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.

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

Рисунок 1 - Реализация функции printer с аннотацией возвращаемого типа None

```python
from printer_function import printer


def main():
    # Проверка типа возвращаемого значения
    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()
```

Рисунок 2 - Код для проверки типа возвращаемого значения функции printer

```
(oop4) 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 3 – 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.

```python
from typing import Generic, Tuple, TypeVar

T = TypeVar("T")


class UniversalPair(Generic[T]):
    def __init__(self, first: T, second: T) -> None:
        self.first = first
        self.second = second

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

Рисунок 4 - Реализация универсального класса UniversalPair с параметризацией типа

```python
from universal_pair import UniversalPair


def main():
    # С целыми числами
    int_pair = UniversalPair[int](10, 20)
    print(f"Пара целых чисел: {int_pair.get_pair()}")

    # Со строками
    str_pair = UniversalPair[str]("hello", "world")
    print(f"Пара строк: {str_pair.get_pair()}")

    # С числами с плавающей точкой
    float_pair = UniversalPair[float](3.14, 2.71)
    print(f"Пара дробных чисел: {float_pair.get_pair()}")


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

Рисунок 5 - Демонстрация работы класса с разными типами данных

```
(oop4) PS C:\Users\4isto\OOP_lab4> python tasks/task2.py
Пара целых чисел: (10, 20)
Пара строк: ('hello', 'world')
Пара дробных чисел: (3.14, 2.71)
```

Рисунок 6 – Результат работы программы

```python
import pytest
from printer_function import printer


def test_printer_return_type():
    result = printer("test")
    assert result is None


def test_printer_annotations():
    annotations = printer.__annotations__
    assert annotations["value"] == str
    assert annotations["return"] is None


def test_printer_output(capsys: pytest.CaptureFixture[str]):
    printer("test message")
    captured = capsys.readouterr()
    assert captured.out == "test message\n"
```

Рисунок 7 - Набор тестов для проверки функции printer и её аннотаций типов

```python
from universal_pair import UniversalPair


def test_universal_pair_int():
    pair = UniversalPair[int](10, 20)
    result = pair.get_pair()
    assert result == (10, 20)
    assert isinstance(result[0], int)
    assert isinstance(result[1], int)


def test_universal_pair_str():
    pair = UniversalPair[str]("hello", "world")
    result = pair.get_pair()
    assert result == ("hello", "world")
    assert isinstance(result[0], str)
    assert isinstance(result[1], str)


def test_universal_pair_same_type():
    pair = UniversalPair[float](3.14, 2.71)
    result = pair.get_pair()
    assert all(isinstance(x, float) for x in result)


def test_universal_pair_list():
    pair = UniversalPair[list]([1, 2], [3, 4])
    result = pair.get_pair()
    assert result == ([1, 2], [3, 4])
```

Рисунок 8 - Набор тестов для проверки универсального класса UniversalPair с различными типами данных

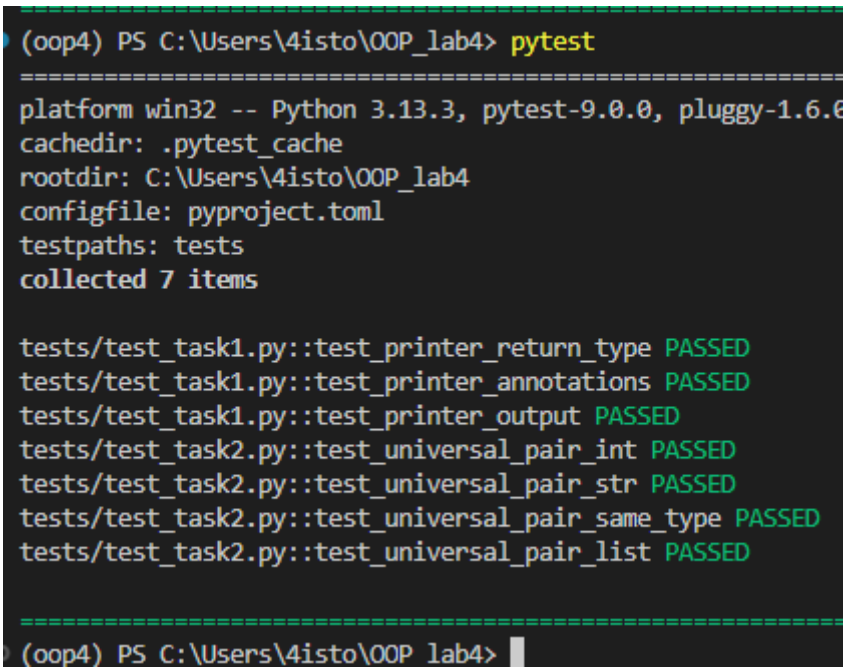

Рисунок 9 – Результат работы тестов

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 mypy 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.