

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образо-
вательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

«Аннотация типов»

**Отчет по лабораторной работе № 4.5
по дисциплине «Объектно-ориентированное программирование»**

Выполнил студент группы ИВТ-б-о-21-1

Богадунов Василий Игоревич.

« » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Цель: 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.

Ход работы:

1. Создание репозитория для выполнения работы.

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 *

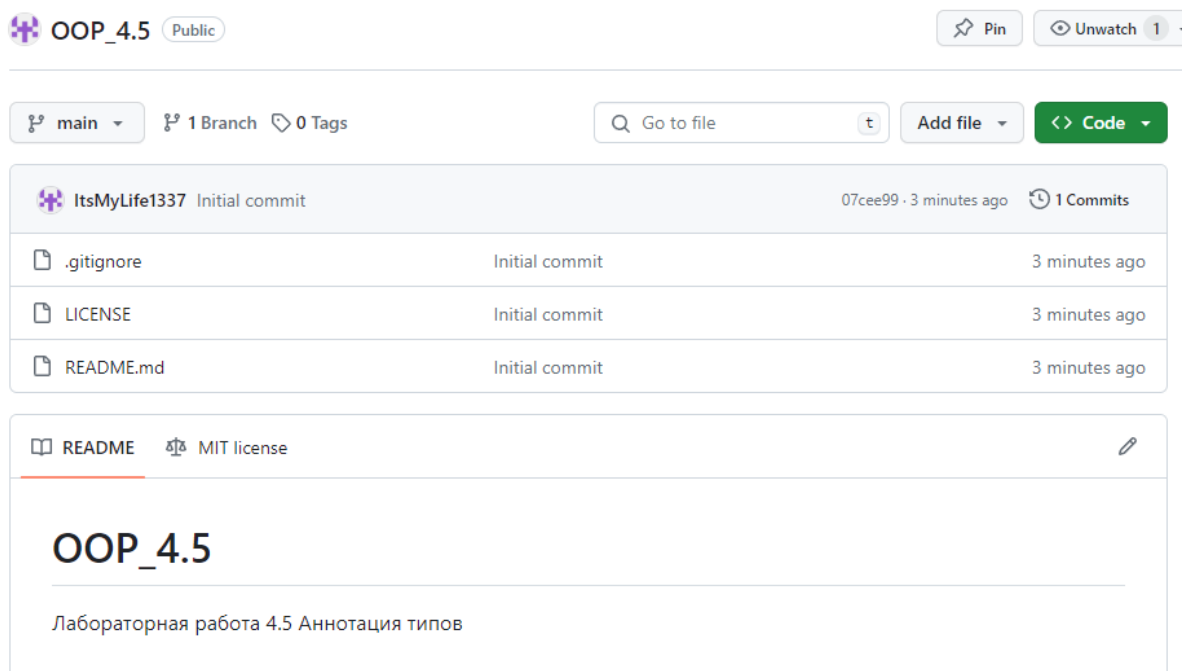
 ItsMyLife1337 /

Great repository names are short and memorable. Need inspiration? How about [vigilant-palm-tree](#) ?

Description (optional)

— 2.1.1

Рисунок №1 – Созданный репозиторий



The screenshot shows a GitHub repository page for 'OOP_4.5' by user 'ItsMyLife1337'. The repository is public and has 1 commit. The commit message is 'Initial commit' and it was made 3 minutes ago. The repository contains three files: '.gitignore', 'LICENSE', and 'README.md', all committed 3 minutes ago. The README file is selected, showing the title 'OOP_4.5' and the description 'Лабораторная работа 4.5 Аннотация типов'.

OOP_4.5 Public

main 1 Branch 0 Tags

Go to file Add file Code

ItsMyLife1337 Initial commit 07cee99 · 3 minutes ago 1 Commits

.gitignore	Initial commit	3 minutes ago
LICENSE	Initial commit	3 minutes ago
README.md	Initial commit	3 minutes ago

README MIT license

OOP_4.5

Лабораторная работа 4.5 Аннотация типов

Рисунок №2 – Созданный репозиторий

```
c:\Users\Admin\Desktop\git>git clone https://github.com/ItsMyLife1337/OOP_4.5.git
Cloning into 'OOP_4.5'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 3 – Клонирование репозитория

```
c:\Users\Admin\Desktop\git\OOP_4.5>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Admin/Desktop/git/OOP_4.5/.git/hooks]

c:\Users\Admin\Desktop\git\OOP_4.5>
```

Рисунок 4 – Организация репозитория в соответствии с моделью ветвления
git flow

Индивидуальное задание: Выполнить индивидуальное задание 2 лабораторной работы 2.19, добавив аннотации типов. Выполнить проверку программы с помощью утилиты `myru`.

```
+ C:\Users\Admin\Desktop\git\OOP_4.5\Task
+ .idea
+ .gitignore
+ misc.xml
+ workspace.xml
+ ind.py
```

Рисунок 5 – Результат работы программы

Вывод: в результате выполнения лабораторной работы были приобретены навыки 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 муру для анализа Python кода.

Ответы на контрольные вопросы:

1. Для чего нужны аннотации типов в языке Python?

Аннотации типов в языке Python представляют собой способ указать ожидаемый тип данных для аргументов функций, возвращаемых значений функций и переменных. Вот несколько причин, по которым аннотации типов могут быть полезны:

1. Документация: Аннотации типов могут служить документацией для кода, помогая другим разработчикам понять ожидаемые типы данных в функциях и методах.

2. Поддержка инструментов статического анализа: Аннотации типов могут использоваться инструментами статического анализа кода, такими как Муру, Pyre или Pyright, чтобы проверять соответствие типов данных во время компиляции или анализа кода.

3. Улучшение читаемости: Аннотации типов могут помочь улучшить читаемость кода, особенно в случае сложных или больших проектов, где явное указание типов данных может помочь понять назначение переменных и результатов функций.

4. Интеграция с IDE: Некоторые интегрированные среды разработки (IDE), такие как PyCharm, могут использовать аннотации типов для предоставления подсказок о типах данных и автоматической проверки соответствия типов.

2. Как осуществляется контроль типов в языке Python?

В языке Python контроль типов данных может осуществляться несколькими способами:

1. Аннотации типов:

Как уже упоминалось, в Python можно использовать аннотации типов

для указания ожидаемых типов данных для аргументов функций, возвращаемых значений функций и переменных. Это позволяет документировать ожидаемые типы данных и использовать инструменты статического анализа кода для проверки соответствия типов.

2. Использование инструментов статического анализа:

Существуют сторонние инструменты, такие как `Myru`, `Pyre` и `Pyright`, которые могут использоваться для статической проверки соответствия типов данных в

Python-коде. Эти инструменты могут обнаруживать потенциальные ошибки типов данных и предоставлять рекомендации по улучшению кода.

3. Вручную проверять типы данных:

В Python можно вручную выполнять проверку типов данных с помощью условных операторов и функций, таких как `isinstance()`. Например, можно написать условие для проверки типа данных перед выполнением определенной операции.

4. Использование аннотаций типов в комбинации с декораторами:

В Python можно использовать декораторы, такие как `@overload` из модуля `functools`, для реализации перегрузки функций с разными типами аргументов.

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

Предложения по усовершенствованию работы с аннотациями типов в Python включают расширение поддержки аннотаций типов, улучшение интеграции с инструментами статического анализа, улучшение документации и рекомендаций, а также разработку стандартной библиотеки типов. Эти изменения могут сделать работу с аннотациями типов более мощной и удобной для разработчиков.

4. Как осуществляется аннотирование параметров и возвращаемых значений функций?

В Python аннотирование параметров и возвращаемых значений функций

осуществляется с использованием двоеточия и указания типа данных после имени параметра или перед знаком ">" для возвращаемого значения.

Например:

```
def greet(name: str) -> str:  
    return "Hello, " + name
```

В этом примере name: str указывает, что параметр name должен быть строкой, а -> str указывает, что функция возвращает строку.

5. Как выполнить доступ к аннотациям функций?

В Python можно получить доступ к аннотациям функций с помощью специального атрибута annotations. Этот атрибут содержит словарь, в котором ключами являются имена параметров или "return" (для возвращаемого значения), а значениями - указанные типы данных. Пример:

```
def greet(name: str) -> str:  
    return "Hello, " + name  
print(greet.__annotations__)
```

Этот код выведет на экран словарь с аннотациями функции greet:

```
{'name': , 'return': }
```

Таким образом, вы можете получить доступ к аннотациям функции и использовать их в своем коде, например, для проверки типов данных или для документирования функций.

6. Как осуществляется аннотирование переменных в языке Python?

В Python переменные можно аннотировать с использованием синтаксиса аннотаций типов. Это позволяет указать ожидаемый тип данных для переменной, хотя интерпретатор Python не выполняет никакой проверки типов во время выполнения.

7. Для чего нужна отложенная аннотация в языке Python?

Отложенная аннотация в Python (Delayed Evaluation Annotation) позволяет создавать аннотации типов, используя строковые литералы вместо ссылок на фактические классы. Это может быть полезно в случаях, когда требуется аннотировать типы данных, которые еще не определены или недоступны

в момент написания аннотации.

Отложенные аннотации могут быть полезны при работе с циклическими зависимостями между классами или модулями, при использовании динамически загружаемых модулей или при аннотации типов в коде, который будет выполняться на разных версиях Python.