

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 №10
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 множествами в языке Python

Цель работы: приобретение навыков по работе с множествами 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 / BSE10

✔ BSE10 is available.

Great repository names are short and memorable. Need inspiration? How about [redesigned-rotary-phone](#) ?

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**

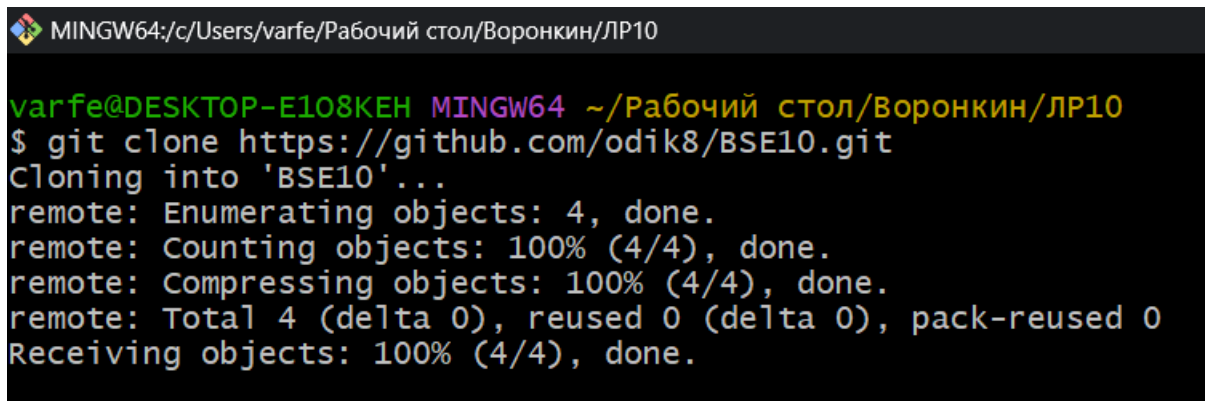
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

You are creating a public repository in your personal account.

[Create repository](#)

Рисунок 1 – Создание нового репозитория.

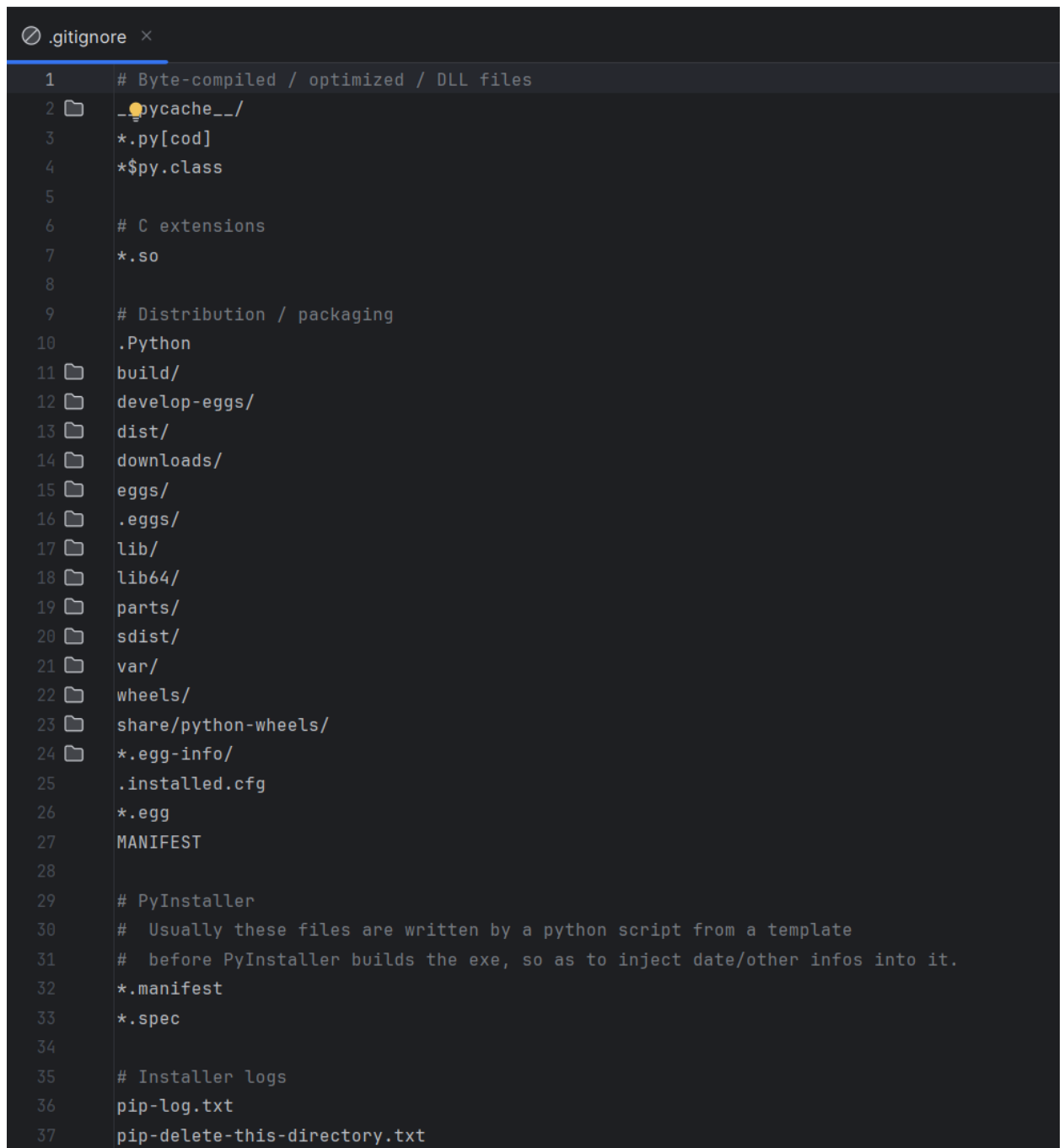
3. Выполнил клонирование созданного репозитория.



```
MINGW64:/c/Users/varfe/Рабочий стол/Воронкин/ЛР10  
  
varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/Воронкин/ЛР10  
$ git clone https://github.com/odik8/BSE10.git  
Cloning into 'BSE10'...  
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.
```

Рисунок 2 – Клонирование репозитория

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

A screenshot of a code editor showing a .gitignore file. The editor has a dark theme and a tab at the top labeled '.gitignore' with a close button. The file content is as follows:

```
1 # Byte-compiled / optimized / DLL files
2 # __pycache__/
3 *.py[co]
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
29 # PyInstaller
30 # Usually these files are written by a python script from a template
31 # before PyInstaller builds the exe, so as to inject date/other infos into it.
32 *.manifest
33 *.spec
34
35 # Installer logs
36 pip-log.txt
37 pip-delete-this-directory.txt
```

Рисунок 3 – файл .gitignore

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

```
varfe@DESKTOP-E108KEH MINGW64 ~/Рабочий стол/Воронкин/ЛР10/bse10 (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]

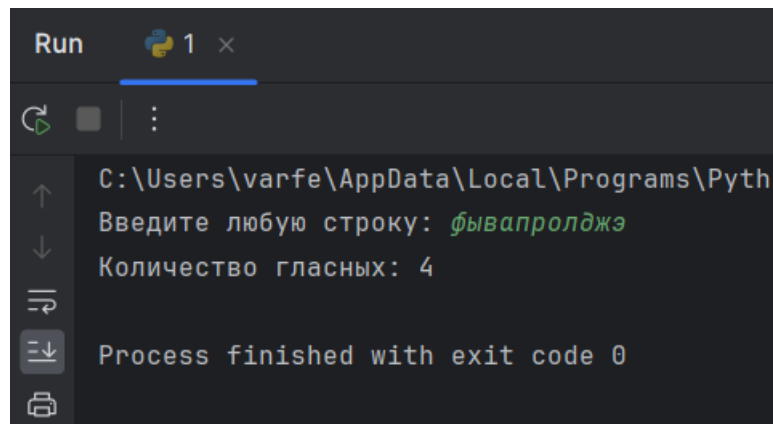
How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/varfe/Рабочий стол/Воронкин/ЛР10/bse10/.git/hooks]
```

Рисунок 4 – Инициализация git-flow

6. Решил задачу: подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  1 usage  1 odyssey
5  def main():
6      user_input = str(input("Введите любую строку: ")).lower()
7      set_of_vowels = set('ауоыэяюёие')
8      vowels_count = sum(1 for char in user_input if char in set_of_vowels)
9
10     return f"Количество гласных: {vowels_count}"
11
12  if __name__ == "__main__":
13      print(main())
14
```

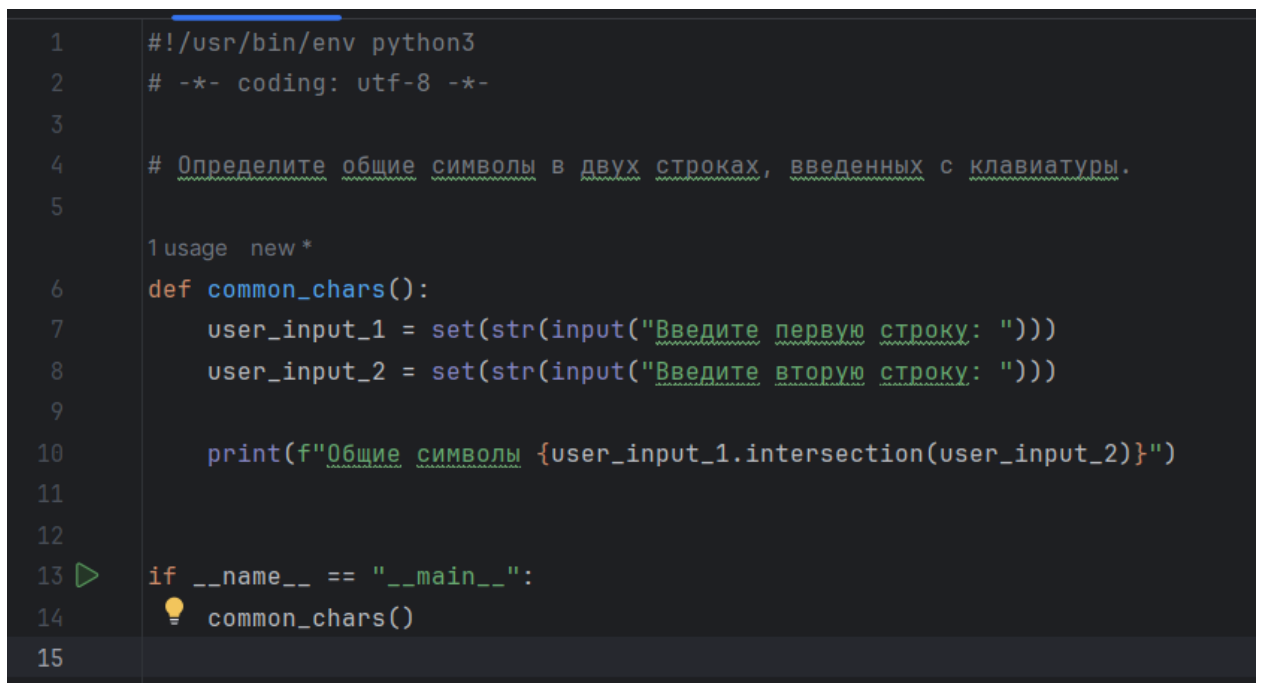
Рисунок 5 – Код первой программы



```
Run 1 x
C:\Users\varfe\AppData\Local\Programs\Python\Python39\python.exe
Введите любую строку: фывапроджэ
Количество гласных: 4
Process finished with exit code 0
```

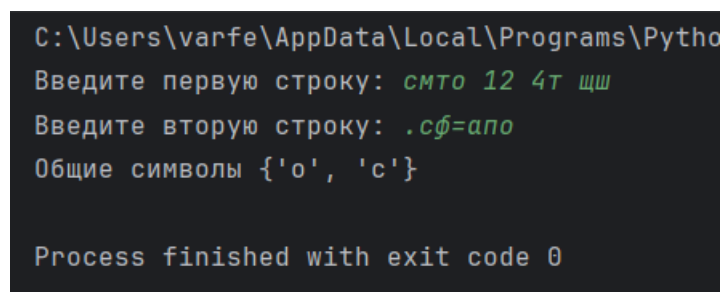
Рисунок 6 – Результат выполнения первой программы

7. Определить общие символы в двух строках, введенных с клавиатуры:



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Определите общие символы в двух строках, введенных с клавиатуры.
5
6  1 usage new *
7  def common_chars():
8      user_input_1 = set(str(input("Введите первую строку: ")))
9      user_input_2 = set(str(input("Введите вторую строку: ")))
10
11      print(f"Общие символы {user_input_1.intersection(user_input_2)}")
12
13  if __name__ == "__main__":
14      common_chars()
15
```

Рисунок 7 – Код решения задачи



```
C:\Users\varfe\AppData\Local\Programs\Python\Python39\python.exe
Введите первую строку: смто 12 4т щш
Введите вторую строку: .сф=апо
Общие символы {'о', 'с'}
Process finished with exit code 0
```

Рисунок 8 – Результат работы программы

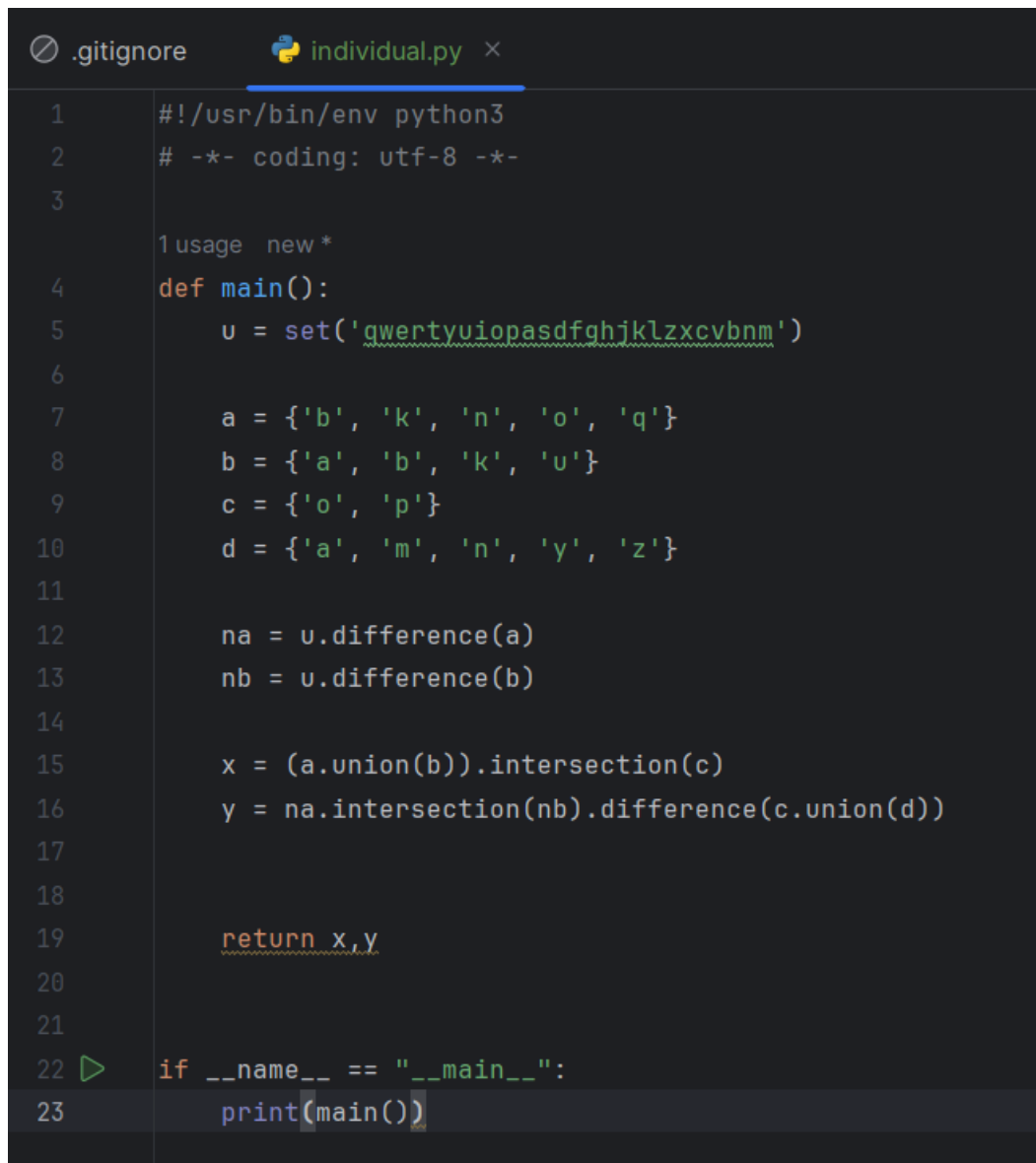
Индивидуальное задание

Вариант 11

Определить результат выполнения операций над множествами. Считать элементы множества строками.

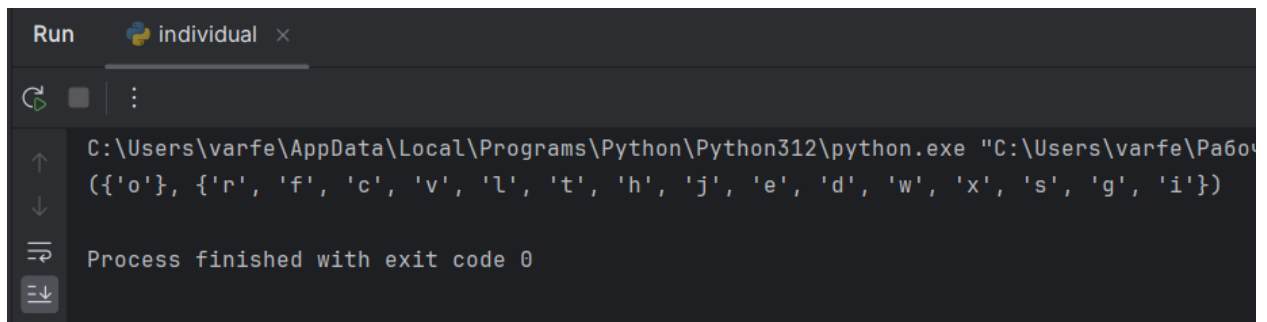
$$X = (A \cup B) \cap C; \quad Y = (\bar{A} \cap \bar{B}) / (C \cup D).$$

$$A = \{b, k, n, o, q\}; \quad B = \{a, b, k, u\}; \quad C = \{o, p\}; \quad D = \{a, m, n, y, z\};$$



```
.gitignore individual.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  usage new *
5  def main():
6      u = set('qwertyuiopasdfghjklzxcvbnm')
7
8      a = {'b', 'k', 'n', 'o', 'q'}
9      b = {'a', 'b', 'k', 'u'}
10     c = {'o', 'p'}
11     d = {'a', 'm', 'n', 'y', 'z'}
12
13     na = u.difference(a)
14     nb = u.difference(b)
15
16     x = (a.union(b)).intersection(c)
17     y = na.intersection(nb).difference(c.union(d))
18
19     return x,y
20
21
22 if __name__ == '__main__':
23     print(main())
```

Рисунок 9 – Код решения индивидуальной задачи



```
Run individual x
C:\Users\varfe\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\varfe\Рабочий стол\... ({'o'}, {'r'}, 'f', 'c', 'v', 'l', 't', 'h', 'j', 'e', 'd', 'w', 'x', 's', 'g', 'i'})
Process finished with exit code 0
```

Рисунок 10 – Результат выполнения программы

Вопросы для защиты работы

1. **Что такое множества в языке Python?** – Python множество (**set**) представляет собой неупорядоченный набор уникальных элементов. Множества используются для выполнения операций над уникальными значениями, такими как объединение, пересечение, разность и другие.

2. **Как осуществляется создание множеств в Python?** – Множества можно создать с использованием фигурных скобок **{}** или с использованием конструктора **set()**. Примеры:

`A = {1, 2, 3} B = set([3, 4, 5])`

3. **Как проверить присутствие/отсутствие элемента в множестве?** – Используйте операторы **in** и **not in**:

`print(2 in A) # Вывод: True`

`print(4 not in A) # Вывод: True`

4. **Как выполнить перебор элементов множества?** – Можно использовать цикл **for**:

`for element in A:`

`print(element)`

5. **Что такое set comprehension?** – Set comprehension — это способ создания множества с использованием синтаксиса, аналогичного генераторам списков, но с использованием фигурных скобок. Пример:


```
squares = {x**2 for x in range(1, 5)}
```

6. **Как выполнить добавление элемента во множество?** – Используйте метод **add()**:

```
A.add(4)
```

7. **Как выполнить удаление одного или всех элементов множества?** – Для удаления одного элемента используйте метод **remove()** или **discard()**. Для удаления всех элементов используйте метод **clear()**.

8. **Как выполняются основные операции над множествами: объединение, пересечение, разность?** – использованием операторов **|** (объединение), **&** (пересечение), и **-** (разность), а также методов **union()**, **intersection()**, и **difference()**.

```
union_set = A | B
```

```
intersection_set = A & B
```

```
difference_set = A - B
```

9. **Как определить, что некоторое множество является надмножеством или подмножеством другого множества?** – Используйте операторы **<=** (подмножество) и **>=** (надмножество) или методы **issubset()** и **issuperset()**.

10. **Каково назначение множеств frozenset?** – **frozenset** - это неизменяемая версия множества. Она может быть использована в качестве ключа в словарях, в отличие от обычных множеств.

11. **Как осуществляется преобразование множеств в строку, список, словарь?** – Используйте функции **str()**, **list()**, и **dict()**:

```
A_str = str(A)
```

```
A_list = list(A)
```

```
A_dict = dict.fromkeys(A, 1) # Преобразование в словарь с ключами из  
множества и значениями 1
```