

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
Отчет по лабораторной работе № 2.5
«Работа с кортежами в языке Python»
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » ноября 2022 г.

Подпись студента _____

Работа защищена

« » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2022

Цель работы:

Приобретение навыков по работе с кортежами при написании программ с помощью языка программирования Python версии 3.x.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: <https://github.com/afk552/lab2.5>

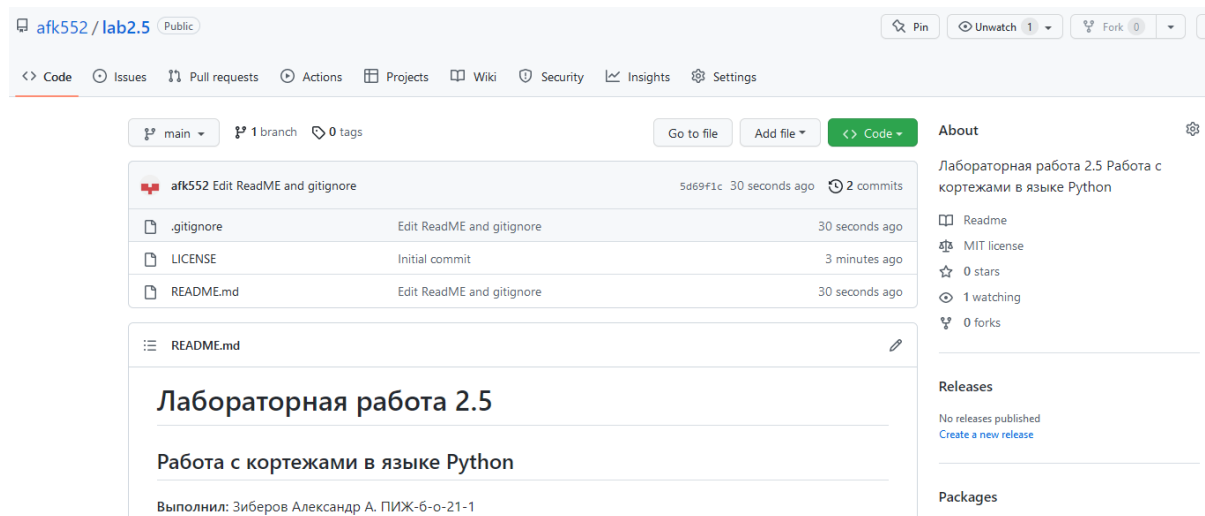


Рисунок 1 – Репозиторий GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

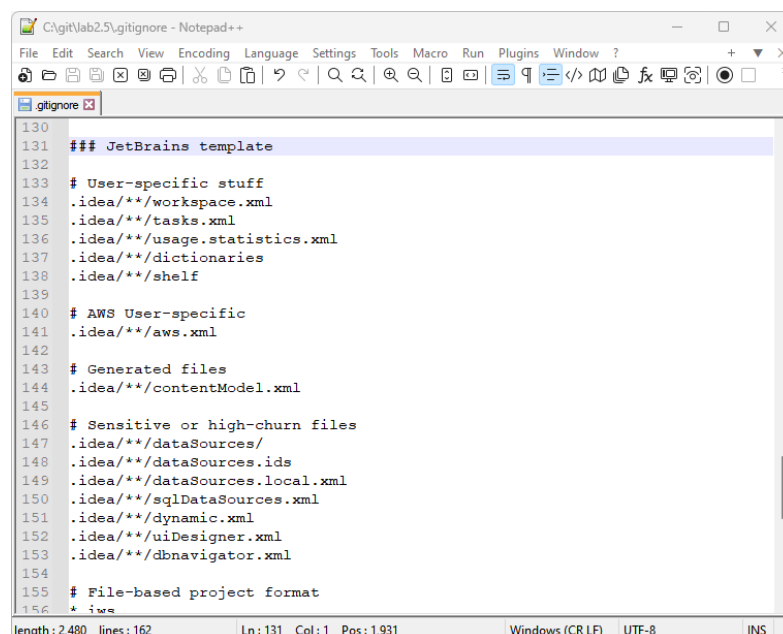


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствии с моделью ветвления git-flow, рисунок 3.



```
C:\Windows\System32\cmd.exe

C:\git\lab2.5>git branch
* main

C:\git\lab2.5>git checkout -b develop
Switched to a new branch 'develop'

C:\git\lab2.5>git branch
* develop
  main

C:\git\lab2.5>git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/afk552/lab2.5/pull/new/develop
remote:
To https://github.com/afk552/lab2.5
 * [new branch]      develop -> develop

C:\git\lab2.5>
```

Рисунок 3 – Окно командной строки

Создайте проект PyCharm в папке репозитория, рисунок 4.

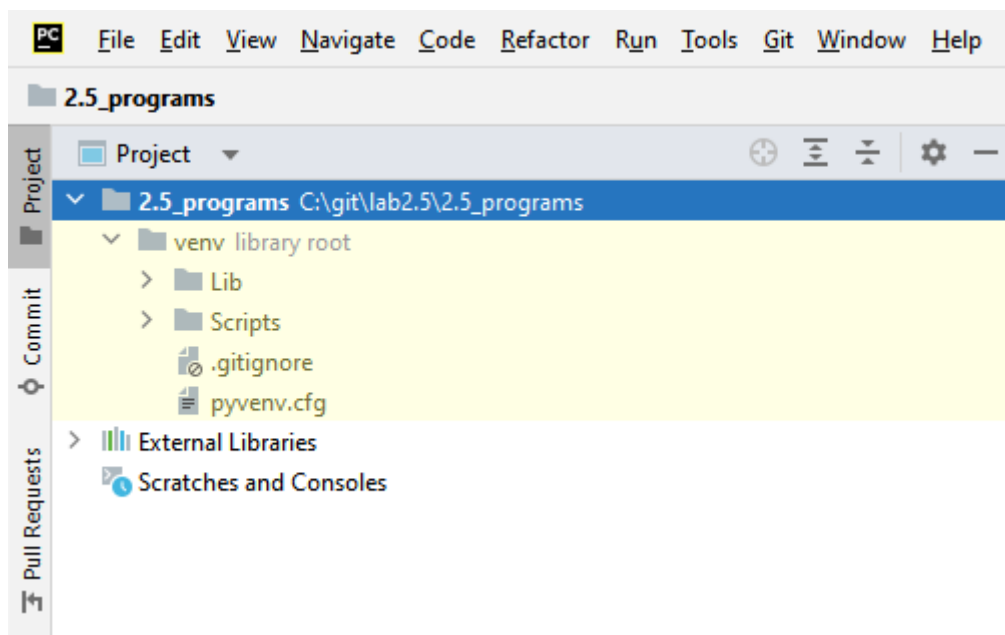
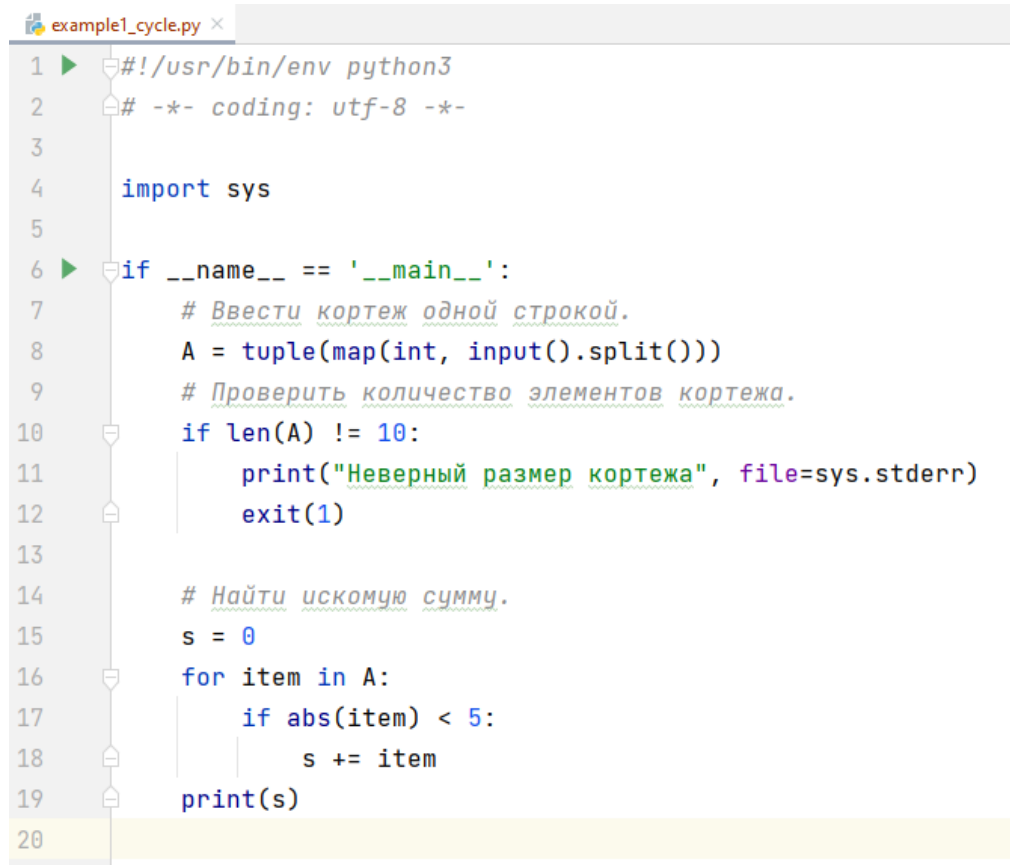


Рисунок 4 – Окно проекта в PyCharm

Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры. Рисунки 5-9.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == '__main__':
7     # Ввести кортеж одной строкой.
8     A = tuple(map(int, input().split()))
9     # Проверить количество элементов кортежа.
10    if len(A) != 10:
11        print("Неверный размер кортежа", file=sys.stderr)
12        exit(1)
13
14    # Найти искомую сумму.
15    s = 0
16    for item in A:
17        if abs(item) < 5:
18            s += item
19    print(s)
20
```

Рисунок 5 – Код программы из примера 1 (через циклы)



```
Run: example1_cycle x
C:\git\lab2.5\2.5_programs\venv\Scripts\python.exe C:/git/lab2.5/2.5_programs/example1_cycle.py
10 20
Неверный размер кортежа
Process finished with exit code 1

Run: example1_cycle x
C:\git\lab2.5\2.5_programs\venv\Scripts\python.exe C:/git/lab2.5/2.5_programs/example1_cycle.py
1 2 3 4 5 1 2 3 4 5
20
Process finished with exit code 0
```

Рисунок 6 – Результат выполнения программы из примера 1 (через циклы)

```

example1_lc.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import sys
5
6  ▶  if __name__ == '__main__':
7      # Ввести список одной строкой
8      A = list(map(int, input().split()))
9      # Проверить количество элементов списка
10     if len(A) != 10:
11         print("Неверный размер списка", file=sys.stderr)
12         exit(1)
13
14     # Найти искомую сумму
15     s = sum([a for a in A if abs(a) < 5])
16     print(s)
17

```

Рисунок 7 – Код программы из примера 1 (через списковое включение)

```

Run: example1_lc x
▶  C:\git\lab2.5\2.5_programs\venv\Scripts\python.exe C:/git/lab2.5/2.5_programs/example1_lc.py
20 10
Неверный размер кортежа
Process finished with exit code 1

Run: example1_lc x
▶  C:\git\lab2.5\2.5_programs\venv\Scripts\python.exe C:/git/lab2.5/2.5_programs/example1_lc.py
1 2 3 4 5 1 2 3 4 5
20
Process finished with exit code 0

```

Рисунок 8 – Результаты выполнения программы из примера 1 (через списковое включение)

```

C:\Windows\System32\cmd.exe

C:\git\lab2.5>git add .

C:\git\lab2.5>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   2.5_programs/example1_cycle.py
    new file:   2.5_programs/example1_lc.py

C:\git\lab2.5>git commit -m "Add examples"
[develop 89365f1] Add examples
 2 files changed, 35 insertions(+)
 create mode 100644 2.5_programs/example1_cycle.py
 create mode 100644 2.5_programs/example1_lc.py

C:\git\lab2.5>

```

Рисунок 9 – Окно командной строки

Выполните индивидуальные задания, согласно своего варианта.

Вариант 12

Задание 1. В начале кортежа записано несколько равных между собой элементов. Определить количество таких элементов и вывести все элементы, следующие за последним из них. Рассмотреть возможность того, что весь массив заполнен одинаковыми элементами. Условный оператор не использовать. Рисунки 10, 11, 12.

```
indiv_task_1.py x
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  ▶  if __name__ == '__main__':
7      option = input("Как заполнить кортеж? 'Вручную' или 'Автоматически': ")
8      if option.lower() == "вручную":
9          print("Введите значения через пробел: ")
10         A = tuple(map(int, input().split()))
11         if len(A) == 0:
12             print("В кортеже нет значений!", file=sys.stderr)
13             exit(1)
14     else:
15         A = (76, 76, 76, 54, 62, 14, 102, 43)
16     print("Введенные значения: ", *A)
17
18     i = 1
19     pos = 0
20     # Обработка ошибки одинаковых значений (выход за границу)
21     try:
22         # Пока не дошли до конца кортежа и нулевой элемент равен следующим
23         while (i <= len(A)) and (A[0] == A[i]):
24             i += 1
25             pos += 1
26     except IndexError:
27         print("В кортеже все значения одинаковые!", file=sys.stderr)
28         exit(1)
29     print("Количество равных элементов в начале: ", pos + 1)
30     print("Элементы, стоящие после равных элементов: ", *A[pos + 1:])
31
```

Рисунок 10 – Код программы индивидуального задания 1

```
Run: indiv_task_1 x
C:\git\lab2.5\2.5_programs\venv\Scripts\python.exe C:/git/lab2.5/2.5_programs/indiv_task_1.py
Введите, как заполнить кортеж: "Вручную" или "Автоматически"
Автоматически
Введенные значения: 76 76 76 54 62 14 102 43
Количество равных элементов в начале: 3
Элементы, стоящие после равных элементов: 54 62 14 102 43
Process finished with exit code 0

Run: indiv_task_1 x
C:\git\lab2.5\2.5_programs\venv\Scripts\python.exe C:/git/lab2.5/2.5_programs/indiv_task_1.py
Введите, как заполнить кортеж: "Вручную" или "Автоматически"
Вручную
76 76 76 54 62 14 102 43
Введенные значения: 76 76 76 54 62 14 102 43
Количество равных элементов в начале: 3
Элементы, стоящие после равных элементов: 54 62 14 102 43
Process finished with exit code 0

Run: indiv_task_1 x
C:\git\lab2.5\2.5_programs\venv\Scripts\python.exe C:/git/lab2.5/2.5_programs/indiv_task_1.py
Введите, как заполнить кортеж: "Вручную" или "Автоматически"
Вручную
76 76 76 76 76 76
Введенные значения: 76 76 76 76 76 76
В кортеже все значения одинаковые!
Process finished with exit code 1
```

Рисунок 11 – Результаты выполнения программы индивидуального задания 1

```
C:\Windows\System32\cmd.exe

C:\git\lab2.5>git add .

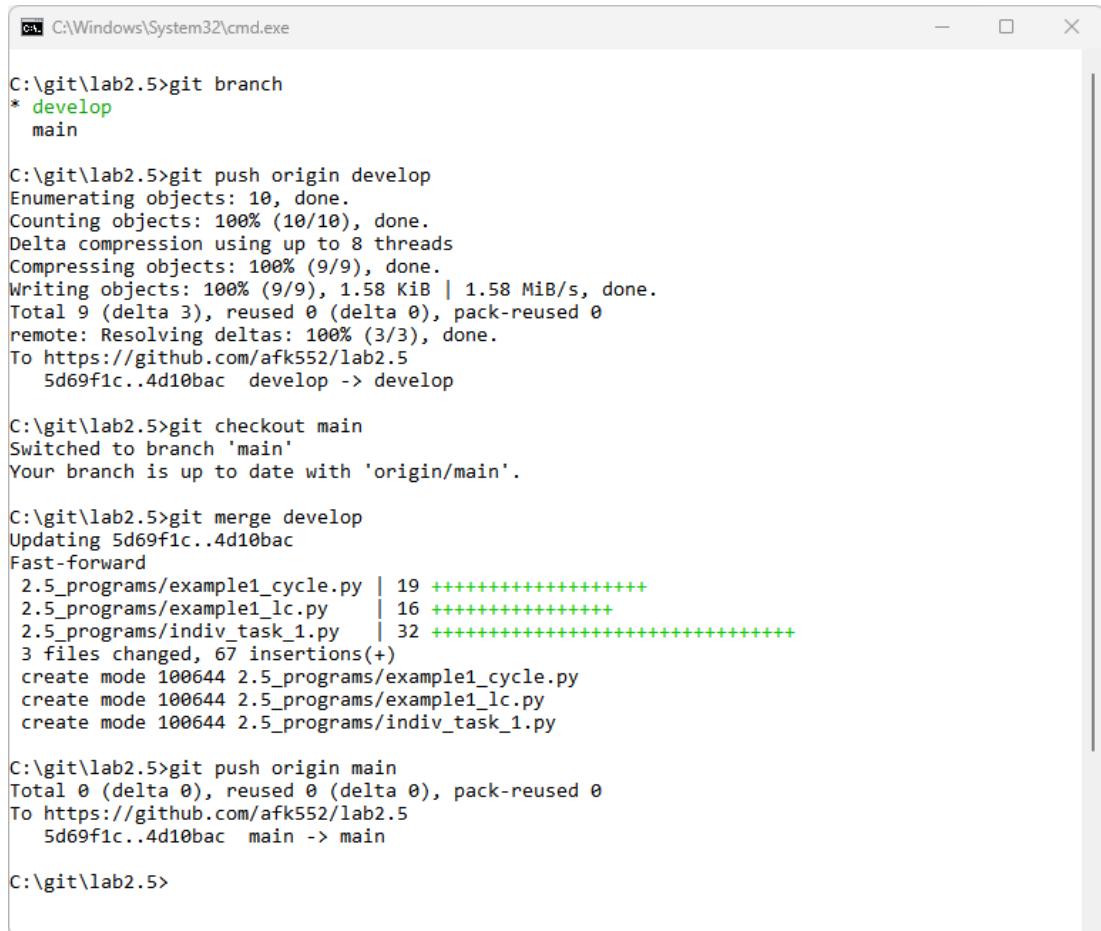
C:\git\lab2.5>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   2.5_programs/indiv_task_1.py

C:\git\lab2.5>git commit -m "Add individual tasks"
[develop 4d10bac] Add individual tasks
1 file changed, 32 insertions(+)
create mode 100644 2.5_programs/indiv_task_1.py

C:\git\lab2.5>
```

Рисунок 12 – Окно командной строки

Выполните слияние ветки для разработки с веткой main / master.
Отправьте сделанные изменения на сервер GitHub. Рисунки 13, 14.



```
C:\Windows\System32\cmd.exe

C:\git\lab2.5>git branch
* develop
  main

C:\git\lab2.5>git push origin develop
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.58 KiB | 1.58 MiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/afk552/lab2.5
   5d69f1c..4d10bac  develop -> develop

C:\git\lab2.5>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git\lab2.5>git merge develop
Updating 5d69f1c..4d10bac
Fast-forward
 2.5_programs/example1_cycle.py | 19 ++++++
 2.5_programs/example1_lc.py    | 16 ++++++
 2.5_programs/indiv_task_1.py   | 32 ++++++
 3 files changed, 67 insertions(+)
 create mode 100644 2.5_programs/example1_cycle.py
 create mode 100644 2.5_programs/example1_lc.py
 create mode 100644 2.5_programs/indiv_task_1.py

C:\git\lab2.5>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/afk552/lab2.5
   5d69f1c..4d10bac  main -> main

C:\git\lab2.5>
```

Рисунок 13 – Окно командной строки

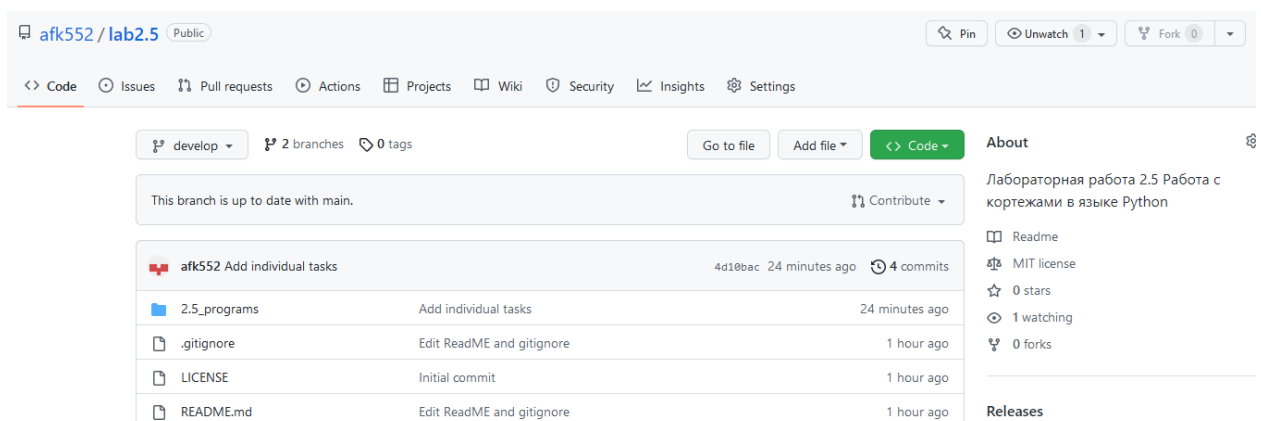


Рисунок 14 – Удаленный репозиторий на GitHub

Вывод: В результате выполнения работы были изучены кортежи в языке программирования Python 3, работа с ними и написание программ с их использованием.

Контрольные вопросы:

1. Что такое списки (вероятно, имеется в виду кортежи) в языке Python?

Кортеж (tuple) – это неизменяемая структура данных, которая очень похожа на список.

2. Каково назначение кортежей в языке Python?

Кортежи помогают обезопасить данные от случайного изменения, например, если нам нужно работать с его элементами, но нельзя модифицировать. Также, кортежи в памяти занимают меньший объем по сравнению со списками. Прирост производительности и меньше время работы.

3. Как осуществляется создание кортежей?

```
a = ()
```

```
b = tuple()
```

```
a = (1, 2, 3, 4, 5)
```

```
a = tuple([1, 2, 3, 4])
```

4. Как осуществляется создание кортежей?

Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса.

5. Зачем нужна распаковка (деструктуризация) кортежа?

Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса.

6. Какую роль играют кортежи в множественном присваивании?

С помощью множественного присваивания, можно реализовать функцию обмена двумя значениями: $(a, b) = (b, a)$

7. Как выбрать элементы кортежа с помощью среза?

$T2 = T1[i:j]$

$T2$ – новый кортеж, который получается из кортежа $T1$;

$T1$ – исходный кортеж, для которого происходит срез;

8. Как выполняется конкатенация и повторение кортежей?

$T3 = T1 + T2$

$T1$, $T2$ – кортежи, для которых нужно выполнить операцию конкатенации. Операнды $T1$, $T2$ обязательно должны быть кортежами. При выполнении операции конкатенации для кортежей, использовать в качестве операндов любые другие типы (строки, списки) запрещено;

$T3$ – кортеж, который есть результатом.

$T2 = T1 * n$

$T2$ – результирующий кортеж;

$T1$ – исходный кортеж, который нужно повторить n раз;

n – количество повторений кортежа $T1$.

9. Как выполняется обход элементов кортежа?

Элементы кортежа можно последовательно просмотреть с помощью операторов цикла `while` или `for`.

10. Как проверить принадлежность элемента кортежу?

`<элемент> in <название кортежа>`

11. Какие методы работы с кортежами Вам известны?

Метод `index()`. Поиск позиции элемента в кортеже

Метод `count()`. Количество вхождений элемента в кортеж

12. Допустимо ли использование функций агрегации таких как `len()` , `sum()` и т. д. при работе с кортежами?

Да, допустимо.

13. Как создать кортеж с помощью спискового включения?

`tuple(randint(0, 10) for i in range(10))`