

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчёт о лабораторной работе №10 по дисциплине основы программной
инженерии**

Выполнила:

Нестеренко Тамара Антоновна,
2 курс, группа ПИЖ-б-о-20-1,

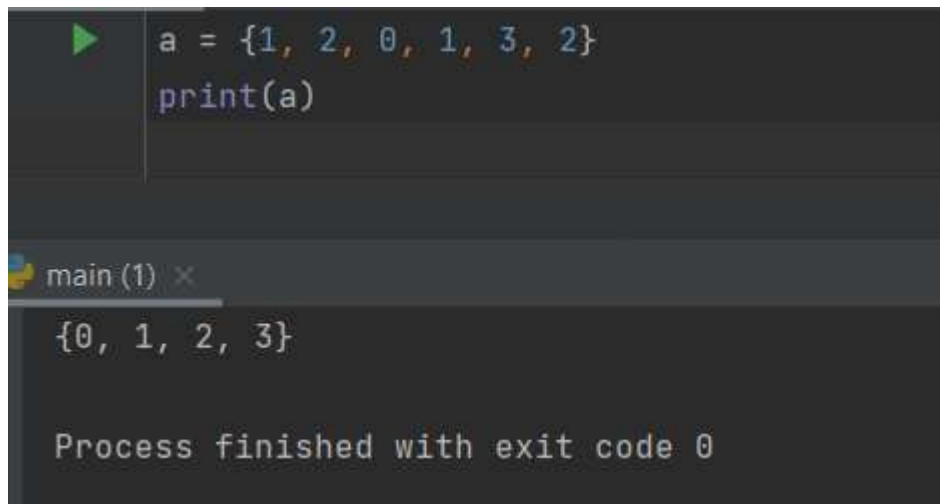
Проверил:

Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ

1. Практическая часть



```
a = {1, 2, 0, 1, 3, 2}
print(a)
```

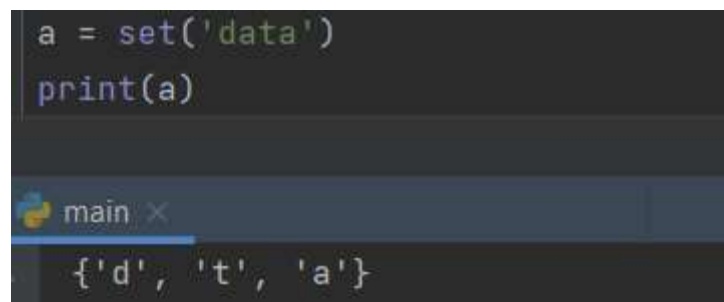
main (1) ×

```
{0, 1, 2, 3}
```

Process finished with exit code 0

The screenshot shows a Python script where a list with duplicate elements is assigned to a variable 'a'. When printed, the output is a set containing the unique elements of the list.

Рисунок 1 – Пример создания множества целых чисел



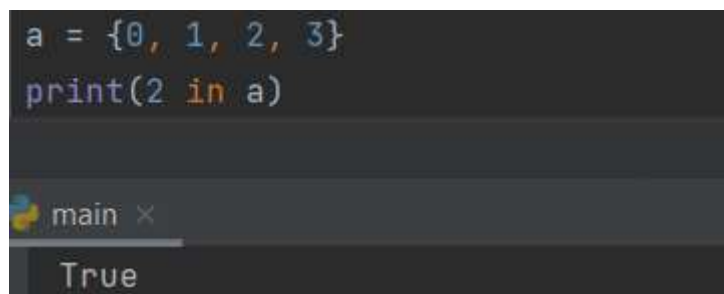
```
a = set('data')
print(a)
```

main ×

```
{'d', 't', 'a'}
```

The screenshot shows a Python script where a string is passed to the 'set()' function. The output is a set of the unique characters from the string.

Рисунок 2 – Пример создания множества с помощью метода set



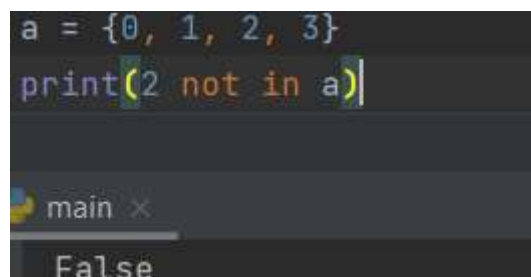
```
a = {0, 1, 2, 3}
print(2 in a)
```

main ×

```
True
```

The screenshot shows a Python script where a set is created and then a membership test is performed. The output is 'True' because the value 2 is present in the set.

Рисунок 3 – Пример проверки, есть ли данное значение в множестве



```
a = {0, 1, 2, 3}
print(2 not in a)
```

main ×

```
False
```

The screenshot shows a Python script where a set is created and then a non-membership test is performed. The output is 'False' because the value 2 is present in the set.

Рисунок 4 – Пример проверки отсутствия значения в множестве

```
for a in {0, 1, 2}:  
    print(a)  
  
for a in {0, 1, 2}  
  
main ×  
0  
1  
2
```

Рисунок 5 – Пример перебора элементов

```
a = {i for i in [1, 2, 0, 1, 3, 2]}  
print(a)  
  
main ×  
{0, 1, 2, 3}
```

Рисунок 6 – Пример генерации множества

```
a = {0, 1, 2, 3}  
print(len(a))  
  
main ×  
4
```

Рисунок 7 – Пример получения размера

```
a = {0, 1, 2, 3}  
a.add(4)  
print(a)  
  
main ×  
{0, 1, 2, 3, 4}
```

Рисунок 8 – Пример добавления элемента

```
a = {0, 1, 2, 3}
a.remove(3)
print(a)
```

main ×

{0, 1, 2}

Рисунок 9 – Пример удаления элемента

```
a = {0, 1, 2, 3}
a.clear()
print(a)
```

set()

main ×

set()

Рисунок 10 – Пример полной очистки

```
a = {0, 1, 12, 'b', 'ab', 3, 2, 'a'}
print(a)
```

main ×

{0, 1, 2, 3, 'ab', 'a', 12, 'b'}

Рисунок 11 – Пример порядка элементов в множестве

```
a = {0, 1, 12, 3, 2}
print(a)
```

main ×

{0, 1, 2, 3, 12}

Рисунок 12 – Пример порядка чисел в множестве

```
a = {0, 1, 12, 3, 2}
b = list(a)
print(b)
```

The screenshot shows a Python IDE with a dark theme. The code defines a set `a` with elements `0, 1, 12, 3, 2`, converts it to a list `b` using `list(a)`, and prints `b`. The output in the console is `[0, 1, 2, 3, 12]`, demonstrating that the order of elements in the list is not necessarily the same as in the set.

Рисунок 13 – Пример порядка элементов в множестве, преобразованном в список

```
a = {0, 1, 2, 3}
b = {4, 3, 2, 1}
c = a.union(b)
print(c)
```

The screenshot shows a Python IDE with a dark theme. The code defines two sets `a = {0, 1, 2, 3}` and `b = {4, 3, 2, 1}`, then creates a new set `c` as the union of `a` and `b` using `a.union(b)`. The output is `{0, 1, 2, 3, 4}`.

Рисунок 14 – Пример операции объединения

```
a = {0, 1, 2, 3}
b = {4, 3, 2, 1}
a.update(b)
print(a)
```

The screenshot shows a Python IDE with a dark theme. The code defines two sets `a = {0, 1, 2, 3}` and `b = {4, 3, 2, 1}`, then updates set `a` with the elements of set `b` using `a.update(b)`. The output is `{0, 1, 2, 3, 4}`.

Рисунок 15 – Пример операции добавления

```
a = {0, 1, 2, 3}
b = {4, 3, 2, 1}
c = a.intersection(b)
print(c)
```

main ×

{1, 2, 3}

Рисунок 16 – Пример операции пересечения

```
a = {0, 1, 2, 3}
b = {4, 3, 2, 1}
c = a.difference(b)
print(c)
```

main ×

{0}

Рисунок 17 – Пример операции разность

```
a = {0, 1, 2, 3, 4}
b = {3, 2, 1}
print(a.issubset(b))
```

main ×

False

Рисунок 18 – Пример определения подмножества

```
a = {0, 1, 2, 3, 4}
b = {3, 2, 1}
print(a.issuperset(b))
```

main ×

True

Рисунок 19 – Пример определения надмножества

```
a = frozenset({"hello", "world"})
print(a)
```

main ×

frozenset({'world', 'hello'})

Рисунок 20 – Пример использования типа frozenset

```
a = {'set', 'str', 'dict', 'list'}
b = ','.join(a)
print(b)
print(type(b))
```

main ×

dict,str,set,list
<class 'str'>

Рисунок 21 – Пример преобразования множества в строку

```
a = {('a', 2), ('b', 4)}  
b = dict(a)  
print(b)  
print(type(b))
```

main ×

```
{'a': 2, 'b': 4}  
<class 'dict'>
```

Рисунок 22 – Пример преобразования множества в словарь

```
a = {1, 2, 0, 1, 3, 2}  
b = list(a)  
print(b)  
print(type(b))
```

main ×

```
[0, 1, 2, 3]  
<class 'list'>
```

Рисунок 23 – Пример преобразования множества в список


```
#!/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}")

    # Найдем дополнения множеств
    bn = u.difference(b)
    cn = u.difference(c)

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

if __name__ == "__main__"
main x
x = {'e', 'o', 'k', 'j', 'd'}
y = {'f', 'v', 'g', 'c', 'o', 'y', 'h'}
```

Рисунок 24 – Пример решения задачи: определить результат выполнения операции над множествами. Считать элементы множества строками.

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

import sys

if __name__ == "__main__":
    u = set("aeiouy")
    x = input("Введите слово: ")
    count = 0
    for letter in x:
        if letter in u:
            count += 1
    print(f"Количество гласных равно: {count}")
```

Zadanie_1 ×

Введите слово: *hello, friend!*

Количество гласных равно: 5

Рисунок 25 – Пример решения задания №1

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

import sys

if __name__ == "__main__":
    x = set(input("Введите первое предложение: "))
    y = set(input("Введите второе предложение: "))
    common_letters = x.intersection(y)
    print(', '.join(common_letters))
```

Zadanie_2 x

Введите первое предложение: *hello, friend!*
Введите второе предложение: *bay, frined!*
e, , , r, !, f, , i, d, n

Рисунок 26 – Пример решения задания №2

```

# Вариант №16
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    u = set("abcdefghijklmnopqrstuvwxyz")
    a = {"b", "d", "f", "g", "l", "u"}
    b = {"d", "e", "f", "m", "n", "z"}
    c = {"h", "i", "r", "x", "y"}
    d = {"a", "e", "f", "k", "r", "s", "x"}

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

    an = u.difference(a)

    y = (an.intersection(d).union(c.difference(b)))
    print(f"y = {y}")

```

```

IDZ x
x = set()
y = {'x', 'h', 'e', 'a', 'k', 'r', 'i', 'y', 's'}

Process finished with exit code 0

```

Рисунок 27 – Пример решения индивидуального задания. Вариант №16

2. Вопросы для защиты работы

1. Что такое множества в языке Python?

Множество называется неупорядоченная совокупность уникальных значений. В качестве элементов набора данных могут выступать неизменяемые объекты, такие как числа, символы, строки.

2. Как осуществляется создание множеств в Python?

Присвоить переменной последовательность значений, выделив их фигурными скобками.

```
a = {1, 2, 0, 1, 3, 2}
print(a)

{0, 1, 2, 3}
```

Вызвать set.

```
a = set('data')
print(a)

{'d', 'a', 't'}
```

3. Как проверить присутствие/отсутствие элемента в множестве?

Проверка присутствия:

```
a = {0, 1, 2, 3}
print(2 in a)

True
```

Проверка отсутствия:

```
a = {0, 1, 2, 3}
print(2 not in a)

False
```

4. Как выполнить перебор элементов множества?

```
for a in {0, 1, 2}:
    print(a)

0
1
2
```

5. Что такое set comprehension?

Set comprehensions – генератор, позволяющий заполнять списки, а также другие наборы данных с учётом неких условий.

```
a = {i for i in [1, 2, 0, 1, 3, 2]}
print(a)

{0, 1, 2, 3}
```

6. Как выполнить добавление элемента во множество?

Чтобы внести новые значения, потребуется вызывать метод `add`. Аргументом в данном случае будет добавляемый элемент последовательности.

```
a = {0, 1, 2, 3}
a.add(4)
print(a)

{0, 1, 2, 3, 4}
```

7. Как выполнить удаление одного или всех элементов множества?

`Remove` – удаление элемента с генерацией исключения в случае, если такого элемента нет;

`Discard` – удаление элемента без генерации исключения, если элемент отсутствует;

`Pop` – удаление первого элемента, генерируется исключение при попытке удаления из пустого множества.

`Clear` – полная очистка.

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

Объединение:

```
a = {0, 1, 2, 3}
b = {4, 3, 2, 1}
c = a.union(b)
print(c)

{0, 1, 2, 3, 4}
```

Пересечение:

```
a = {0, 1, 2, 3}
b = {4, 3, 2, 1}
c = a.intersection(b)
print(c)

{1, 2, 3}
```

Разность:

```
a = {0, 1, 2, 3}
b = {4, 3, 2, 1}
c = a.difference(b)
print(c)

{0}
```

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

Определение подмножества:

```
a = {0, 1, 2, 3, 4}
b = {3, 2, 1}
print(a.issubset(b))

False
```

Определения надмножества:

```
a = {0, 1, 2, 3, 4}
b = {3, 2, 1}
print(a.issuperset(b))

True
```

9. Каково назначение множеств frozenset?

Множество, содержимое которого не поддаётся изменению имеет тип frozenset. Значения из этого набора нельзя удалить, как и добавить новые.

```
a = frozenset({"hello", "world"})
print(a)

frozenset({'hello', 'world'})
```

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

Для преобразования множества в строку используется конкатенация текстовых значений, которую обеспечивает функция join. В этом случае её аргументом является набор данных в виде нескольких строк. Запятая в кавычках выступает в качестве символа, разделяющего значения. Метод type возвращает тип данных объекта в конце приведённого кода.

```
a = {'set', 'str', 'dict', 'list'}
b = ','.join(a)
print(b)
print(type(b))

set,dict,list,str
<class 'str'>
```

Чтобы получить из множества словарь, следует передать функции dict набор из нескольких пар значений, в каждом из которых будет находиться ключ. Функция print демонстрирует на экране содержимое полученного объекта, а type отображает его тип.

```
a = {('a', 2), ('b', 4)}
b = dict(a)
print(b)
print(type(b))

{'b': 4, 'a': 2}
<class 'dict'>
```

По аналогии с предыдущими преобразованиями можно получить список неких объектов. На этот раз используется вызов list, получающий в качестве аргумента множество a. На выходе функции print отображаются уникальные значения для изначального набора чисел.

```
a = {1, 2, 0, 1, 3, 2}
b = list(a)
print(b)
print(type(b))

[0, 1, 2, 3]
<class 'list'>
```

Ссылка на репозиторий: https://github.com/tamaranesterenko/Python.LR_10