

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное образовательное  
учреждение высшего образования**  
**«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**  
**Кафедра инфокоммуникаций**  
**Институт цифрового развития**

**ОТЧЁТ**

по лабораторной работе №4.1

Дисциплина: «Объектно-ориентированное программирование»

Тема: «Элементы объектно-ориентированного программирования в  
языке Python»

Вариант 5

Выполнила: студентка 3 курса,  
группы ИВТ-б-о-21-1  
Диченко Дина Алексеевна

Ставрополь 2023

**Цель работы:** приобретение навыков по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.

### Практическая часть:

1. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

**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 \***

DinaDichenko / Lab-4.1

✔ Lab-4.1 is available.

Great repository names are short and memorable. Need inspiration? How about [super-duper-lamp](#) ?

**Description** (optional)

**Visibility**

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

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

**Initialize this repository with:**

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

[Create repository](#)

Рисунок 1. Создание репозитория

2. Выполнила клонирование созданного репозитория.

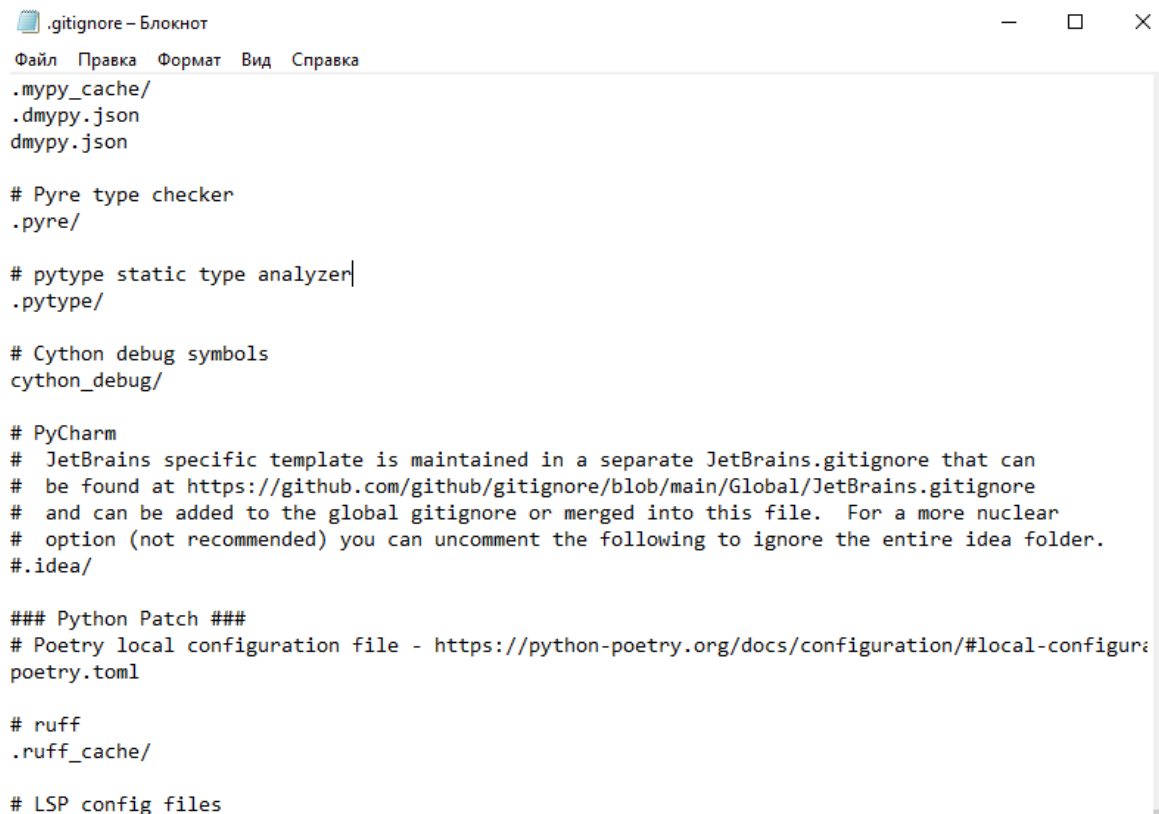
```

: \Users\super\OneDrive\Рабочий стол\DI\ВУЗЬ>git clone https://github.com/DinaDichenko/Lab-4.1.git
Cloning into 'Lab-4.1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
: \Users\super\OneDrive\Рабочий стол\DI\ВУЗЬ>_

```

Рисунок 2. Клонирование репозитория

3. Дополнила файл .gitignore необходимыми правилами для работы с IDE PyCharm.



```

.gitignore – Блокнот
Файл  Правка  Формат  Вид  Справка

.pyru_cache/
.dmpy.json
dmpy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file.  For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
#.idea/

### Python Patch ###
# Poetry local configuration file - https://python-poetry.org/docs/configuration/#local-configuration
poetry.toml

# ruff
.ruff_cache/

# LSP config files

```

Рисунок 3. Изменение файла gitignore

4. Организовала свой репозиторий в соответствии с моделью ветвления git-flow.

```

C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗб\Lab-4.1>git commit -m "gitignore"
[main fb6c4bd] gitignore
1 file changed, 129 insertions(+)

C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗб\Lab-4.1>git branch develop

C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗб\Lab-4.1>git checkout develop
Switched to branch 'develop'

C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗб\Lab-4.1>git flow init

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

Which branch should be used for integration of the "next release"?
- develop
Branch name for "next release" development: [develop] develop

How to name your supporting branch prefixes?
Feature branches? [feature/] fra
Bugfix branches? [bugfix/] bug
Release branches? [release/] rel
Hotfix branches? [hotfix/] fix
Support branches? [support/] sup
Version tag prefix? [] ver
Hooks and filters directory? [C:/Users/super/OneDrive/Рабочий стол/DI/ВУЗб/Lab-4.1/.git/hooks] hook

C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗб\Lab-4.1>

```

Рисунок 4. Организация репозитория в соответствии с git flow

#### 4. Проработала примеры лабораторной работы.

```

3/4
Введите обыкновенную дробь: 5/6
5/6
19/12
1/12
5/8
10/9
PS C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗб\Lab-4.1\prog>

```

Рисунок 5. Выполнение примера

#### 6. Выполнила индивидуальные задания.

##### Задание 1

Парой называется класс с двумя полями, которые обычно имеют имена `first` и `second`. Требуется реализовать тип данных с помощью такого класса. Во всех заданиях обязательно должны присутствовать: метод инициализации `__init__`; метод должен контролировать значения аргументов на корректность; ввод с клавиатуры `read`; вывод на экран `display`.

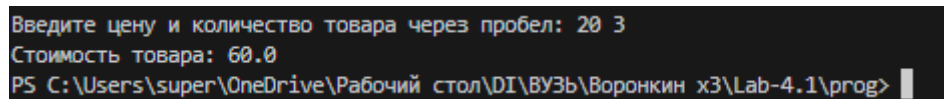
Реализовать внешнюю функцию с именем `make_тип()`, где `тип` — тип реализуемой структуры. Функция должна получать в качестве аргументов значения для полей структуры и возвращать структуру требуемого типа. При

передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ == '__main__':` добавить код, демонстрирующий возможности разработанного класса.

(вариант 5)

Поле `first` — дробное положительное число, цена товара; поле `second` — целое положительное число, количество единиц товара. Реализовать метод `cost()` — вычисление стоимости товара.



```
Введите цену и количество товара через пробел: 20 3
Стоимость товара: 60.0
PS C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗб\Воронкин х3\Lab-4.1\prog>
```

Рисунок 6. Выполнение индивидуального задания 1

## Задание 2

Составить программу с использованием классов и объектов для решения задачи. Во всех заданиях, помимо указанных в задании операций, обязательно должны быть реализованы следующие методы:

метод инициализации `__init__` ;

ввод с клавиатуры `read` ;

вывод на экран `display` .

Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ == '__main__':` добавить код, демонстрирующий возможности разработанного класса.

(вариант 5)

Создать класс `Angle` для работы с углами на плоскости, задаваемыми величиной в градусах и минутах. Обязательно должны быть реализованы:

перевод в радианы, приведение к диапазону 0-360, увеличение и уменьшение угла на заданную величину, получение синуса, сравнение углов.

```
Введите угол(градусы и минуты вводятся через пробел): 30 0
Введите угол для сравнения: 50 9
30.0 меньше 50.15
В радианах: 0.5235987755982988
В диапазоне 0-360: 30.0
Синус: 0.49999999999999994
Изменение на определенный угол: 60.0
Сравнение с другим углом: 30.0 меньше 50.15
```

Рисунок 7. Выполнение индивидуального задания 2

### Контрольные вопросы:

1. Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова `class` и имени класса:

```
# class syntax
class MyClass:
    var = ... # некоторая переменная

    def do_smt(self):
        # какой-то метод
```

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса являются общими для всех объектов класса, а атрибуты экземпляра специфическими для каждого экземпляра. Более того, атрибуты класса определяются внутри класса, но вне каких-либо методов, а атрибуты экземпляра обычно определяются в методах, чаще всего в `__init__`.

3. Каково назначение методов класса?

Методы определяют функциональность объектов, принадлежащих конкретному классу.

4. Для чего предназначен метод `__init__()` класса?

Чтобы настроить начальное состояние экземпляра, используется метод `__init__`.

Метод `__init__` является конструктором. Конструкторы - это концепция объектно-ориентированного программирования. Класс может иметь один и только один конструктор. Если `__init__` определен внутри класса, он автоматически вызывается при создании нового экземпляра класса.

Метод `__init__` указывает, какие атрибуты будут у экземпляров нашего класса.

#### 5. Каково назначение `self` ?

Аргумент `self` представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам.

Важно использовать параметр `self` внутри метода, если мы хотим сохранить значения экземпляра для последующего использования.

#### 6. Как добавить атрибуты в класс?

Метод `__init__` указывает, какие атрибуты будут у экземпляров нашего класса.

Добавить атрибут можно следующим образом:

Объект.атрибут = значение

#### 7. Как осуществляется управление доступом к методам и атрибутам в языке Python?

Хорошим тоном считается, что для чтения/изменения какого-то атрибута должны использоваться специальные методы, которые называются `getter/setter`, их можно реализовать, но ничего не мешает изменить атрибут напрямую. При этом есть соглашение, что метод или атрибут, который начинается с нижнего подчеркивания, является скрытым, и снаружи класса трогать его не нужно (хотя сделать это можно).

#### 8. Каково назначение функции `isinstance` ?

Встроенная функция `isinstance(obj, Cls)` , используемая при реализации методов арифметических операций и операций отношения, позволяет узнать что некоторый объект `obj` является либо экземпляром класса `Cls` либо экземпляром одного из потомков класса `Cls`.

**Вывод:** в результате выполнения работы были приобретены навыки по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.