

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО–КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
дисциплины
«Объектно–ориентированное программирование»
Вариант 13

Выполнил:
Рябинин Егор Алексеевич
3 курс, группа ИВТ–б–о–23–2,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Проверил:
Доцент департамента цифровых,
робототехнических систем и
электроники института перспективной
инженерии
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2025 г

Тема: Аннотации типов.

Цель: приобретение навыков по работе с аннотациями типов при написании программ с помощью языка программирования Python версии 3.13.3.

Порядок выполнения работы:

Ссылка на репозиторий:

https://github.com/bohemiaaaaa/Lab4_Object-oriented-programming

Задание 1. Проверка корректности типов при возврате None.

Создайте функцию:

```
def printer(value: str) -> None:  
    print(value)
```

После вызова функции проверьте, что тип возвращаемого значения равен NoneType. Выведите результат проверки.

Листинг программы:

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
  
def printer(value: str) -> None:  
    print(value)
```

Листинг программы:

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
  
from printer_function import printer  
  
def main():  
    result = printer("Hello, World!")  
    print(f"Тип возвращаемого значения: {type(result)}")  
    print(f"Значение возвращаемого значения: {result}")  
    print(f"Является ли NoneType: {result is None}")  
  
if __name__ == "__main__":  
    main()
```

```
PS C:\Users\4isto\OOP_lab4> python tasks/task1.py
Hello, World!
Тип возвращаемого значения: <class 'NoneType'>
Значение возвращаемого значения: None
Является ли NoneType: True
```

Рисунок 1 – Результат работы программы

Задание 2. Универсальная пара значений.

Создайте класс, хранящий два значения одного и того же типа, и метод, возвращающий их в виде кортежа. Тип элементов должен быть параметризован.

Листинг программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from typing import Generic, Tuple, TypeVar

T = TypeVar("T")

class UniversalPair(Generic[T]):
    def __init__(self, first: T, second: T) -> None:
        if type(first) is not type(second):
            raise TypeError(
                f"Типы first и second должны совпадать: {type(first)} != {type(second)}"
            )
        self.first = first
        self.second = second

    def get_pair(self) -> Tuple[T, T]:
        return (self.first, self.second)
```

Листинг программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from universal_pair_package.universal_pair import UniversalPair

def main():
    int_pair = UniversalPair(10, 3)
    print(f"Пара целых чисел: {int_pair.get_pair()}")

    str_pair = UniversalPair("hello", "world")
    print(f"Пара строк: {str_pair.get_pair()}")

    float_pair = UniversalPair(3.14, 2.71)
    print(f"Пара дробных чисел: {float_pair.get_pair()}")

if __name__ == "__main__":
    main()
```

```
PS C:\Users\4isto\OOP_lab4> python tasks/task2.py
Пара целых чисел: (10, 3)
Пара строк: ('hello', 'world')
Пара дробных чисел: (3.14, 2.71)
```

Рисунок 2 – Результат работы программы

```
PS C:\Users\4isto\OOP_lab4> pytest
===== test session starts
platform win32 -- Python 3.11.0, pytest-8.3.5, pluggy-1.6.0 -- C:\Program Files\Python311\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\4isto\OOP_lab4
configfile: pyproject.toml
testpaths: tests
plugins: anyio-4.9.0
collected 7 items

tests/test_task1.py::test_printer_return_type PASSED
tests/test_task1.py::test_printer_annotations PASSED
tests/test_task1.py::test_printer_output PASSED
tests/test_task2.py::test_universal_pair_int PASSED
tests/test_task2.py::test_universal_pair_str PASSED
tests/test_task2.py::test_universal_pair_same_type PASSED
tests/test_task2.py::test_universal_pair_list PASSED

===== 7 passed in 0.10s
```

Рисунок 3 – Результат работы pytest для написанных программ

Контрольные вопросы:

1. Что такое аннотация типов и какую роль она играет в языке Python?

Аннотации типов в Python – это синтаксис для указания ожидаемых типов данных для переменных, параметров функций и возвращаемых значений. Они играют роль подсказок для разработчиков и инструментов статического анализа, но не влияют на выполнение кода интерпретатором.

2. Почему интерпретатор Python не использует аннотации во время выполнения программы?

Интерпретатор Python игнорирует аннотации типов во время выполнения для сохранения динамической природы языка и обратной совместимости. Аннотации доступны через annotations, но их проверка не выполняется автоматически.

3. В чем разница между Union и Optional?

Union означает, что переменная может иметь любой из указанных типов. Optional[Type] эквивалентен Union[Type, None] – указывает, что значение может быть указанного типа или None.

4. Какой модуль Python используется для аннотации коллекций и обобщенных типов?

Модуль `typing` используется для аннотации коллекций и обобщенных типов, например `List[int]`, `Dict[str, int]`.

5. Как включить отложенную обработку аннотаций и зачем она нужна?

Отложенную обработку аннотаций включают через `from future import annotations` или установкой `future.annotations = True`. Это нужно для решения проблем циклических импортов и аннотаций, которые еще не определены.

6. Что делает инструмент муру и какие ошибки он может обнаружить?

Муру – это статический анализатор типов для Python. Он может обнаружить ошибки несоответствия типов, отсутствующие атрибуты, неправильное количество аргументов, несовместимые возвращаемые значения и другие ошибки типизации.

7. Чем Literal отличается от Enum при ограничении возможных значений?

`Literal` ограничивает конкретными значениями (например, `Literal["red", "blue"]`), а `Enum` определяет перечисление с именованными константами. `Literal` проще для фиксированных значений, `Enum` лучше для группировки связанных констант.

8. Приведите пример использования TypeVar и объясните его смысл.

Пример: `T = TypeVar('T')` `def first_item(items: List[T]) -> T: return items[0]`. `TypeVar` позволяет создавать обобщенные функции, работающие с разными типами, сохраняя информацию о типе.

9. В каких случаях следует использовать тип Any и почему его избыточное использование нежелательно?

Any следует использовать когда тип неизвестен или должен быть динамическим. Его избыточное использование нежелательно, так как это ослабляет проверку типов и снижает преимущества статической типизации.

10. Какая разница между `→ None` и `→ NoReturn`?

`→ None` указывает, что функция возвращает `None` (явно или неявно). –
> `NoReturn` означает, что функция никогда не возвращает управление (например, всегда вызывает исключение или бесконечный цикл).

Вывод: в ходе лабораторной работы были приобретены навыки по работе с аннотациями типов при написании программ с помощью языка программирования Python версии 3.13.3.