

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»**

Выполнил:
Щегольков Савва Игоревич
2 курс, группа ИВТ-б-о-24-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Автоматизированные системы
обработки информации и
управления», очная форма обучения

(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

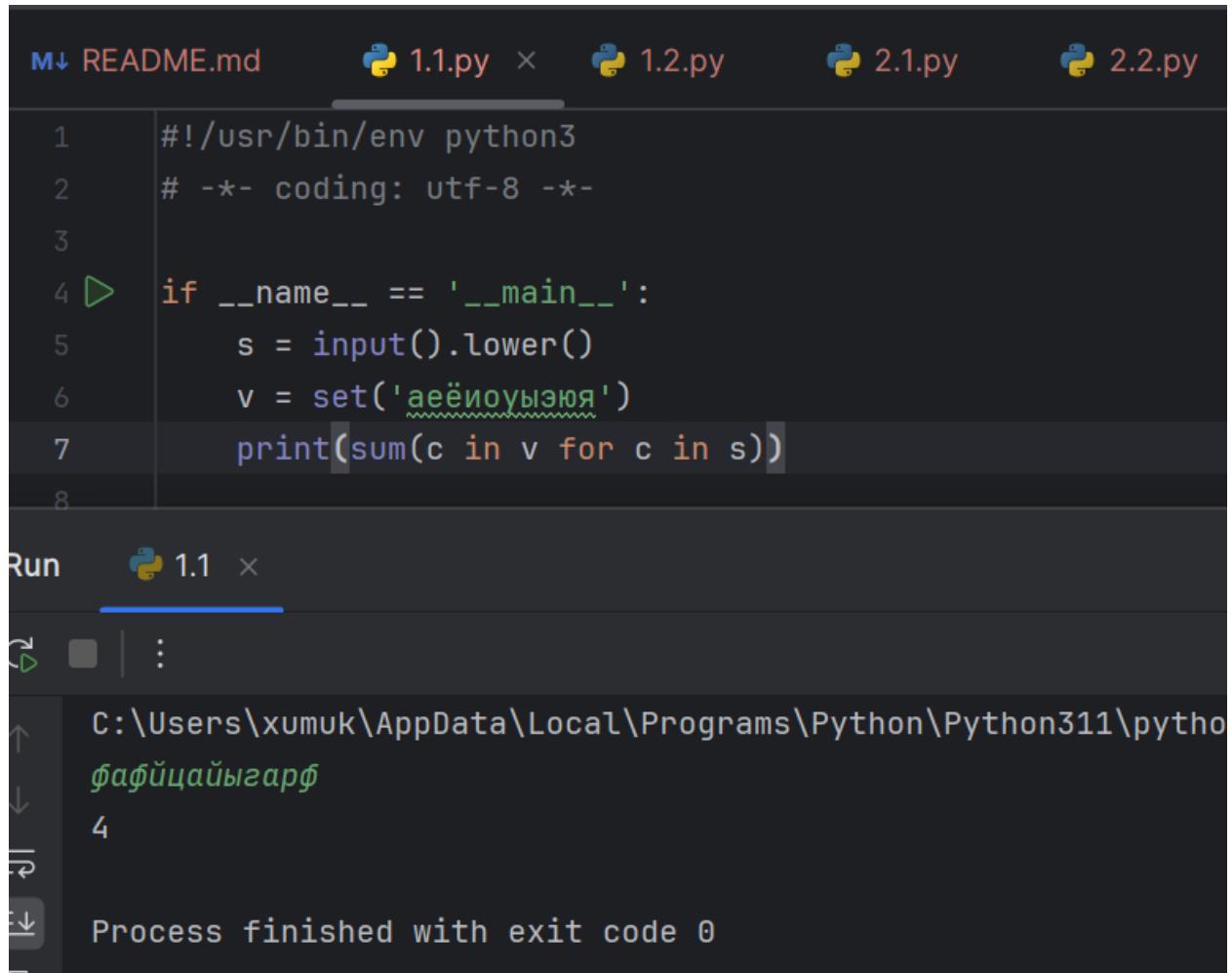
Тема: Работа с множествами и словарями в языке Python.

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

Порядок выполнения работы:

Ссылка на репозиторий: <https://github.com/xouixao/lab5>

Задание 1: Подсчитать количество гласных в строке, введённой с клавиатуры, с использованием множеств



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     s = input().lower()
6     v = set('aeёиоуыэюя')
7     print(sum(c in v for c in s))
8
```

Run 1.1

C:\Users\xumuk\AppData\Local\Programs\Python\Python311\python
фафыцайыгарф
4
Process finished with exit code 0

Рисунок 1. Задание 1

Задание 2: определить общие символы в двух строках, введённых с клавиатуры.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3 ▶  if __name__ == '__main__':
4      print(set(input()) & set(input()))
5
```

Run Python 1.2 ×

⟳ | :

```
C:\Users\xumuk\AppData\Local\Programs\Python\Python311\p
abcde
edceqwe
{'d', 'c', 'e'}
```

🖨 Process finished with exit code 0

Рисунок 2 - Задание 2.

Задание 3: Создать словарь school, изменить несколько значений, добавить и удалить класс, посчитать общее количество учеников.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3 ▶  if __name__ == '__main__':
4      school = {"1A": 25, "1B": 23, "2A": 27, "2B": 26}
5      school["1A"] = 26
6      school["3A"] = 20
7      del school["2B"]
8      print(sum(school.values()))
```

Run Python 1.3 ×

⟳ | :

```
C:\Users\xumuk\AppData\Local\Programs\Python\Python311\pyt
96
```

🖨 Process finished with exit code 0

Рисунок 3 – Задание 3

Задание 4: Создать словарь (ключи – числа, значения – строки), по items() сделать обратный словарь.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == '__main__':
4      d = {1: "one", 2: "two", 3: "three"}
5      d_rev = {v: k for k, v in d.items()}
6      print(d_rev)
7
```

Run 1.4 ×

⟳ | :

↑ C:\Users\xumuk\AppData\Local\Programs\Python\Py
↓ {'one': 1, 'two': 2, 'three': 3}
≡ Process finished with exit code 0
≡

Рисунок 4 - Задание 4.

$$\begin{aligned}A &= \{a, b, d, I, x\}; \\B &= \{d, e, h, i, n, u\}; \\C &= \{e, f, m, n\}; \\D &= \{a, c, h, k, r, s, w, x\}; \\X &= (A/C) \cap \bar{B}; \\Y &= (\bar{A} \cap D) \cup (C/B).\end{aligned}$$

Рисунок 5 – Условие задания 5.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == '__main__':
4      u = set("abcdefghijklmnopqrstuvwxyz")
5      a = {"a", "b", "d", "i", "x"}
6      b = {"d", "e", "h", "i", "n", "u"}
7      c = {"e", "f", "m", "n"}
8      d = {"a", "c", "h", "k", "r", "s", "w", "x"}
9      an = u.difference(a)
10     bn = u.difference(b)
11     x = a.difference(c).intersection(bn)
12     y = an.intersection(d).union(c.difference(b))
13     print(x)
14     print(y)
```

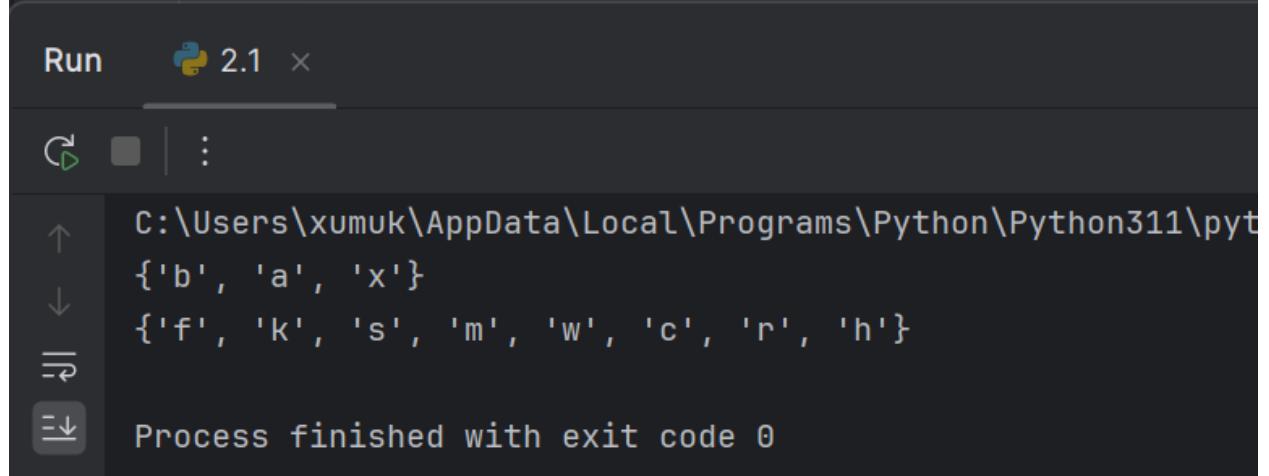


Рисунок 6 – Задание 5

5. Использовать словарь, содержащий следующие ключи: название пункта назначения рейса; номер рейса; тип самолета. 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 пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение.

Рисунок 7 – Условие задания 6

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3 ▶ if __name__ == '__main__':
4      r = []
5      while True:
6          comand = input().lower()
7          if comand == "exit":
8              break
9          if comand == "add":
10             to = input("To where? ")
11             n = input("Flight id? ")
12             t = input("Type of plain? ")
13             rn = {
14                 "to": to,
15                 "n": n,
16                 "t": t
17             }
18             r.append(rn)
19             if len(r) > 0:
20                 r.sort(key=lambda item: item.get("to"))
21             if comand == "list":
22                 t1 = input("Type of plain? ")
23                 d = 0
24                 for rn in r:
25                     if rn.get("t") == t1:
26                         print("Going to " + rn.get("to") + " With flight id " + rn.get("n"))
27                         d += 1
28                 if d == 0:
29                     print("Not a type of plain or that type doesn't servicing a flight")

```

Рисунок 8 – Задание 6

Контрольные вопросы:

1. Множество (set) в Python — это неупорядоченная коллекция уникальных хешируемых элементов; оно изменяется, не допускает дубликатов и оптимизировано под быстрые операции проверки принадлежности и множество-логические операции.

2. Создают множества литералом {1, 2, 3}, функцией set(iterable) (например, set([1,2,2])), пустое множество только через set(), а также с помощью set-включений вида {expr for x in iterable}.

3. Проверка присутствия или отсутствия элемента выполняется операторами in и not in: выражения x in s и x not in s возвращают логическое значение, при этом проверка в среднем выполняется за O(1).

4. Перебор элементов множества реализуют циклом for: for x in s: ...; порядок обхода не гарантируется, но можно дополнительно отсортировать элементы: for x in sorted(s):

5. Set comprehension — это конструкция вида {expr for x in iterable if cond}, которая за один проход формирует новое множество, одновременно преобразуя элементы и отфильтровывая ненужные.

6. Добавление одного элемента во множество выполняется методом s.add(x), а сразу нескольких — s.update(iterable), где iterable может быть списком, множеством, кортежем и т.п.

7. Удаление элемента выполняют методами s.remove(x) (бросает KeyError, если элемента нет) или s.discard(x) (молча игнорирует отсутствие); s.pop() удаляет и возвращает произвольный элемент, а s.clear() полностью очищает множество.

8. Основные операции над множествами: объединение s | t или s.union(t), пересечение s & t или s.intersection(t), разность s - t или s.difference(t) симметрическая разность s ^ t или s.symmetric_difference(t); есть также присваивающие варианты |=, &=, -=, ^=.

9. Определить отношение подмножества и надмножества можно методами s.issubset(t) и s.issuperset(t) либо операторами <=, <, >=, >: например, a <= b означает, что a — подмножество b, а a >= b — надмножество.

10. Множество frozenset — это неизменяемый (иммутабельный) вариант множества: элементы в нем те же (уникальные, неупорядоченные), но после создания его нельзя модифицировать, зато такие объекты хешируемы и могут использоваться как ключи словаря или элементы других множеств.

11. Преобразование множества в строку чаще всего выполняют через str(s) или с помощью ' '.join(map(str, s)); в список и кортеж — через list(s) и tuple(s); в словарь множество можно превратить, например, используя генератор пар: d = {x: True for x in s} или dict.fromkeys(s, value).

12. Словарь (`dict`) — это изменяемое отображение «ключ → значение», где ключи уникальны и хешируемы, а значения могут быть любыми объектами; словарь хранит пары и позволяет очень быстро получать значение по ключу.

13. Функция `len()` применяется к словарям и возвращает количество пар ключ—значение, то есть размер словаря: `len(d)`.

14. Обход словаря можно выполнять по ключам `for k in d: ...` или `for k in d.keys(): ...`, по значениям `for v in d.values(): ...`, а также сразу по парам `for k, v in d.items(): ...`; все эти варианты возвращают итерируемые объекты, которые можно использовать в циклах и генераторах.

15. Значение по ключу получают оператором индексирования `d[key]` (при отсутствии ключа будет `KeyError`), методом `d.get(key, default=None)` для безопасного получения с значением по умолчанию, либо `d.setdefault(key, default)` — он возвращает значение по ключу и при отсутствии создаёт новую пару с указанным значением.

16. Установить значение по ключу можно простым присваиванием `d[key] = value` (ключ создаётся или значение перезаписывается), методом `d.update({key: value})` или `d.update(iterable_of_pairs)`, а также через `setdefault`, если нужно задать значение только для отсутствующих ключей.

17. Словарь включений (`dict comprehension`) — это выражение вида `{key_expr: value_expr for x in iterable if cond}`, позволяющее за один проход сформировать новый словарь из другого итерируемого объекта с одновременной обработкой и фильтрацией данных.

18. Функция `zip()` создаёт итератор кортежей, «склеивая» несколько итерируемых объектов по позициям: `zip(a, b)` даёт пары `(a[i], b[i])`; её используют для параллельного обхода коллекций (`for x, y in zip(xs, ys): ...`), построения словарей `dict(zip(keys, values))`, а также для «транспонирования» матриц `list(zip(*matrix))`.

19. Модуль `datetime` предоставляет классы `date`, `time`, `datetime`, `timedelta`, `timezone` и функции для работы с датой и временем: получение текущего момента (`datetime.now()`, `date.today()`), арифметику с интервалами (`datetime + timedelta`), сравнение дат, форматирование и разбор строк (`strftime`, `strptime`), управление часовыми поясами и перевод времени между ними.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе с множествами и словарями при написании программ.