without ever touching the command line.

Get AI-based coding suggestions

Try GitHub Copilot free for 30 days, which suggests entire functions in real time, right from your editor.

RomanGorchakov / Py3_6
 Type to search

[Code](#)
[Issues](#)
[Pull requests](#)
[Actions](#)
[Projects](#)
[Wiki](#)
[Security](#)
[Insights](#)
[Settings](#)

Add a license to your project

Apache License 2.0	<p>A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.</p> <table border="1"> <thead> <tr> <th>Permissions</th> <th>Limitations</th> <th>Conditions</th> </tr> </thead> <tbody> <tr> <td>✓ Commercial use</td> <td>✗ Liability</td> <td>① License and copyright notice</td> </tr> <tr> <td>✓ Modification</td> <td>✗ Warranty</td> <td></td> </tr> <tr> <td>✓ Distribution</td> <td></td> <td></td> </tr> <tr> <td>✓ Private use</td> <td></td> <td></td> </tr> </tbody> </table> <p>This is not legal advice. Learn more about repository licenses.</p>	Permissions	Limitations	Conditions	✓ Commercial use	✗ Liability	① License and copyright notice	✓ Modification	✗ Warranty		✓ Distribution			✓ Private use			<p>To adopt MIT License, enter your details. You'll have a chance to review before committing a <i>LICENSE</i> file to a new branch or the root of your project.</p> <p>Year ① <input type="text" value="2024"/></p> <p>Full name ① <input type="text" value="RomanGorchakov"/></p> <p>Review and submit</p>
Permissions		Limitations	Conditions														
✓ Commercial use		✗ Liability	① License and copyright notice														
✓ Modification		✗ Warranty															
✓ Distribution																	
✓ Private use																	
GNU General Public License v3.0																	
MIT License																	
BSD 2-Clause "Simplified" License																	
BSD 3-Clause "New" or "Revised" License																	
Boost Software License 1.0																	
Creative Commons Zero v1.0 Universal																	
Eclipse Public License 2.0																	
GNU Affero General Public License v3.0																	
GNU General Public License v2.0																	

MIT License

Copyright (c)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

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».

RomanGorchakov / Py3_6
 Type to search

[Code](#)
[Issues](#)
[Pull requests](#)
[Actions](#)
[Projects](#)
[Wiki](#)
[Security](#)
[Insights](#)
[Settings](#)

Your license is ready. Please review it below and either commit it to the main branch or to a new branch.

Py3_6 / `.gitignore` in `main` [Cancel changes](#) [Commit changes...](#)

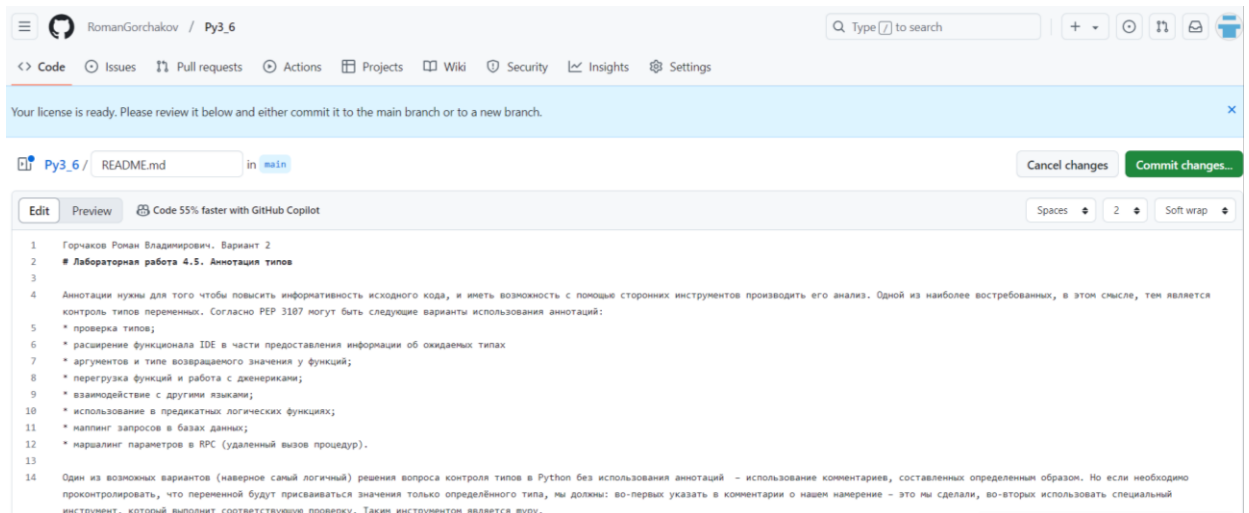
Choose .gitignore template: Python [Code 55% faster with GitHub Copilot](#)

Spaces No wrap

```

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/
    
```

3. Теперь создаём файл «README.md», где вносим ФИО и теоретический конспект лекции. Сохраняем набранный текст через кнопку «Commit changes».



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 соответствии 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_6 (main) $ git checkout -b develop
Switched to a new branch 'develop'
• @RomanGorchakov →/workspaces/Py3_6 (develop) $ git branch feature_branch
• @RomanGorchakov →/workspaces/Py3_6 (develop) $ git branch release/1.0.0
• @RomanGorchakov →/workspaces/Py3_6 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
• @RomanGorchakov →/workspaces/Py3_6 (main) $ git branch hotfix
• @RomanGorchakov →/workspaces/Py3_6 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/Py3_6 (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 55.3 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_6 (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_6 (develop) $ pre-commit sample-config > .pre-commit.config.yaml
● @RomanGorchakov → /workspaces/Py3_6 (develop) $ conda env export > environment.yml
○ @RomanGorchakov → /workspaces/Py3_6 (develop) $

```

5. Создаём файл «example.py», в котором нужно добавить аннотации типов для примера 1 лабораторной работы 14.

```

Command Prompt
C:\Users\Admin>cd C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\6\Пример
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\6\Пример>python example.py
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? Директор
Год поступления? 2007
>>> add
Фамилия и инициалы? Петров П.П.
Должность? Бухгалтер
Год поступления? 2010
>>> add
Фамилия и инициалы? Сидоров С.С.
Должность? Главный инженер
Год поступления? 2012
>>> list
+-----+-----+-----+-----+
| %s | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Иванов И.И. | Директор | 2007 |
| 2 | Петров П.П. | Бухгалтер | 2010 |
| 3 | Сидоров С.С. | Главный инженер | 2012 |
+-----+-----+-----+-----+
>>> exit
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\6\Пример>

```

6. Создаём файл «individual.py», в котором нужно добавить аннотацию типов в индивидуальном задании 2 лабораторной работы 2.19. Выполняем проверку программы с помощью утилиты mypy.

```
Command Prompt
2024-12-11 10:22:41,903 [INFO]: Database created successfully.
2024-12-11 10:22:41,903 [INFO]: Plane added successfully: Kazan, 3780, 1
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\6\Индивидуальное задание>python individual.py add --db planes.db -r "Moscow" -n 8850 -t 9
2024-12-11 10:22:47,561 [INFO]: Database created successfully.
2024-12-11 10:22:47,595 [INFO]: Plane added successfully: Moscow, 8850, 9
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\6\Индивидуальное задание>python individual.py add --db planes.db -r "Novoalexandrovsk" -n 3234 -t 62
2024-12-11 10:22:50,483 [INFO]: Database created successfully.
2024-12-11 10:22:50,529 [INFO]: Plane added successfully: Novoalexandrovsk, 3234, 62
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\6\Индивидуальное задание>python individual.py add --db planes.db -r "Saint-Petersburg" -n 9864 -t 48
2024-12-11 10:22:52,694 [INFO]: Database created successfully.
2024-12-11 10:22:52,711 [INFO]: Plane added successfully: Saint-Petersburg, 9864, 48
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\6\Индивидуальное задание>python individual.py add --db planes.db -r "Stavropol" -n 9120 -t 82
2024-12-11 10:22:55,229 [INFO]: Database created successfully.
2024-12-11 10:22:55,259 [INFO]: Plane added successfully: Stavropol, 9120, 82
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\6\Индивидуальное задание>python individual.py display --db planes.db
2024-12-11 10:22:59,302 [INFO]: Database created successfully.
+-----+-----+-----+-----+
| No | Destination | Race number | Plane type |
+-----+-----+-----+-----+
| 1 | Kazan | 3780 | 1 |
+-----+-----+-----+-----+
| 2 | Moscow | 8850 | 9 |
+-----+-----+-----+-----+
| 3 | Novoalexandrovsk | 3234 | 62 |
+-----+-----+-----+-----+
| 4 | Saint-Petersburg | 9864 | 48 |
+-----+-----+-----+-----+
| 5 | Stavropol | 9120 | 82 |
+-----+-----+-----+-----+
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\6\Индивидуальное задание>python -m mypy individual.py
Success: no issues found in 1 source file
```

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

```
Fast-forward
 .pre-commit.config.yaml
 .python-version
 edu.pyoop.code-workspace
 environment.yml
 pyproject.toml
 setup.cfg
 uv.lock
 .../individual.py"
 ... \320\2404.4 \320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
 "... \320\237\321\200\320\270\320\274\320\265\321\200\example.py"
 10 files changed, 1107 insertions(+)
 create mode 100644 .pre-commit.config.yaml
 create mode 100644 .python-version
 create mode 100644 edu.pyoop.code-workspace
 create mode 100644 environment.yml
 create mode 100644 pyproject.toml
 create mode 100644 setup.cfg
 create mode 100644 uv.lock
 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\320\276\320\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\individual.py"
 create mode 100644 "... \320\236\321\202\321\207\321\221\321\202\320\233\320\2404.4_ \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\example.py"
• @RomanGorchakov → /workspaces/Py3_6 (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), 1.06 MiB | 3.57 MiB/s, done.
Total 15 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/RomanGorchakov/Py3_6
 c5cb23d..b0b3a81 main -> main
branch 'main' set up to track 'origin/main'.
• @RomanGorchakov → /workspaces/Py3_6 (main) $
```

Py3_6 Public

main 1 Branch 0 Tags

Go to file Add file Code

RomanGorchakov Annotations b0b3a81 · now 4 Commits

Индивидуальное задание	Annotations	now
Отчёт	Annotations	now
Пример	Annotations	now
.gitignore	Create .gitignore	10 hours ago
.pre-commit.config.yaml	Annotations	now
.python-version	Annotations	now
LICENSE	Create LICENSE	10 hours ago
README.md	Create README.md	9 hours ago
edu.pyoop.code-workspace	Annotations	now
environment.yml	Annotations	now
pyproject.toml	Annotations	now
setup.cfg	Annotations	now
uv.lock	Annotations	now

README MIT license

Горчаков Роман Владимирович. Вариант 2

Лабораторная работа 4.5. Аннотация типов

About No description, website, or topics provided.

Readme MIT license Activity 0 stars 1 watching 0 forks

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 using Anaconda Configure

Контрольные вопросы

1. Для чего нужны аннотации типов в языке Python?

Аннотации нужны для того чтобы повысить информативность исходного кода, и иметь возможность с помощью сторонних инструментов производить его анализ.

2. Как осуществляется контроль типов в языке Python?

Один из возможных вариантов (наверное самый логичный) решения вопроса контроля типов в Python без использования аннотаций — использование комментариев, составленных определенным образом. Но если необходимо проконтролировать, что переменной будут присваиваться значения только определённого типа, мы должны: во-первых указать в комментарии о нашем намерении — это мы сделали, во-вторых использовать специальный инструмент,

который выполнит соответствующую проверку. Таким инструментом является `mypy`.

3. Какие существуют предложения по усовершенствованию Python для работы с аннотациями типов?

PEP 3107 – Function Annotations, является исторически первым из перечисленных выше документов. В нём описывается синтаксис использования аннотаций в функциях Python. Важным является то, что аннотации не имеют никакого семантического значения для интерпретатора Python и предназначены только для анализа сторонними приложениями. Аннотировать можно аргументы функции и возвращаемое ей значение.

PEP 484 – Type Hints. В нём представлены рекомендации по использованию аннотаций типов. Аннотация типов упрощает статический анализ кода, рефакторинг, контроль типов в рантайме и кодогенерацию, использующую информацию о типах.

В PEP 526 – Syntax for Variable Annotations приводится описание синтаксиса для аннотации типов переменных (базируется на PEP 484), использующего языковые конструкции, встроенные в Python.

PEP 563 — Postponed Evaluation of Annotations. Данный PEP вступил в силу с выходом Python 3.7. У подхода работы с аннотация до этого PEP’а был ряд проблем связанных с тем, что определение типов переменных (в функциях, классах и т.п.) происходит во время импорта модуля, и может сложиться такая ситуация, что тип переменной объявлен, но информации об этом типе ещё нет, в таком случае тип указывают в виде строки – в кавычках.

4. Как осуществляется аннотирование параметров и возвращаемых значений функций?

Аннотация для аргумента определяется через двоеточие после его имени. Аннотация, определяющая тип возвращаемого функцией значения, указывается после ее имени с использованием символов `->`.

5. Как выполнить доступ к аннотациям функций?

Доступ к использованным в функции аннотациям можно получить через атрибут `__annotations__`, в котором аннотации представлены в виде словаря, где ключами являются атрибуты, а значениями – аннотации. Возвращаемое функцией значение хранится в записи с ключом `return`.

6. Как осуществляется аннотирование переменных в языке Python?

```
var = value # type: annotation  
var: annotation; var = value  
var: annotation = value
```

7. Для чего нужна отложенная аннотация в языке Python?

Отложенная аннотация в языке Python нужна для ускорения выполнения программы. Это достигается за счёт того, что переменные определяются до получения информации об их типах, а проверка типов не тратится при загрузке модулей, а делается перед работой с переменными.