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 № 3

Выполнил:
Левашев Тимур Рашидович
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 Р.А

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема работы: “Аннотации типов”.

Цель работы: приобретение навыков по работе с аннотациями типов при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

Ссылка на git репозиторий: https://github.com/mazy99/oop_pract_4

1. Базовый пример аннотаций типов.

```python
# no type annotations
def total_price_1(price, tax):
    return price + (price * tax)


# with type annotations
def total_price_2(price: float, tax: float) -> float:
    return price + (price * tax)
```

```
{'price': <class 'float'>, 'tax': <class 'float'>, 'return': <class 'float'>}
```

Рисунок 1 – Ожидаемые типы данных для функции total_price_2

2. Пример аннотации типов коллекций.

```python
from typing import Dict, List, Set, Tuple

numbers: List[int] = [1, 2, 3, 4, 5]

gradesL: Dict[str, float] = {"math": 4.5, "physics": 5.0}

record: Tuple[str, int] = ("John", 25)

tags: Set[str] = {"python", "typing", "annotations"}
```

3. Пример аннотации типов с использованием Union.

```python
from typing import Union
```

```python
def normalize(value: Union[int, float, str]) -> float:
    if isinstance(value, str):
        value = value.replace(",", ".")
        return float(value)
    return float(value)
```

```
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4> uv run examples\example_union.py
{'value': int | float | str, 'return': <class 'float'>}
```

Рисунок 2 – Ожидаемые типы данных для функции normalize

4. Пример аннотации типов с использованием Optional.

```python
from typing import Optional


def find_user_id(name: str) -> Optional[int]:
    user = {"Alice": 1, "Bob": 2, "Charlie": 3}
    return user.get(name)
```

```
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4> uv run examples\example_optional.py
{'name': <class 'str'>, 'return': int | None}
```

Рисунок 3 – Ожидаемые типы данных для find_user_id

5. Пример доступа к аннотации в рантайме.

```python

def add(a: int, b: int) -> int:
    return a + b


print(add.__annotations__)
```

```
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4> uv run examples\example_run_time_annotation.py
{'a': <class 'int'>, 'b': <class 'int'>, 'return': <class 'int'>}
```

Рисунок 4 – Ожидаемые типы данных для add

6. Использование аннотаций типов в классах.

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


class Student:
```

```
    name: str
    grades: list[int]

    def __init__(self, name: str, grades: list[int]) -> None:
        self._name = name
        self._grades = grades

    def average(self) -> float:
        return sum(self._grades) / len(self._grades)
```

```
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4> uv run examples\example_class_annotations.py
Ожидаемые типы данных для класса Student: {'name': <class 'str'>, 'grades': list[int]}
Ожидаемые типы данных для метода __init__: {'name': <class 'str'>, 'grades': list[int], 'return': None}
Ожидаемые типы данных для метода average: {'return': <class 'float'>}
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4>
```

Рисунок 5 – Ожидаемые типы данных для класса Student

7. Ограничение типов значений с помощью Literal.

```
from typing import Literal


def set_status(status: Literal["new", "in_progress", "completed"]) -> None:
    print(f"Текущий статус: {status}")
```

Пример корректного вызова функций с ограничением значений:

```
set_status("new")
set_status("in_progress")
set_status("completed")
```

```
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4> mypy examples\example_literal.py
Success: no issues found in 1 source file
```

Рисунок 6 – Результат корректного вызова функции

Пример некорректного вызова функции:

```
set_status("archived")
```

```
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4> mypy examples\example_literal.py
examples\example_literal.py:11: error: Argument 1 to "set_status" has incompatible type "Literal['archived']"; expected "Literal['new', 'in_progress', 'completed']"  [arg-type]
Found 1 error in 1 file (checked 1 source file)
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4>
```

Рисунок 7 – Ошибка при запуске кода

8. Использование универсального типа данных.

```
#!/usr/bin/env python3
```

```python
# -*- coding: utf-8 -*-


from typing import Generic, TypeVar

T = TypeVar("T")


class Box(Generic[T]):
    def __init__(self, value: T) -> None:
        self.value = value

    def get(self) -> T:
        return self.value
```

```
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4> uv run examples\universal_type_class.py
Типы данных класса Box: {}
Типы данных метода __init__:{'value': ~T, 'return': None}
Типы данных метода get: {'return': ~T}
```

Рисунок 8 – Типы данных класса Box

9. Выполнение индивидуального задания 1.

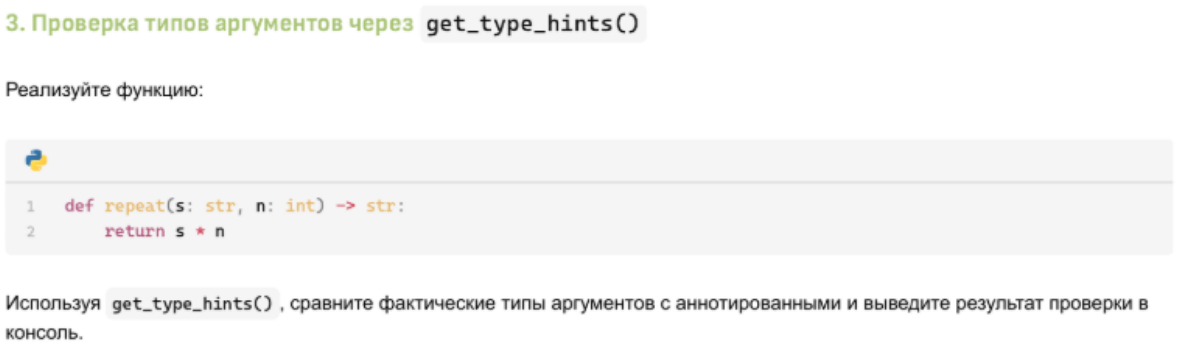

Рисунок 9 – Индивидуальное задание 1

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


def repeat(s: str, n: int) -> str:
    return s * n

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


from typing import get_type_hints

from repeat import repeat
```

```python
def main():
    s_val = "Hello "
    n_val = 3
    rep = repeat(s_val, n_val)
    func_hints = get_type_hints(repeat)
    actual_s = type(s_val)
    actual_n = type(n_val)
    actual_return = type(rep)
    print(f"\nРезультат вызова функции repeat: {rep}")
    print(f"\nТип аргумента 's': ожидается {func_hints['s']}, получен
{actual_s}")
    print(f"\nТип аргумента 'n': ожидается {func_hints['n']}, получен
{actual_n}")
    print(
        f"\nТип возвращаемого значения: ожидается {func_hints['return']}\
, получен {actual_return}\n"
    )


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

```
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4> uv run tasks\task_1.py

Результат вызова функции repeat: Hello Hello Hello

Тип аргумента 's': ожидается <class 'str'>, получен <class 'str'>

Тип аргумента 'n': ожидается <class 'int'>, получен <class 'int'>

Тип возвращаемого значения: ожидается <class 'str'>, получен <class 'str'>

(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4>
```

Рисунок 10 – Пример вызова программы

10. Выполнение индивидуального задания 2.

3. Обобщённая функция поиска минимума

Напишите универсальную функцию, которая возвращает минимальный элемент из списка любого типа, поддерживающего операцию сравнения.
Реализуйте аннотацию типов с `TypeVar`.

Рисунок 11– Индивидуальное задание 2

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


from typing import List, Protocol, TypeVar
```

```python
class AnyComparableClass(Protocol):
    def __lt__(self, other: AnyComparableClass) -> bool: ...


T = TypeVar("T", bound=AnyComparableClass)


def find_min(items: List[T]) -> T:
    if items:
        return min(items)
    raise ValueError("Список не должен быть пустым")


def main():
    items = ["apple", "banana", "cherry", "date"]
    min_item = find_min(items)
    print(f"Минимальный элемент в списке: {min_item}")


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

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

from find_min_package import find_min


def main():
    items = ["apple", "banana", "cherry", "date"]
    min_item = find_min(items)
    print(f"Минимальный элемент в списке: {min_item}")


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

```
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4> uv run tasks\task_2.py
Минимальный элемент в списке: apple
Минимальное число в списке: 8
Минимальное число с плавающей точкой в списке: 1.41
Минимальный элемент в смешанном списке: 2
(oop-pract-4) PS C:\учеба\ООП\4 лаба\oop_pract_4>
```

Рисунок 12 – Пример вызова программы

Вывод: в ходе выполнения работы были получены навыки по использованию аннотаций типов при написании программ с помощью языка программирования Python версии 3.x.