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 №5
дисциплины «Программирование на Python»

Выполнила:
Федорова Дарья Юрьевна
2 курс, группа ИВТ-б-о-24-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Вариант 25

Ссылка на репозиторий: https://github.com/Vishhh123/Repozzitoriy_5

Тема: Работа с множествами и словарями в языке Python.

Цель: приобретение навыков по работе с множествами и словарями при написании программ с помощью языка программирования Python версии 3.x.

Ход работы:

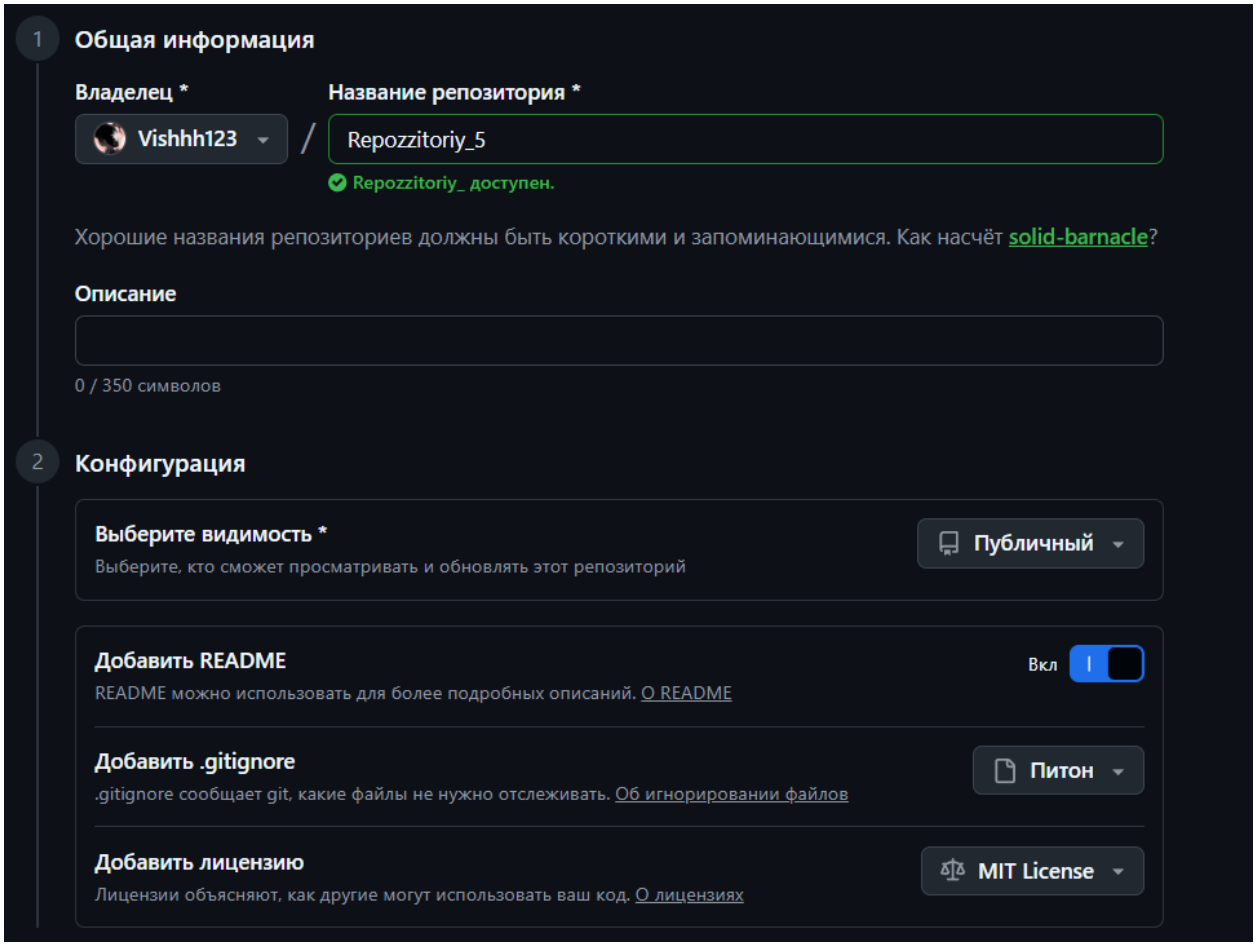


Рис. 1 - создание общедоступного репозитория с лицензией MIT, языком программирования Python.

```
C:\Users\user>git clone https://github.com/Vishhh123/Repozzitoriy_5.git
Cloning into 'Repozzitoriy_5'...
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 (from 0)
Receiving objects: 100% (5/5), done.

C:\Users\user>cd Repozzitoriy_5

C:\Users\user\Repozzitoriy_5>git status
On branch main
Your branch is up to date with 'origin/main'.
```

Рис. 2 - клонирование репозитория на устройство, переход к репозиторию, его проверка.

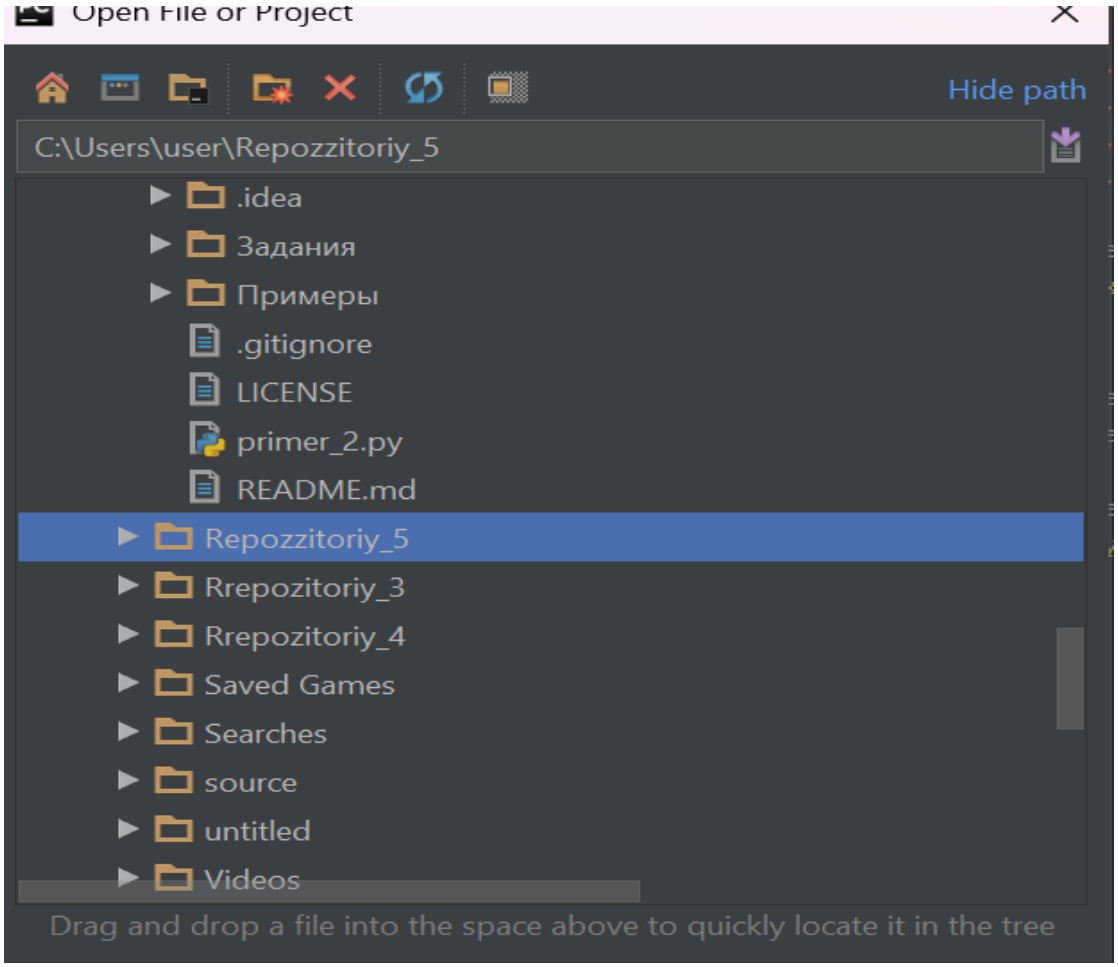


Рис. 3 - создание в PyCharm проекта из существующих файлов (file -> open -> путь к файлу)

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

if __name__ == "__main__":

    u = set("abcdefghijklmnopqrstuvwxyz")

    a = {"b", "c", "h", "o"}
    b = {"d", "f", "g", "o", "y"}
    c = {"d", "e", "j", "k"}
    d = {"a", "b", "f", "g"}

    x = (a.intersection(b)).union(c)
    print(f"x = {x}")


    bn = u.difference(b)
    cn = u.difference(c)

    y = (a.difference(d)).union(cn.difference(bn))
    print(f"y = {y}")
```

```
prim_1
"C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Примеры/prim_1.py
x = {'d', 'o', 'j', 'e', 'k'}
y = {'c', 'h', 'g', 'o', 'y', 'f'}

Process finished with exit code 0
```

Рис. 4 - результат выполнения примера 1.

```
C:\Users\user\Repozzitoriy_5>git add .
warning: in the working copy of '.idea/Repozzitoriy_5.iml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.idea/misc.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.idea/modules.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.idea/workspace.xml', LF will be replaced by CRLF the next time Git touches it

C:\Users\user\Repozzitoriy_5>git commit -m "Код к примеру 1"
[main 9b163e3] Код к примеру 1
 5 files changed, 93 insertions(+)
 create mode 100644 .idea/Repozzitoriy_5.iml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/workspace.xml
 create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213/prim_1.py"
```

Рис. 5 – сохранение, добавление коммита.

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

import sys
from datetime import date

if __name__ == '__main__':
    # Список работников.
    workers = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break

        elif command == 'add':
            # Запросить данные о работнике.
            name = input("Фамилия и инициалы? ")
            post = input("Должность? ")
            year = int(input("Год поступления? "))

            # Создать словарь.
            worker = {
                'name': name,
                'post': post,
                'year': year,
            }
```

Рис. 6 - код к примеру 2.

```
        # Добавить словарь в список.
        workers.append(worker)

        # Отсортировать список в случае необходимости.
        if len(workers) > 1:
            workers.sort(key=lambda item: item.get('name', ''))

    elif command == 'list':
        # Заголовок таблицы.
        line = '+{}+{}+{}+{}+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "No",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)
```

Рис. 7 - код к примеру 2.

```
        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(workers, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
        print(line)

    elif command.startswith('select '):
        # Получить текущую дату.
        today = date.today()

        # Разбить команду на части для выделения номера года.
        parts = command.split(' ', maxsplit=1)
        # Получить требуемый стаж.
        period = int(parts[1])

        # Инициализировать счетчик.
        count = 0
        # Проверить сведения работников из списка.
        for worker in workers:
            if today.year - worker.get('year', today.year) >= period:
                count += 1
                print(
                    '{:>4}: {}'.format(count, worker.get('name', ''))
                )
```

Рис. 8 - код к примеру 2.

```
            # Если счетчик равен 0, то работники не найдены.
            if count == 0:
                print("Работники с заданным стажем не найдены.")

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить работника;")
            print("list - вывести список работников;")
            print("select <стаж> - запросить работников со стажем;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)
```

```
rim_2  prim_2  prim_2  prim_2
"C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Примеры/prim_2.py
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>>
```

Рис. 9 - код к примеру 2.

```
"C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Примеры/prim_2.py
>>> add
Фамилия и инициалы? Миева Л.Ю.
Должность? Маркетолог
Год поступления? 2025
>>> add
Фамилия и инициалы? Сойка Л.М.
Должность? Бухгалтер
Год поступления? 2024
list
+----+-----------------------------+--------------------+--------+
| No |            Ф.И.О.           |      Должность     |  Год   |
+----+-----------------------------+--------------------+--------+
|  1 | Миева Л.Ю.                  | Маркетолог         |   2025 |
|  2 | Сойка Л.М.                  | Бухгалтер          |   2024 |
+----+-----------------------------+--------------------+--------+
>>> exit
```

Рис. 10 - результат выполнения примера 2.

```
C:\Users\user\Repozzitoriy_5>git add .
warning: in the working copy of '.idea/Repozzitoriy_5.iml', LF will be replaced by CRLF the next time Git touches it

C:\Users\user\Repozzitoriy_5>git commit -m "Код к примеру 2"
[main 2c9a902] Код к примеру 2
 3 files changed, 115 insertions(+), 2 deletions(-)
 create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213/prim_2.py"

C:\Users\user\Repozzitoriy_5>git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 24 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.91 KiB | 1.91 MiB/s, done.
Total 7 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Vishhh123/Repozzitoriy_5.git
   9b163e3..2c9a902  main -> main
```

Рис. 11 - сохранение, добавление коммита, отправка на Github.

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

Рис. 12 - задание 1.

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

if __name__ == "__main__":

    alfavit_glasn = set("aeiouyaeёиоуыэюяAEIOUYAEЁИОУЫЭЮЯ")

    text = input("Введите строку: ")

    chars = set(text)

    peresechenie = chars.intersection(alfavit_glasn)

    count = 0

    for char in text:
        if char in alfavit_glasn:
            count += 1

    print(f"Найдено гласных букв: {count}")
    print(f"Уникальные гласные в строке: {peresechenie}")
```

```
danie_1
  "C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Задания/zadanie_1.py
  Введите строку: qwertyйцукенг
  Найдено гласных букв: 4
  Уникальные гласные в строке: {'e', 'y', 'e', 'y'}

  Process finished with exit code 0
```

Рис. 13 - код к заданию 1.

```
C:\Users\user\Repozzitoriy_5>git add .

C:\Users\user\Repozzitoriy_5>git commit -m "код к заданию 1"
[main 9e14921] код к заданию 1
 1 file changed, 30 insertions(+)
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/zadanie_1.py"

C:\Users\user\Repozzitoriy_5>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 24 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 692 bytes | 692.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Vishhh123/Repozzitoriy_5.git
   2c9a902..9e14921  main -> main
```

Рис. 14 - сохранение, добавление коммита, отправка на Github.

Решите задачу: определите общие символы в двух строках, введенных с клавиатуры.

Рис. 15 - задание 2.

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

if __name__ == "__main__":

    stroka_1 = input("Введите первую строку: ")

    stroka_2 = input("Введите вторую строку: ")

    mnojestvo_1 = set(stroka_1)
    mnojestvo_2 = set(stroka_2)

    obshchie = mnojestvo_1.intersection(mnojestvo_2)

    print(f"Общие символы в двух строках: {obshchie}")
    print(f"Количество общих символов: {len(obshchie)}")
```

```
adanie_2
  "C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Задания/zadanie_2.py
  Введите первую строку: 12396432
  Введите вторую строку: 4647873223сыавтнт
  Общие символы в двух строках: {'4', '6', '2', '3'}
  Количество общих символов: 4

  Process finished with exit code 0
```

Рис. 16 - код к заданию 2.

```
C:\Users\user\Repozzitoriy_5>git add .

C:\Users\user\Repozzitoriy_5>git commit -m "Код к заданию 2"
[main 64f4527] Код к заданию 2
 2 files changed, 21 insertions(+), 1 deletion(-)
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/zadanie_2.py"

C:\Users\user\Repozzitoriy_5>git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 24 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 699 bytes | 699.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Vishhh123/Repozzitoriy_5.git
   9e14921..64f4527  main -> main
```

Рис. 17 - сохранение, добавление коммита, отправка на Github.

Решите задачу: создайте словарь, связав его с переменной `school`, и наполните данными, которые бы отражали количество учащихся в разных классах (1а, 1б, 2б, 6а, 7в и т. п.). Внесите изменения в словарь согласно следующему: а) в одном из классов изменилось количество учащихся, б) в школе появился новый класс, с) в школе был расформирован (удален) другой класс. Вычислите общее количество учащихся в школе.

Рис. 18 - задание 3.

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

if __name__ == "__main__":

    school = {'1а': 25, '1б': 30, '2б': 28, '6а': 32, '7в': 29}

    print(f"Начальный словарь: {school}")

    school['1а'] = 27
    print(f" В классе 1а изменилось кол-во учащихся: {school}")

    school['8г'] = 31
    print(f"Появление нового класса 8г: {school}")

    del school['2б']
    print(f"Был расформирован один класса 2б: {school}")

    obshchee_kolichestvo = sum(school.values())
    print(f"Общее количество учеников в школе: {obshchee_kolichestvo}")
```

```
adanie_3
"C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Задания/zadanie_3.py
Начальный словарь: {'1а': 25, '1б': 30, '2б': 28, '6а': 32, '7в': 29}
 В классе 1а изменилось кол-во учащихся: {'1а': 27, '1б': 30, '2б': 28, '6а': 32, '7в': 29}
Появление нового класса 8г: {'1а': 27, '1б': 30, '2б': 28, '6а': 32, '7в': 29, '8г': 31}
Был расформирован один класса 2б: {'1а': 27, '1б': 30, '6а': 32, '7в': 29, '8г': 31}
Общее количество учеников в школе: 149
```

Рис. 19 - код к заданию 3.

```
C:\Users\user\Repozzitoriy_5>git add .

C:\Users\user\Repozzitoriy_5>git commit -m "код к заданию 3"
[main 0cabae4] код к заданию 3
 1 file changed, 22 insertions(+)
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/zadanie_3.py"

C:\Users\user\Repozzitoriy_5>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 24 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 771 bytes | 771.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Vishhh123/Repozzitoriy_5.git
   64f4527..0cabae4  main -> main
```

Рис. 20 - сохранение, добавление коммита, отправка на Github.

Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод *items()*, с с помощью полученного объекта `dict_items` создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.

Рис. 21 - задание 4.

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

if __name__ == "__main__":

    slovar1 = {1: 'one', 2: 'two', 3: 'three'}

    print(slovar1)

    items_object = slovar1.items()

    print(items_object)

    slovar2 = {v: k for k, v in items_object}

    print(slovar2)
```

```
zadanie_4
"C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Задания/zadanie_4.py
{1: 'one', 2: 'two', 3: 'three'}
dict_items([(1, 'one'), (2, 'two'), (3, 'three')])
{'one': 1, 'two': 2, 'three': 3}

Process finished with exit code 0
```

Рис. 22 - код к заданию 4.

```
C:\Users\user\Repozzitoriy_5>git add .

C:\Users\user\Repozzitoriy_5>git commit -m "Код к заданию 4"
[main 7ed38fe] Код к заданию 4
 1 file changed, 18 insertions(+)
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/zadanie_4.py"
```

Рис. 23 - сохранение, добавление коммита.

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

$$A = \{a, e, g, o, p\};$$
$$B = \{e, h, i, o, u\};$$
$$C = \{g, h, p, s, t, w\};$$
$$D = \{f, h, n, s, t, x, y\};$$
$$X = (A/C) \cap \bar{B};$$
$$Y = (\bar{A} \cap \bar{B})/(C \cup D).$$

Рис. 24 - индивидуальное задание 1.

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

if __name__ == "__main__":

    u = set("abcdefghijklmnopqrstuvwxyz")

    a = {"a", "e", "g", "o", "p"}
    b = {"e", "h", "i", "o", "u"}
    c = {"g", "h", "p", "s", "t", "w"}
    d = {"f", "h", "n", "s", "t", "x", "y"}

    x = (a.difference(c)).intersection(u.difference(b))
    y = (u.difference(a).intersection(u.difference(b))).difference(c.union(d))

    print(f"x = {x}")
    print(f"y = {y}")
```

individual__1

```
"C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Задания/individual__1.py
x = {'a'}
y = {'c', 'v', 'l', 'b', 'm', 'd', 'q', 'k', 'z', 'j', 'r'}
```

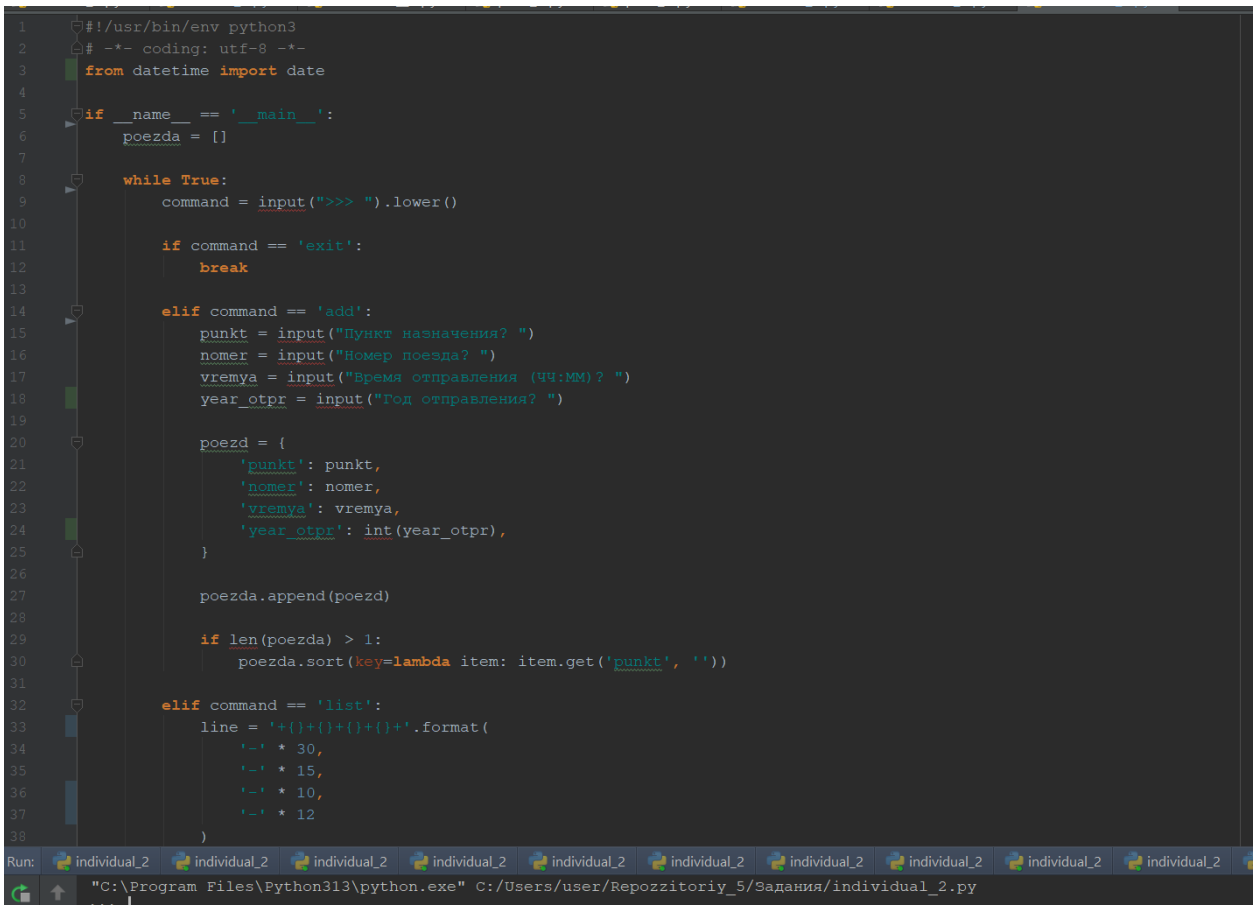
Рис. …25- код к индивидуальному заданию 1.

```
C:\Users\user\Repozzitoriy_5>git add .

C:\Users\user\Repozzitoriy_5>git commit -m "Код к индивидуальному заданию 1"
[main 8f11a93] Код к индивидуальному заданию 1
 1 file changed, 19 insertions(+)
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/individual__1.py"
```

Рис. 26 - сохранение, добавление коммита.

Использовать словарь, содержащий следующие ключи: название пункта назначения; номер поезда; время отправления. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения; вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени; если таких поездов нет, выдать на дисплей соответствующее сообщение.

Рис. 28 - индивидуальное задание 2.

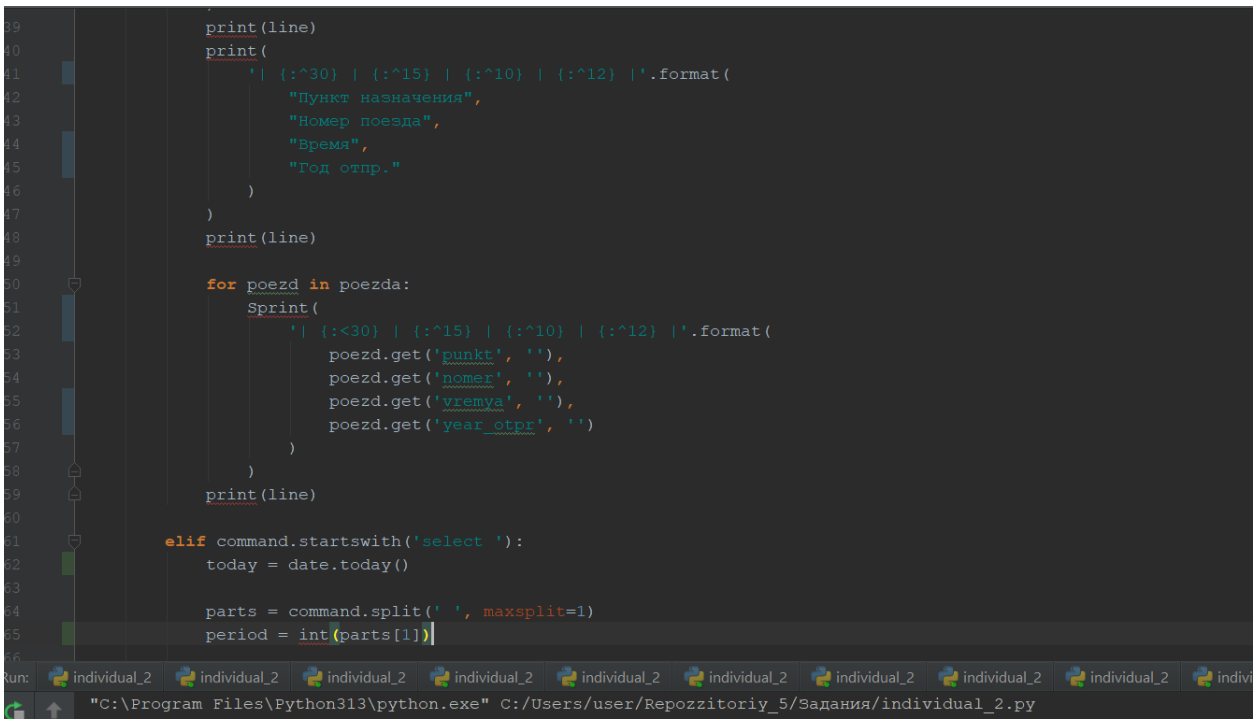


```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from datetime import date

if __name__ == '__main__':
    poezda = []

    while True:
        command = input(">>> ").lower()

        if command == 'exit':
            break

        elif command == 'add':
            punkt = input("Пункт назначения? ")
            nomer = input("Номер поезда? ")
            vremya = input("Время отправления (ЧЧ:ММ)? ")
            year_otpr = input("Год отправления? ")

            poezd = {
                'punkt': punkt,
                'nomer': nomer,
                'vremya': vremya,
                'year_otpr': int(year_otpr),
            }

            poezda.append(poezd)

            if len(poezda) > 1:
                poezda.sort(key=lambda item: item.get('punkt', ''))

        elif command == 'list':
            line = '+{}+{}+{}+{}+'.format(
                '-' * 30,
                '-' * 15,
                '-' * 10,
                '-' * 12
            )
```

Run: individual_2 individual_2 individual_2 individual_2 individual_2 individual_2 individual_2 individual_2 individual_2 individual_2

```
"C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Задания/individual_2.py
```

Рис. 29 - код к инд. заданию 2.



```
            print(line)
            print(
                '| {:^30} | {:^15} | {:^10} | {:^12} |'.format(
                    "Пункт назначения",
                    "Номер поезда",
                    "Время",
                    "Год отпр."
                )
            )
            print(line)

            for poezd in poezda:
                Sprint(
                    '| {:<30} | {:^15} | {:^10} | {:^12} |'.format(
                        poezd.get('punkt', ''),
                        poezd.get('nomer', ''),
                        poezd.get('vremya', ''),
                        poezd.get('year_otpr', '')
                    )
                )
            print(line)

        elif command.startswith('select '):
            today = date.today()

            parts = command.split(' ', maxsplit=1)
            period = int(parts[1])
```

Run: individual_2 individual_2 individual_2 individual_2 individual_2 individual_2 individual_2 individual_2 individual_2 indivi

```
"C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Задания/individual_2.py
```

Рис. 30 - код к инд. заданию 2.

```
            count = 0

            for poezd in poezda:
                if today.year - poezd.get('year_otpr', today.year) >= period:
                    count += 1
                    print(
                        '{:>4}: {} - №{} - {} (год: {})'.format(
                            count,
                            poezd.get('punkt', ''),
                            poezd.get('nomer', ''),
                            poezd.get('vremya', ''),
                            poezd.get('year_otpr', '')
                        )
                    )

            if count == 0:
                print("Поездов, добавленных более указанного периода, не найдено.")

        else:
            print("Неизвестная команда {}".format(command))
```

```
ndividual_2  individual_2  individual_2  individual_2  individual_2  individual_2  individual_2  individual_2  individual_2
"C:\Program Files\Python313\python.exe" C:/Users/user/Repozzitoriy_5/Задания/individual_2.py
>>>
```

Рис. 31 - код к инд. заданию 2.

```
>>> add
Пункт назначения? Сочи
Номер поезда? 123А
Время отправления (ЧЧ:ММ)? 12:30
Год отправления? 2020
>>> add
Пункт назначения? Лос Анджелес
Номер поезда? 234М
Время отправления (ЧЧ:ММ)? 14:10
Год отправления? 2025
>>> list
+-----------------------------+---------------+----------+------------+
|        Пункт назначения     |  Номер поезда |   Время  |  Год отпр. |
+-----------------------------+---------------+----------+------------+
| Лос Анджелес                |      234М     |   14:10  |     2025   |
| Сочи                        |      123А     |   12:30  |     2020   |
+-----------------------------+---------------+----------+------------+
>>> select 5
   1: Сочи - №123А - 12:30 (год: 2020)
>>> exit
```

Рис. 32 - результат выполнения программы.

```
C:\Users\user\Repozzitoriy_5>git add .

C:\Users\user\Repozzitoriy_5>git commit -m "код к инд.заданию 2"
[main ab2812e] код к инд.заданию 2
 2 files changed, 91 insertions(+), 3 deletions(-)
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/individual_2.py"

C:\Users\user\Repozzitoriy_5>git push
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 24 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (14/14), 2.39 KiB | 1.19 MiB/s, done.
Total 14 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 2 local objects.
To https://github.com/Vishhh123/Repozzitoriy_5.git
   0cabae4..ab2812e  main -> main
```

Рис. 33 - сохранение, добавление коммита, отправка на Github.

Ответы на контрольные вопросы:

1. Множества - неупорядоченные коллекции уникальных элементов без дубликатов.
2. Создание: my_set = {1, 2, 3} или my_set = set([1, 2, 3])
3. Проверка элемента: if элемент in множество: (присутствие) или if элемент not in множество: (отсутствие)
4. Перебор: for элемент in множество:
5. Set comprehension - создание множества в одной строке: {x for x in range(10) if x % 2 == 0}
6. Добавление: множество.add(элемент) или множество.update([элементы])
7. Удаление:
множество.remove(элемент) - удаляет элемент, ошибка если его нет
множество.discard(элемент) - удаляет если есть, без ошибки
множество.clear() - удаляет все элементы
8. Операции:
A | B или A.union(B) - объединение
A & B или A.intersection(B) - пересечение
A - B или A.difference(B) - разность
9. Надмножество/подмножество:
A.issuperset(B) - A надмножество B (все элементы B в A)
A.issubset(B) - A подмножество B (все элементы A в B)
10. Frozenset - неизменяемое множество (нельзя добавлять/удалять элементы)
11. Преобразование:
в строку: str(множество)
в список: list(множество)
в словарь: через zip или цикл, т.к. у множества нет пар ключ-значение
12. Словари - неупорядоченные коллекции пар ключ-значение, где ключи уникальны.
13. len() - да, возвращает количество пар ключ-значение: len(словарь)
14. Методы обхода:
for ключ in словарь:
for ключ, значение in словарь.items():
for ключ in словарь.keys():
for значение in словарь.values():
15. Получение значения:
словарь[ключ] - ошибка если ключа нет
словарь.get(ключ) или словарь.get(ключ, значение_по_умолчанию) безопасно
16. Установка значения:
словарь[ключ] = значение
словарь.update({ключ: значение})
17. Словарь включений - создание словаря в одной строке: {x: x**2 for x in range(5)}
18. Функция zip() - объединяет элементы из нескольких последовательностей:

python
names = ['Иван', 'Мария']
ages = [25, 30]
dict(zip(names, ages)) # {'Иван': 25, 'Мария': 30}
19. Модуль datetime - работа с датой и временем:
datetime.now() - текущая дата и время
date.today() - текущая дата
Создание дат: datetime(2024, 12, 25, 14, 30)
Форматирование: .strftime('%d.%m.%Y')
Разность дат: date1 - date2

Вывод: приобрели навыки по работе с множествами и словарями при написании программ с помощью языка программирования Python версии 3.x.