

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»
Вариант 19**

Выполнил:
Поляков Никита Александрович
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., доцент департамента
цифровых, робототехнических систем и
электроники института перспективной
инженерии

(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.

Практическая часть:

Для начала был создан новый репозиторий на GitHub. Ссылка на репозиторий: https://github.com/Kovirum/PyProg_Practice_5

Далее в отдельных модулях языка Python были проработаны примеры лабораторной работы.

Пример 1. Определить результат выполнения операций над множествами. Считать элементы множества строками.

$$\begin{aligned}A &= \{b, c, h, o\}; \\B &= \{d, f, g, o, v, y\}; \\C &= \{d, e, j, k\}; \\D &= \{a, b, f, g\}; \\X &= (A \cap B) \cup C; \\Y &= (A / D) \cup (\bar{C} / \bar{B}).\end{aligned}$$

Рисунок 1. Условия для примера 1

Выполнение примера:

The screenshot shows the PyCharm IDE interface. The left sidebar displays a project structure for 'PyProg_Practice_5' with files like 'example1.py', 'example2.py', 'main_task_1.py', and 'main_task_2.py'. The main editor window contains Python code for set operations. The bottom run tab shows the output of 'example1.py' execution, displaying the sets 'x' and 'y' and their intersections.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
if __name__ == "__main__":
    # Определим универсальное множество
    u = set("abcdefghijklmnopqrstuvwxyz")
    a = {'b', 'c', 'h', 'o'}
    b = {'d', 'f', 'g', 'o', 'v', '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}")

Process finished with exit code 0
```

Рисунок 2. Результат выполнения примера 1 лабораторной работы

Пример 2. Использовать словарь, содержащий следующие ключи: фамилия и инициалы работника, название занимаемой должности, год поступления на работу. Написать программу, выполняющую следующие действия:

- Ввод с клавиатуры данных в список, состоящий из заданных словарей
- Записи должны быть размещены по алфавиту

- Вывод на дисплей фамилий работников, чей стаж работы в организации превышал значение, введенное с клавиатуры
- Если таких работников нет, вывести на дисплей соответствующее сообщение

Выполнение примера:

```

Project  PyProg_Practice_5  main
example1.py  example2.py  main_task_1.py  main_task_2.py

PyProg_Practice_5 C:\Users\user\PycharmProjects\PyProg_Practice_5\examples
doc
examples
example1.py
example2.py
individual_tasks
individual_task_1.py
individual_task_2.py

print(line)

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

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

Run  example2  x

C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\PyProg_Practice_5\examples\example2.py"
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? Должность1
Год поступления? 2025
>>> add
Фамилия и инициалы? Сидоров С.С.
Должность? Должность2
Год поступления? 2000
>>> Петров П.П.
>>> Неизвестная команда петров п.п.

add
Фамилия и инициалы? Петров П.П.
Должность? Должность3
Год поступления? 2020
>>> list
+---+-----+-----+
| № | Ф.И.О. | Должность | Год |
+---+-----+-----+
| 1 | Иванов И.И. | Должность1 | 2025 |
+---+-----+-----+
| 2 | Петров П.П. | Должность3 | 2020 |
+---+-----+-----+
| 3 | Сидоров С.С. | Должность2 | 2000 |
+---+-----+-----+
>>> select 2021
Работники с заданным стажем не найдены
>>> select 10
1: Сидоров С.С.
>>> select 1

```

Рисунок 3. Результат выполнения примера 2 лабораторной работы

Далее были выполнены общие задания лабораторной работы:

Задание 1. Подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств.

Код выполненного задания:

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

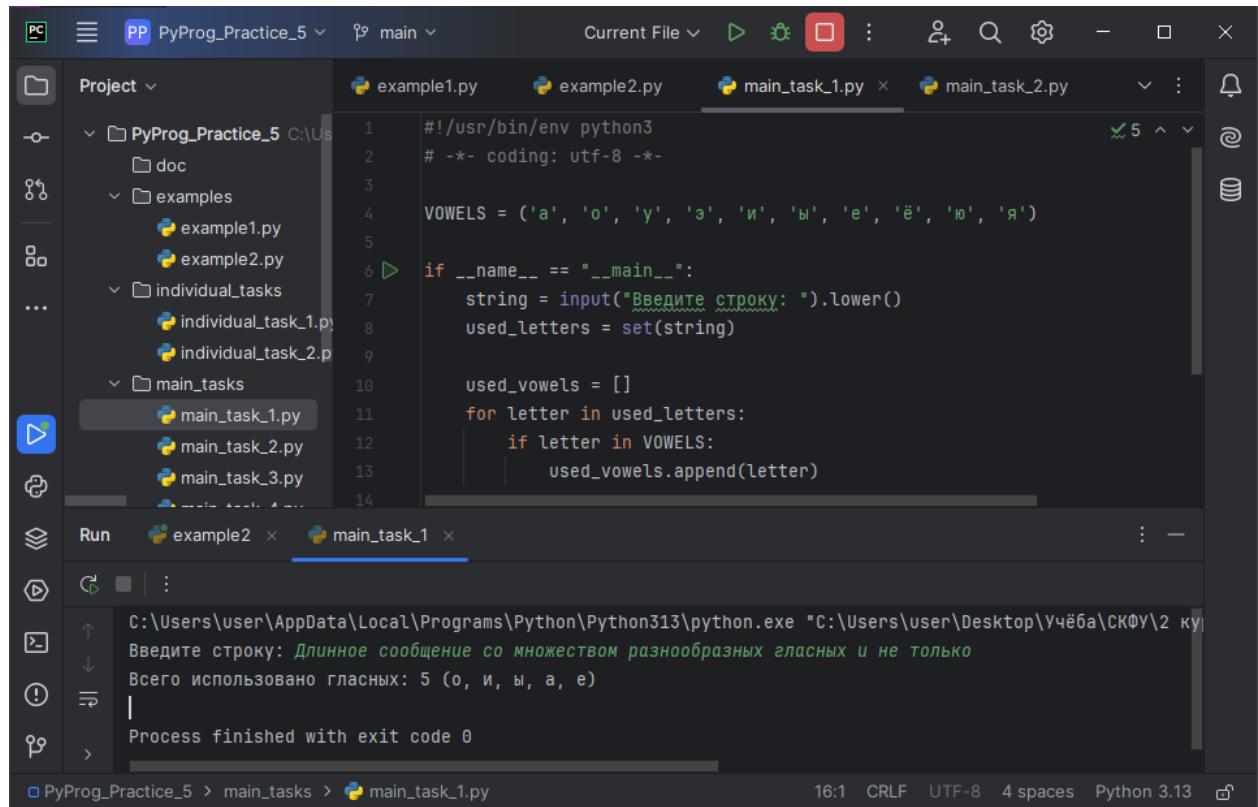
VOWELS = ('а', 'о', 'у', 'э', 'и', 'ы', 'е', 'ё', 'ю',
'я')

if __name__ == "__main__":
    string = input("Введите строку: ").lower()
    used_letters = set(string)

    used_vowels = []
    for letter in used_letters:
        if letter in VOWELS:
            used_vowels.append(letter)

    print(f"Всего использовано гласных:
{len(used_vowels)} ({', '.join(used_vowels)})")
```

Демонстрация выполнения задания:



The screenshot shows the PyCharm IDE interface with the following details:

- Project:** PyProg_Practice_5
- Current File:** main_task_1.py
- Code Content:**

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

VOWELS = ('а', 'о', 'у', 'э', 'и', 'ы', 'е', 'ё', 'ю',
'я')

if __name__ == "__main__":
    string = input("Введите строку: ").lower()
    used_letters = set(string)

    used_vowels = []
    for letter in used_letters:
        if letter in VOWELS:
            used_vowels.append(letter)

    print(f"Всего использовано гласных:
{len(used_vowels)} ({', '.join(used_vowels)})")
```
- Run Tab:** Shows the command: C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\PyProg_Practice_5\main_tasks\main_task_1.py". The output window displays:

```
Введите строку: Длинное сообщение со множеством разнообразных гласных и не только
Всего использовано гласных: 5 (о, и, ы, а, е)
Process finished with exit code 0
```
- Status Bar:** 16:1 CRLF UTF-8 4 spaces Python 3.13

Рисунок 4. Демонстрация работы основного задания 1

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

Код выполненного задания:

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

if __name__ == '__main__':
    string1 = input("Введите первую строку: ")
    string2 = input("Введите вторую строку: ")

    string1_letters = set(string1)
    string2_letters = set(string2)

    common_letters = []

    for string1_letter in string1_letters:
        if string1_letter in string2_letters:
            common_letters.append(string1_letter)

    print(f"Общих символов в строках: {len(common_letters)} ({', '.join(common_letters)})")
```

Демонстрация выполнения задания:

The screenshot shows the PyCharm IDE interface with the project 'PyProg_Practice_5' open. The 'main_task_2.py' file is selected in the top navigation bar. The code in the editor window is identical to the one provided above. In the bottom terminal window, the user inputs two strings: 'первая строка' and 'совсем другое'. The program outputs the count of common letters (6) and their list ('в, с, , р, е, о'). The status bar at the bottom right indicates the process finished with exit code 0.

Рисунок 5. Демонстрация выполнения основного задания 2

Задание 3. 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 другой класс. Вычислите общее количество учащихся в школе

Код выполненного задания:

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

if __name__ == '__main__':
    school = {
        '1б': 30,
        '2а': 35,
        '7в': 23,
        '11а': 15,
        '5в': 25,
        '6я': 19
    }

    # В одном из классов изменилось количество учащихся
    school['11а'] += 2

    # В школе появился новый класс
    school['9а'] = 20

    # В школе был расформирован (удален) другой класс
    del school['6я']

    # Общее количество учащихся:
    cnt = 0
    for _, class_student_count in school.items(): #
        Можно было бы и просто values(), но так прикольнее
        cnt += class_student_count

print(f"Всего учеников в школе: {cnt}")
```

Демонстрация выполнения задания:

The screenshot shows the PyCharm IDE interface. The top bar displays the project name 'PyProg_Practice_5' and the current file 'main_task_3.py'. The left sidebar shows the project structure with files like 'example2.py', 'main_task_1.py', 'main_task_2.py', and 'main_task_3.py'. The main code editor window contains Python code for a school student count system. The terminal below shows the execution results.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     school = {
6         '16': 30,
7         '2а': 35,
8         '7в': 23,
9         '11а': 15,
10        '5з': 25,
11        '6я': 19
12    }
13
14    # В одном из классов изменилось количество учащихся
15    school['11а'] += 2
16
17    # В школе появился новый класс
18    school['9а'] = 20
19
20    # В школе был расформирован (удален) другой класс
21    del school['6я']
22
23    # Общее количество учащихся:
24    cnt = 0
25    for _, class_student_count in school.items(): # Можно было бы и просто
26        cnt += class_student_count
27
28    print(f"Всего учеников в школе: {cnt}")
29
```

Run example2 x main_task_3 x

C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\PyProg_Practice_5\main_tasks\main_task_3.py"
Всего учеников в школе: 150
Process finished with exit code 0

PyProg_Practice_5 > main_tasks > main_task_3.py

Рисунок 6. Демонстрация выполнения основного задания 3

Задание 4. Создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод items(), с помощью полученного объекта dict_items создайте новый словарь, «обратный» исходному, т.е ключами являются строки, а значениями – числа.

Код выполненного задания:

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

```
if __name__ == '__main__':
    dictionary = {
        'a': 1,
        'b': 2,
        'c': 3,
        'd': 4,
        'e': 5,
        'f': 6,
        'g': 7,
        'h': 8,
        'i': 9,
        'j': 10,
        'k': 11,
        'l': 12,
        'm': 13
    }

    dict_items = dictionary.items()
    dictionary2 = {}

    for key, value in dict_items:
        dictionary2[value] = key

    print(f"Исходный словарь: {dictionary}")
    print(f"Словарь с измененными местами ключами и
значениями: {dictionary2}")
```

Демонстрация выполнения задания:

The screenshot shows the PyCharm IDE interface. The top bar displays the project name 'PyProg_Practice_5' and the current file 'main_task_4.py'. The left sidebar shows the project structure with files like 'main_task_1.py', 'main_task_2.py', 'main_task_3.py', and 'main_task_4.py'. The main editor window contains Python code that prints two dictionaries. The bottom terminal window shows the execution results:

```
C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\Индивидуальные задания\PyProg_Practice_5\main_tasks\main_task_4.py"
Исходный словарь: {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8, 'i': 9, 'j': 10, 'k': 11, 'l': 12, 'm': 13}
Словарь с измененными местами ключами и значениями: {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f', 7: 'g', 8: 'h', 9: 'i', 10: 'j', 11: 'k', 12: 'l', 13: 'm'}
Process finished with exit code 0
```

Рисунок 7. Демонстрация выполнения основного задания 4

Далее были выполнены индивидуальные задания согласно варианту, а также составлены UML-диаграммы деятельности:

Индивидуальное задание 1. Определить результат выполнения операций над множествами. Считать элементы массива строками.

$$\begin{aligned}
A &= \{a, b, f, g, i\}; \\
B &= \{c, f, g, i, s, v\}; \\
C &= \{a, g, h, i\}; \\
D &= \{f, w, x\}; \\
X &= (A \cap B) \cup C; \\
Y &= (A \cap \bar{B}) \cup (C \setminus D).
\end{aligned}$$

Рисунок 8. Условия для индивидуального задания 1

Код выполненного задания:

```

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

if __name__ == '__main__':
    # Определим универсальное множество
    u = set("abcdefghijklmnopqrstuvwxyz")

    A = {'a', 'b', 'f', 'g', 'i'}
    B = {'c', 'f', 'g', 'i', 's', 'v'}
    C = {'a', 'g', 'h', 'i'}
    D = {'f', 'w', 'x'}

    X = (A.intersection(B)).union(C)
    Y =
    (A.intersection(u.difference(B))).union(C.difference(B))
)

    print(f"X={X}")
    print(f"Y={Y}")

```

Демонстрация выполнения задания:

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a file structure for a project named 'PyProg_Practice_5'. It includes a 'doc' folder, an 'examples' folder containing 'example1.py' and 'example2.py', an 'individual_tasks' folder containing 'individual_task_1.py', 'individual_task_2.py', 'main_task_1.py', 'main_task_2.py', 'main_task_3.py', and 'main_task_4.py', and files '.gitignore', 'LICENSE', and 'README.md'. In the center, the Editor tab for 'individual_task_1.py' is active, showing Python code that performs set operations on strings. On the right, the Run tool window shows the command-line output of running the script, displaying the sets X and Y and their intersections and differences.

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

if __name__ == '__main__':
    # Определим универсальное множество
    u = set("abcdefghijklmnopqrstuvwxyz")

    A = {'a', 'b', 'f', 'g', 'i'}
    B = {'c', 'f', 'g', 'i', 's', 'v'}
    C = {'a', 'g', 'h', 'i'}
    D = {'f', 'w', 'x'}

    X = (A.intersection(B)).union(C)
    Y = (A.intersection(u.difference(B))).union(C.difference(B))

    print(f"X={X}")
    print(f"Y={Y}")
```

Run example2 individual_task_1

```
C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\PyProg_Practice_5\individual_tasks\individual_task_1.py"
X={'f', 'i', 'g', 'a', 'h'}
Y={'h', 'b', 'a'}
Process finished with exit code 0
```

PyProg_Practice_5 > individual_tasks > individual_task_1.py 18:1 CRLF UTF-8 4 spaces Python 3.13

Рисунок 8. Демонстрация выполнения индивидуального задания 1

UML-диаграмма деятельности:

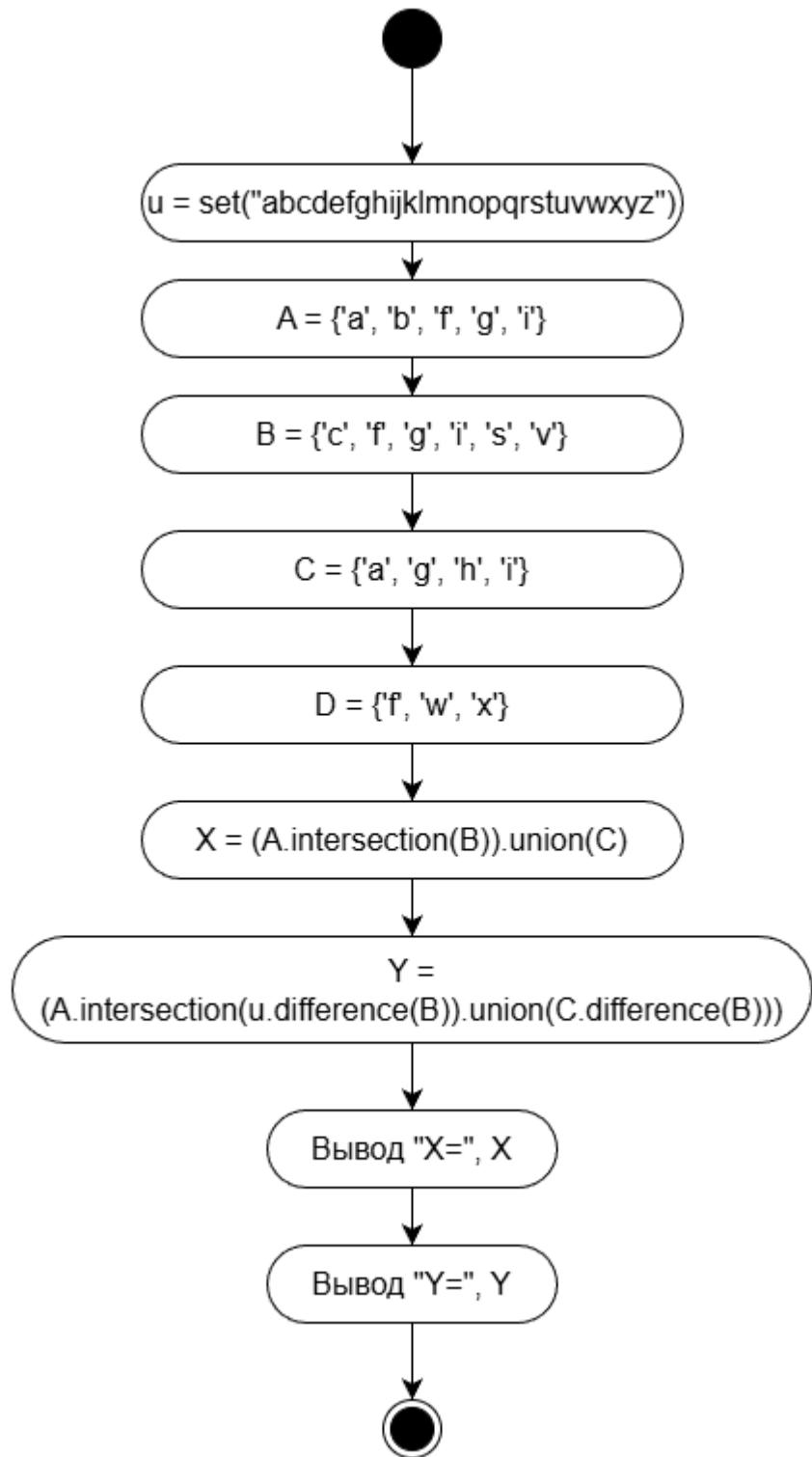


Рисунок 9. UML-диаграмма деятельности для индивидуального задания 1

Индивидуальное задание 2. Использовать словарь, содержащий следующие ключи: расчетный счет плательщика, расчетный счет получателя, перечисляемая сумма в руб. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей

заданной структуры, записи должны быть размещены в алфавитном порядке по расчетным счетам плательщиков, вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры, если такого расчетного счета нет, выдать на дисплей соответствующее сообщение.

Код выполненного задания:

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

import sys

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

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

        match command:
            case "add":
                payer_account_number =
int(input("Введите расчетный счет плательщика: "))
                    recipient_account_number =
int(input("Введите расчетный счет получателя: "))
                    amount = int(input("Введите перечисляемую сумму в рублях: "))

                new_transaction = {
                    'payer': payer_account_number,
                    'recipient':
recipient_account_number,
                    'amount': amount
                }

                transactions.append(new_transaction)

            if len(transactions) > 1:
                transactions.sort(key=lambda x:
x.get('payer', ''))

            case "list":
                print("Список операций:")

                for num, transaction in
```

```

enumerate(transactions, 1):
    print(f"Операция №{num}:")

    print(f"\tРасчетный счет
плательщика: {transaction.get('payer', '0')} ")
    print(f"\tРасчетный счет
получателя: {transaction.get('recipient', '0')} ")
    print(f"\tПеречисляемая сумма в
рублях: {transaction.get('amount', '0')} ")

case command if command.startswith("find"):
    target_payer = int(command.split()[1])

    amount = 0
    for transaction in transactions:
        if transaction.get('payer', '') == target_payer:
            amount += transaction.get('amount', 0)

        if amount == 0:
            print("Операции снятия денег с
данного расчетного счета не найдены!", file=sys.stderr)
        else:
            print(f"С расчетного счёта
плательщика №{target_payer} было снято всего {amount}
руб.")

case "exit":
    break

case _:
    print("Незвестная команда!",
file=sys.stderr)

```

Демонстрация выполнения задания:

The screenshot shows a PyCharm interface with the following details:

- Title Bar:** PyProg_Practice_5
- Project:** Project
- Files:** ADME.md, example1.py, individual_task_1.py, individual_task_2.py
- Terminal:** C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\PyProg_Practice_5\individual_tasks\individual_task_2.py"
- Output:**

```
>>> add
Введите расчетный счет плательщика: 100
Введите расчетный счет получателя: 200
Введите перечисляемую сумму в рублях: 589
>>> add
Введите расчетный счет плательщика: 100
Введите расчетный счет получателя: 200
Введите перечисляемую сумму в рублях: 12567
>>> add
Введите расчетный счет плательщика: 100
Введите расчетный счет получателя: 300
Введите перечисляемую сумму в рублях: 999
>>> list
Список операций:
Операция #1:
    Расчетный счет плательщика: 100
    Расчетный счет получателя: 200
    Перечисляемая сумма в рублях: 589
Операция #2:
    Расчетный счет плательщика: 100
    Расчетный счет получателя: 200
    Перечисляемая сумма в рублях: 12567
Операция #3:
    Расчетный счет плательщика: 100
    Расчетный счет получателя: 300
    Перечисляемая сумма в рублях: 999
Операция #4:
    Расчетный счет плательщика: 200
    Расчетный счет получателя: 900
    Перечисляемая сумма в рублях: 5678
>>> find 100
С расчетного счёта плательщика №100 было снято всего 14155 руб.
>>> find 200
С расчетного счёта плательщика №200 было снято всего 5678 руб.
```
- Status Bar:** PyProg_Practice_5 > individual_tasks > individual_task_2.py | 15:34 | CRLF | UTF-8 | 4 spaces | Python 3.13

Рисунок 10. Демонстрация выполнения индивидуального задания 2

UML-диаграмма деятельности:

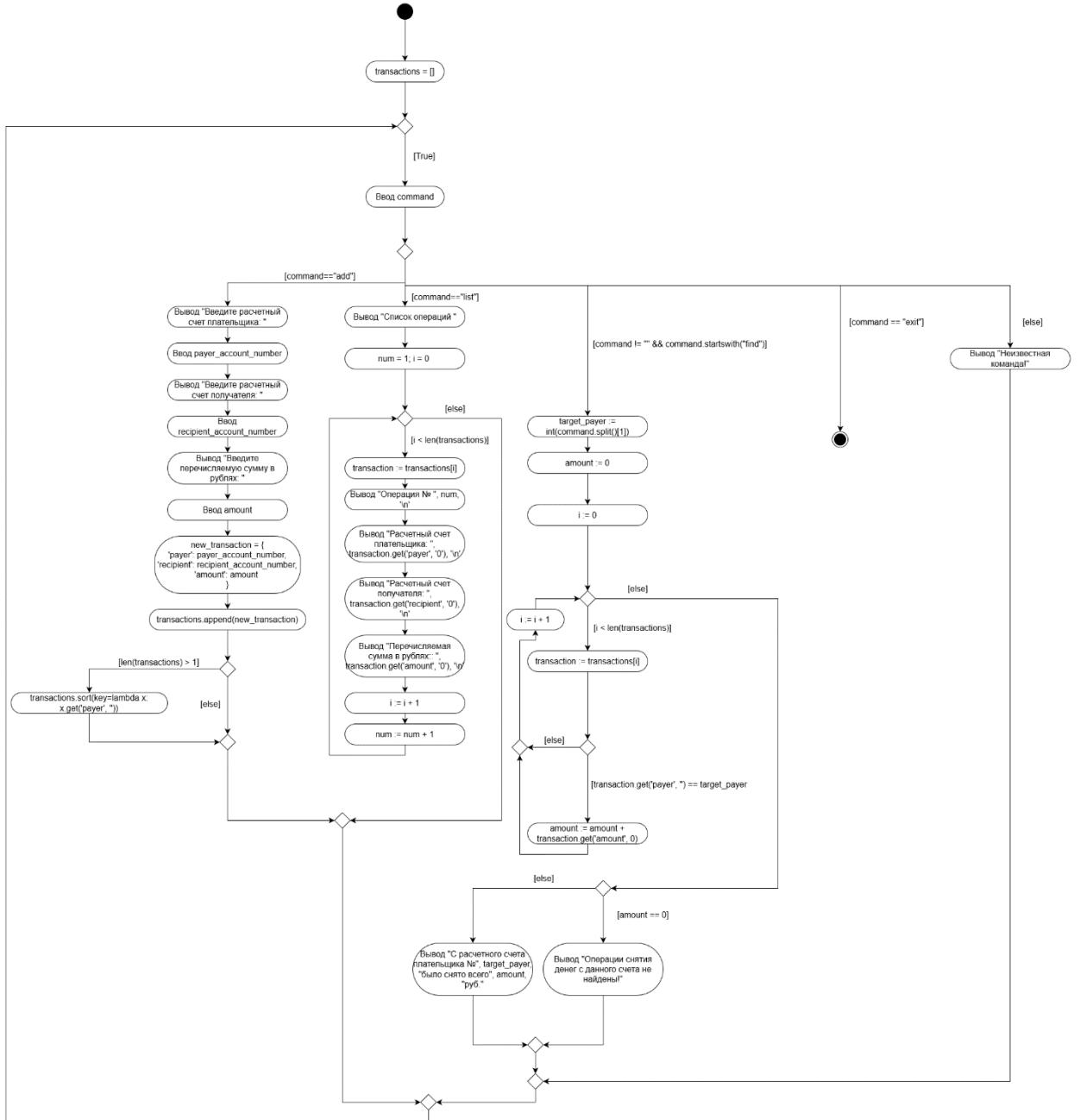


Рисунок 11. UML-диаграмма деятельности для индивидуального задания 2

Контрольные вопросы:

1. Что такое множества в языке Python?

Множество (set) — неупорядоченная, изменяемая коллекция уникальных элементов. Поддерживает математические операции (объединение, пересечение и др.). Элементы должны быть хешируемыми (неизменяемыми).

2. Как осуществляется создание множеств в Python?

`set()` — от любого итерируемого объекта;

{1, 2, 3} — литерал множества (нельзя {} для пустого, это словарь);

пустое множество: `set()`.

3. Как проверить присутствие/отсутствие элемента в множестве?

`element in set` — True, если элемент есть;

`element not in set` — True, если отсутствует.

4. Как выполнить перебор элементов множества?

`for item in set:` — порядок произвольный (не гарантируется).

5. Что такое set comprehension?

Генератор множества: `{x**2 for x in range(5)}` — создаёт множество из элементов, полученных в выражении, с возможной фильтрацией.

6. Как выполнить добавление элемента во множество?

`set.add(element)` — добавляет один элемент;

`set.update([a,b,c])` — добавляет несколько элементов из итерируемого объекта.

7. Как выполнить удаление одного или всех элементов множества?

`set.remove(e)` — удаляет e, `KeyError` при отсутствии;

`set.discard(e)` — удаляет e, ошибки нет;

`set.pop()` — удаляет и возвращает произвольный элемент;

`set.clear()` — удаляет все элементы.

8. Как выполняются основные операции над множествами: объединение, пересечение, разность?

`a | b` или `a.union(b)` — объединение;

`a & b` или `a.intersection(b)` — пересечение;

`a - b` или `a.difference(b)` — разность (в `a`, но не в `b`);

`a ^ b` или `a.symmetric_difference(b)` — симметрическая разность.

9. Как определить, что некоторое множество является надмножеством или подмножеством другого множества?

`a.issubset(b)` или `a <= b` — подмножество;

`a.issuperset(b)` или `a >= b` — надмножество;

строгие аналоги: `<` и `>`.

10. Каково назначение множеств `frozenset`?

Неизменяемое множество, хешируемое (можно ключом словаря или элементом другого множества). Создаётся `frozenset(iterable)`.

11. Как осуществляется преобразование множеств в строку, список, словарь?

`str(set)` — строковое представление;

`list(set)` — список элементов;

дикт: `dict.fromkeys(set)` или генератор: `{k: v for k, v in zip(set, values)}`.

12. Что такое словари в языке Python?

`dict` — неупорядоченная (до Python 3.7 — неупорядоченная, с 3.7 — упорядоченная по вставке) изменяемая коллекция пар ключ-значение. Ключи должны быть хешируемыми.

13. Может ли функция `len()` быть использована при работе со словарями?

Да, возвращает количество пар ключ-значение.

14. Какие методы обхода словарей Вам известны?

`.keys()`, `.values()`, `.items()` — возвращают view-объекты, поддерживающие итерацию;

for key in dict — перебор ключей;
for k, v in dict.items(): — перебор пар.

15. Какими способами можно получить значения из словаря по ключу?
dict[key] — KeyError при отсутствии;
dict.get(key) — None или заданное значение по умолчанию;
dict.setdefault(key, default) — возвращает значение, при отсутствии вставляет default.

16. Какими способами можно установить значение в словаре по ключу?
dict[key] = value — добавление/изменение;
dict.update({k: v}) или dict.update(k=v) — обновление несколькимиарами.

17. Что такое словарь включений?

{x: x**2 for x in range(5)} — генератор словаря, создаёт словарь из итерируемого объекта с выражением для ключей и значений.

18. Самостоятельно изучите возможности функции zip() приведите примеры ее использования.

zip(*iterables) — итератор, объединяющий элементы из нескольких итерируемых объектов в кортежи; останавливается по кратчайшему.

Примеры:

list(zip([1,2], ['a','b'])) → [(1,'a'), (2,'b')];
dict(zip(['a','b'], [1,2])) → {'a':1, 'b':2};
for a,b in zip(list1, list2): — параллельный обход.

19. Самостоятельно изучите возможности модуля datetime. Каким функционалом по работе с датой и временем обладает этот модуль?

Модуль предоставляет классы:

date — год, месяц, день;

`time` — час, минута, секунда, микросекунда;

`datetime` — комбинация даты и времени;

`timedelta` — разница во времени;

`tzinfo` — часовые пояса.

Методы: `today()`, `now()`, `strftime()`, `strptime()`, арифметика с `timedelta`.

Вывод:

В результате выполнения данной лабораторной работы были приобретены навыки по работе с множествами и словарями при написании программ с помощью языка программирования Python версии 3.x.