

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

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**  
дисциплины «Основы программной инженерии»

Выполнил:  
Мелтонян Одиссей  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Проверил: Воронкин Р. А.

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

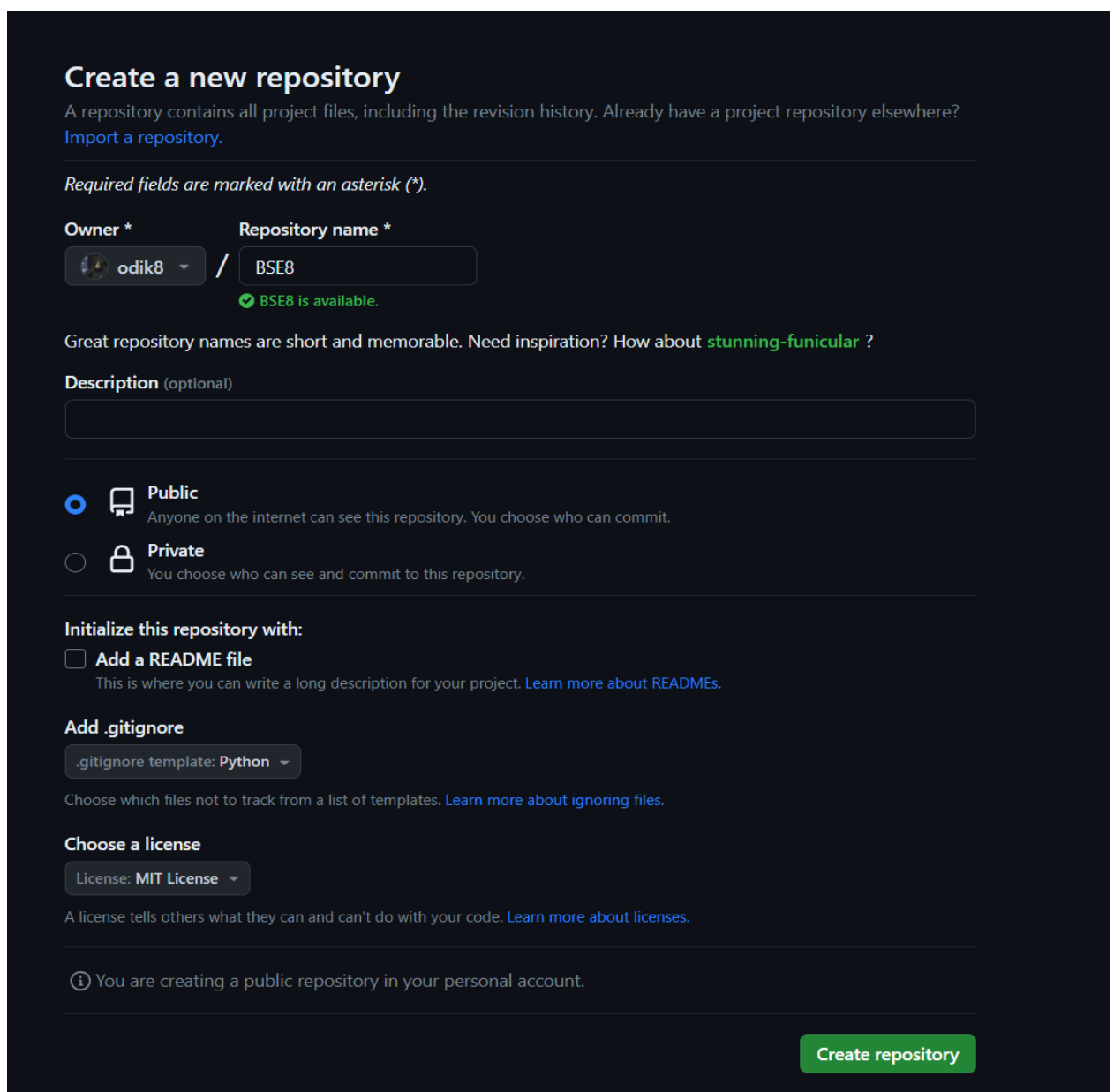
Ставрополь, 2023 г

Тема: Работа с кортежами в языке Python

Цель работы: приобретение навыков по работе с кортежами при написании программ с помощью языка программирования Python.

Ход работы:

1. Изучил теоретический материал работы.
2. Создал 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 \***

 odik8 / BSE8

✔ BSE8 is available.

Great repository names are short and memorable. Need inspiration? How about [stunning-funicular](#) ?

**Description** (optional)

☒ **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**


.gitignore template: **Python**

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

**Choose a license**

License: **MIT License**

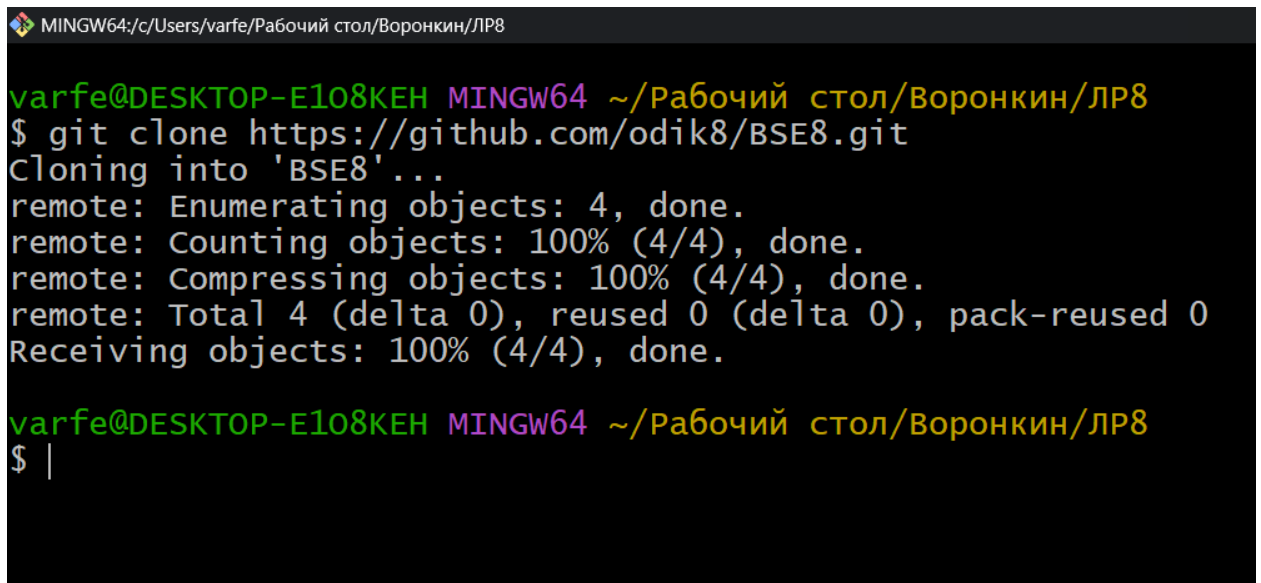
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

[Create repository](#)

Рисунок 1 – Создание репозитория

3. Выполнил клонирование созданного репозитория.

A screenshot of a terminal window with a dark background. The title bar at the top shows the path 'MINGW64:/c:/Users/varfe/Рабочий стол/Воронкин/ЛР8'. The terminal text shows a user 'varfe@DESKTOP-E108KEH' in a 'MINGW64' environment at the directory '~/Рабочий стол/Воронкин/ЛР8'. The user enters the command '\$ git clone https://github.com/odik8/BSE8.git'. The output shows the cloning process: 'Cloning into 'BSE8'...', 'remote: Enumerating objects: 4, done.', 'remote: Counting objects: 100% (4/4), done.', 'remote: Compressing objects: 100% (4/4), done.', 'remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0', and 'Receiving objects: 100% (4/4), done.'. The prompt '\$ |' is visible at the bottom.

```
MINGW64:/c:/Users/varfe/Рабочий стол/Воронкин/ЛР8
varfe@DESKTOP-E108KEH MINGW64 ~/Рабочий стол/Воронкин/ЛР8
$ git clone https://github.com/odik8/BSE8.git
Cloning into 'BSE8'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
varfe@DESKTOP-E108KEH MINGW64 ~/Рабочий стол/Воронкин/ЛР8
$ |
```

Рисунок 2 – Клонирование репозитория

4. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
.gitignore x
1 # Byte-compiled / optimized / DLL files
2 __pycache__/
3 *.py[cod]
4 *$py.class
5
6 # C extensions
7 *.so
8
9 # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28
```

Рисунок 3 – файл .gitignore

5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow.

```
varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/Воронкин/ЛР8/bse8 (main)
$ 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/varfe/Рабочий стол/Воронкин/ЛР8/bse8/.git/hooks]

varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/Воронкин/ЛР8/bse8 (develop)
$ |
```

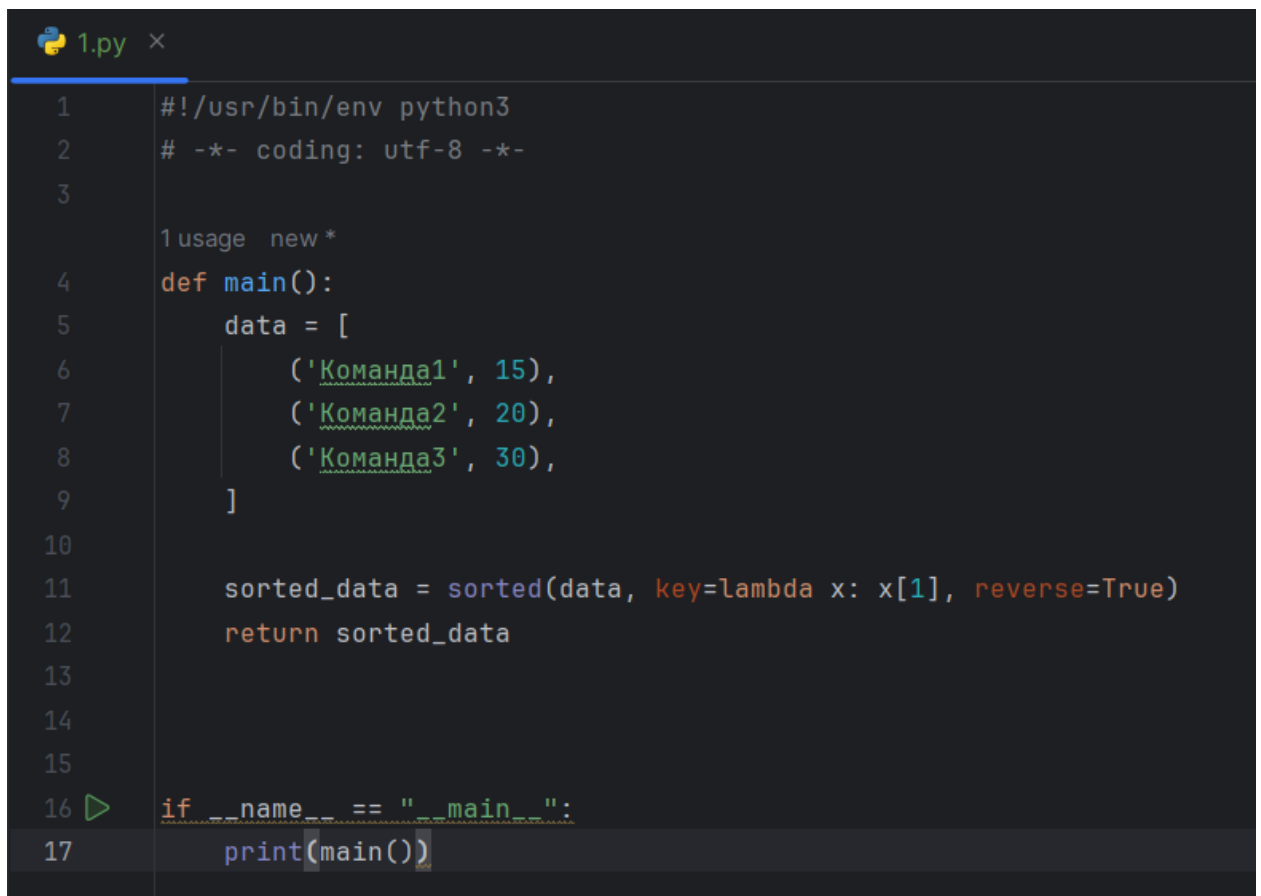
Рисунок 4 – Инициализация git-flow

Примеры лабораторной работы:

Вариант 11:

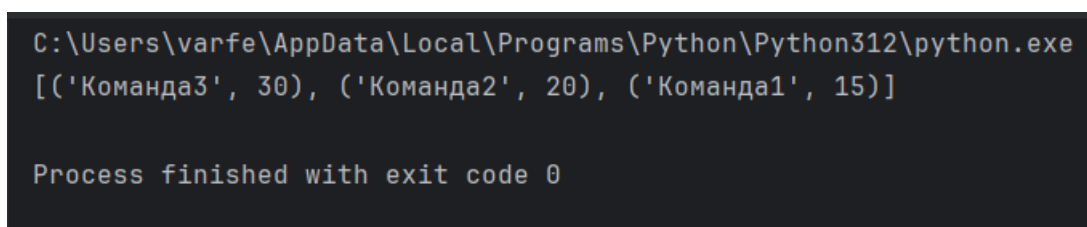
11\_1. Имеются данные о сумме очков, набранных в чемпионате каждой из футбольных команд. Определить, перечислены ли команды в списке в соответствии с занятыми ими местами в чемпионате.

Код решения:

A screenshot of a Python IDE window titled '1.py'. The code defines a 'main' function that takes a list of tuples representing teams and their points. The list is sorted in descending order of points using a lambda function. The sorted list is then returned. The main function is called when the script is executed directly.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  1 usage new *
5  def main():
6      data = [
7          ('Команда1', 15),
8          ('Команда2', 20),
9          ('Команда3', 30),
10     ]
11
12     sorted_data = sorted(data, key=lambda x: x[1], reverse=True)
13     return sorted_data
14
15
16 if __name__ == "__main__":
17     print(main())
```

Рисунок 5 – Код решения задачи

A screenshot of a Windows command prompt showing the execution of the Python script. The command path is 'C:\Users\varfe\AppData\Local\Programs\Python\Python312\python.exe'. The output shows the sorted list of teams and points: [('Команда3', 30), ('Команда2', 20), ('Команда1', 15)]. The process finished with exit code 0.

```
C:\Users\varfe\AppData\Local\Programs\Python\Python312\python.exe
[('Команда3', 30), ('Команда2', 20), ('Команда1', 15)]

Process finished with exit code 0
```

Рисунок 6 – Результат выполнения программы

Вопросы для защиты работы:

1. **Что такое списки в языке Python?** – Список в языке Python представляет собой упорядоченную коллекцию элементов, которая может содержать объекты различных типов. Списки являются изменяемыми, что означает возможность добавления, удаления и изменения элементов.

2. **Каково назначение кортежей в языке Python?** – это неизменяемая упорядоченная коллекция элементов. Кортежи часто используются для представления неизменяемых последовательностей данных, например, для хранения координат, даты и других значений, которые не должны изменяться.

3. **Как осуществляется создание кортежей?** – Кортеж можно создать, перечислив его элементы в круглых скобках и разделив их запятыми, например: `my_tuple = (1, 2, 'three')`.

4. **Как осуществляется доступ к элементам кортежа?** – Доступ к элементам кортежа осуществляется по индексу, например: `element = my_tuple[0]`.

5. **Зачем нужна распаковка (деструктуризация) кортежа?** – Распаковка кортежа позволяет присваивать значения элементов кортежа переменным одновременно. Например, `a, b, c = my_tuple` присвоит значения элементов кортежа переменным `a`, `b` и `c`.

6. **Какую роль играют кортежи в множественном присваивании?** – Кортежи используются для удобного множественного присваивания, где каждой переменной присваивается соответствующий элемент кортежа.

7. **Как выбрать элементы кортежа с помощью среза?** – Элементы кортежа можно выбирать с использованием срезов, например: `subset_tuple = my_tuple[1:3]` выберет элементы с индексами 1 и 2.

8. **Как выполняется конкатенация и повторение кортежей?** – Кортежи можно конкатенировать с использованием оператора `+`, их можно

также повторять с использованием оператора **\***, например: **new\_tuple = tuple1 + tuple2**.

9. **Как выполняется обход элементов кортежа?** – Элементы кортежа можно обойти с использованием цикла **for**, например:

```
for item in my_tuple:
```

```
    print(item)
```

10. **Как проверить принадлежность элемента кортежу?** – Принадлежность элемента кортежу можно проверить с использованием оператора **in**, например: **result = 2 in my\_tuple**.

11. **Какие методы работы с кортежами Вам известны?** – Некоторые методы работы с кортежами включают **count()** (для подсчета вхождений элемента) и **index()** (для получения индекса первого вхождения элемента).

12. **Допустимо ли использование функций агрегации таких как len(), sum() и т. д. при работе с кортежами?** – Да, функции агрегации, такие как **len()** и **sum()**, могут использоваться с кортежами для определения их длины или суммирования элементов, так как эти операции не изменяют кортеж.

13. **Как создать кортеж с помощью спискового включения.** – Кортеж можно создать с помощью спискового включения и преобразования списка в кортеж, например: **new\_tuple = tuple(x for x in my\_list)**.