

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего
образования**

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра информационных систем и технологий

Отчет по лабораторной работе №10.

Дисциплина: «Основы программной инженерии»

Выполнил:

Студент группы ПИЖ-б-о-22-1,
направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Гуртовой Ярослав Дмитриевич

Проверил:

Воронкин Р. А.

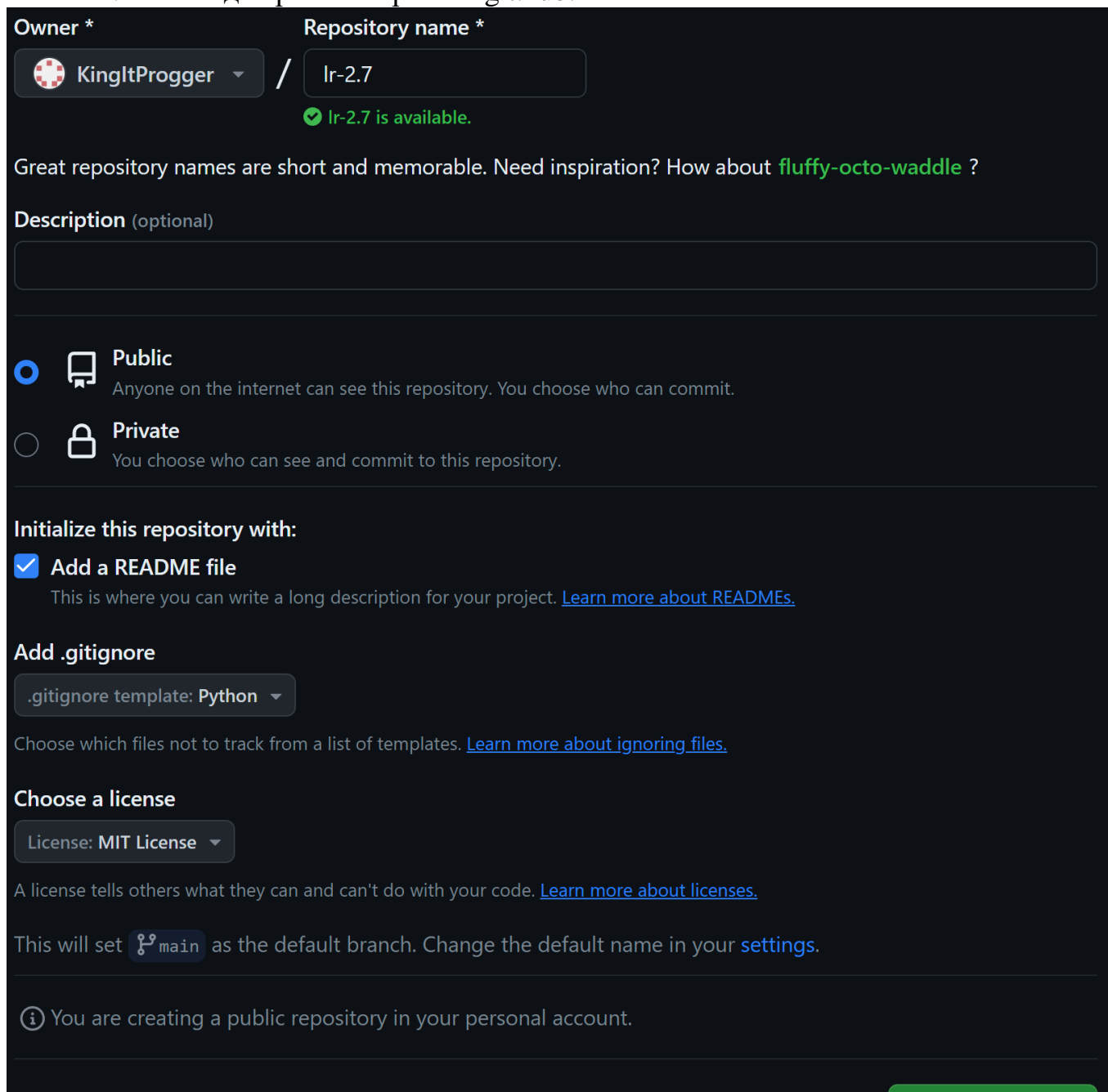
Ставрополь 2023

Тема: Лабораторная работа 2.7 Работа с множествами в языке Python.

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

Выполнение работы:

1. Изучил теоретический материал работы.
2. Создал репозиторий на git.hub.



The screenshot shows the GitHub 'Create new repository' page. At the top, the 'Owner' is set to 'KingItProgger' and the 'Repository name' is 'lr-2.7'. A green checkmark indicates 'lr-2.7 is available'. Below this, a suggestion for repository names is shown: 'fluffy-octo-waddle'. The 'Description' field is empty. Under 'Visibility', the 'Public' option is selected with a radio button, and the 'Private' option is unselected. The 'Initialize this repository with:' section has 'Add a README file' checked. Below this, the '.gitignore' template is set to 'Python'. The 'Choose a license' section has 'MIT License' selected. At the bottom, a note states: 'This will set main as the default branch. Change the default name in your settings.' and another note says: 'You are creating a public repository in your personal account.'

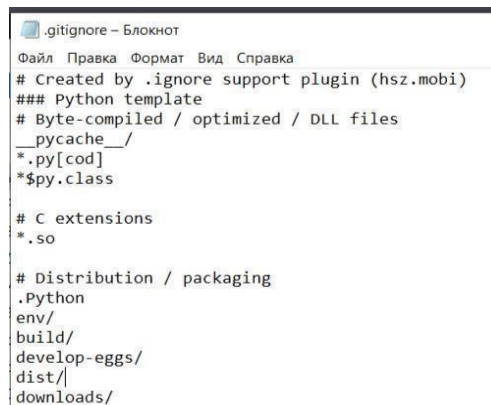
Рисунок 1 – создание репозитория

3. Клонировал репозиторий.

```
PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr_2.7> git clone http://...
Cloning into 'lr-2.7'...
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
Receiving objects: 100% (5/5), done.
PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr_2.7>
```

Рисунок 2 – клонирование репозитория 4.

4. Дополнить файл gitignore необходимыми правилами.



```
.gitignore - Блокнот
Файл Правка Формат Вид Справка
# Created by .ignore support plugin (hsz.mobi)
### Python template
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[od]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
```

Рисунок 3 – .gitignore для IDE PyCharm

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

```
PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr_2.7\lr-2.7> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr_2.7\lr-2.7>
```

Рисунок 4 – создание ветки develop

6. Проработал примеры из методички.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      # Определим универсальное множество
6      u = set("abcdefghijklmnopqrstuvwxyz")
7      a = {"b", "c", "h", "o"}
8      b = {"d", "f", "g", "o", "v", "y"}
9      c = {"d", "e", "j", "k"}
10     d = {"a", "b", "f", "g"}
11
12     x = (a.intersection(b)).union(c)
13     print(f"x = {x}")
14
15     # Найдем дополнения множеств
16     bn = u.difference(b)
17     cn = u.difference(c)
18     y = (a.difference(d)).union(cn.difference(bn))
19     print(f"y = {y}"]
```

Рисунок 5 – пример 1

```
C:\Users\User\AppData\Local\Programs\Python
x = {'j', 'k', 'o', 'e', 'd'}
y = {'g', 'c', 'f', 'v', 'o', 'h', 'y'}

Process finished with exit code 0
```

Рисунок 6 – пример выполнения примера 1

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

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      input_string = input("Введите строку: ")
6      input_string = input_string.lower()
7      vowels = set("aeiou")
8
9      # Найдём сумму
10     count = sum(1 for char in input_string if char in vowels)
11     print(f"Количество гласных в строке: {count}")

```

Рисунок 7 – задание 9

```

C:\Users\User\AppData\Local\Programs
Введите строку: esfoijrsfjnkjsk
Количество гласных в строке: 3

Process finished with exit code 0

```

Рисунок 8 – пример выполнения 9 задания

8. Определите общие символы в двух строках, введенных с клавиатуры.

```

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

if __name__ == "__main__":
    a = set(input("Введите 1 строку: "))
    b = set(input("Введите 2 строку: "))
    c = a.intersection(b)

    print(f"Общие элементы у первой и второй строк: {c}")

```

Рисунок 9 – выполнение задания 11

```
C:\Users\User\AppData\Local\Programs\Python\Python311\python.  
Введите 1 строку: yaroslav  
Введите 2 строку: gurtovoy  
Общие элементы у первой и второй строк: {'o', 'y', 'v', 'r'}  
  
Process finished with exit code 0
```

Рисунок 10 – результат выполнения задания 11

1. Индивидуальное задание. Определить результат выполнения операций над множествами. Считать элементы множества строками. Проверить результаты вручную.

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
  
if __name__ == "__main__":  
    # Определим универсальное множество  
    u = set("abcdefghijklmnopqrstuvwxyz")  
    a = {"a", "d", "k", "l", "o", "s"}  
    b = {"d", "e", "k", "s", "u", "x"}  
    c = {"o", "p", "w"}  
    d = {"d", "n", "r", "y", "z"}  
  
    x = (a.difference(b)).union(c.intersection(d))  
    print(f"x = {x}")  
  
    da = u.difference(a)  
    db = u.difference(b)  
    y = (da.intersection(db)).difference(c.union(d))  
    print(f"y = {y}")
```

Рисунок 11 – выполнение индивидуального задания

```
C:\Users\User\AppData\Local\Programs\Python\Python311\python
x = {'o', 'a', 'l'}
y = {'b', 'h', 'm', 'j', 'f', 'v', 'i', 't', 'g', 'q', 'c'}

Process finished with exit code 0
```

Рисунок 12 – результат выполнения индивидуального задания

9.Зафиксировал все изменения в github в ветке develop.

```
PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr_2.7\lr-2.7> git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   pycharm/.idea/.gitignore
    new file:   pycharm/.idea/inspectionProfiles/profiles_settings.xml
    new file:   pycharm/.idea/lr 2.7.iml
    new file:   pycharm/.idea/misc.xml
    new file:   pycharm/.idea/modules.xml
    new file:   pycharm/ex1.py
    new file:   pycharm/my_task.py
    new file:   pycharm/task1.py
    new file:   pycharm/task2.py

PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr_2.7\lr-2.7> git commit -m "laba"
[develop dd5dcc8] laba
 9 files changed, 86 insertions(+)
 create mode 100644 pycharm/.idea/.gitignore
 create mode 100644 pycharm/.idea/inspectionProfiles/profiles_settings.xml
 create mode 100644 pycharm/.idea/lr 2.7.iml
 create mode 100644 pycharm/.idea/misc.xml
 create mode 100644 pycharm/.idea/modules.xml
 create mode 100644 pycharm/ex1.py
 create mode 100644 pycharm/my_task.py
 create mode 100644 pycharm/task1.py
 create mode 100644 pycharm/task2.py
PS C:\Users\User\Desktop\учеба\Зсемак\змії\lr 2.7\lr-2.7>
```

Рисунок 13 – фиксация изменений в ветку develop

10.Слил ветки.

```

PS C:\Users\User\Desktop\учеба\Зсемак\эми\lr_2.7\lr-2.7> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\User\Desktop\учеба\Зсемак\эми\lr_2.7\lr-2.7> git merge develop
Updating 1a6cccf..dd5dcc8
Fast-forward
 pycharm/.idea/.gitignore           | 3 +++
 .../.idea/inspectionProfiles/profiles_settings.xml | 6 ++++++
 pycharm/.idea/lr 2.7.iml           | 8 ++++++++
 pycharm/.idea/misc.xml             | 4 ++++
 pycharm/.idea/modules.xml          | 8 ++++++++
 pycharm/ex1.py                     | 19 +++++++++++++++++++++
 pycharm/my_task.py                  | 18 ++++++++++++++++++++
 pycharm/task1.py                    | 11 ++++++++
 pycharm/task2.py                    | 9 ++++++++
9 files changed, 86 insertions(+)
create mode 100644 pycharm/.idea/.gitignore
create mode 100644 pycharm/.idea/inspectionProfiles/profiles_settings.xml
create mode 100644 pycharm/.idea/lr 2.7.iml
create mode 100644 pycharm/.idea/misc.xml
create mode 100644 pycharm/.idea/modules.xml
create mode 100644 pycharm/ex1.py
create mode 100644 pycharm/my_task.py
create mode 100644 pycharm/task1.py
create mode 100644 pycharm/task2.py
PS C:\Users\User\Desktop\учеба\Зсемак\эми\lr_2.7\lr-2.7>

```

Рисунок 14 – сливание ветки develop в ветку main

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

Контрольные вопросы:

1. Что такое множества в языке Python?

Множеством в языке программирования Python называется неупорядоченная совокупность уникальных значений. В качестве элементов этого набора данных могут выступать любые неизменяемые объекты, такие как числа, символы, строки. В отличие от массивов и списков, порядок следования значений не учитывается при обработке его содержимого. Над одним, а также несколькими множествами можно выполнять ряд операций, благодаря функциям стандартной библиотеки языка программирования Python.

2. Как осуществляется создание множеств в Python?

Сделать это можно, просто присвоив переменной последовательность значений, выделив их фигурными скобками. Существует и другой способ создания множеств, который подразумевает использование вызова `set`. Аргументом этой функции может быть набор неких данных или даже строка с текстом.

3. Как проверить присутствие/отсутствие элемента в множестве? Для этого используется `in`.

4. Как выполнить перебор элементов множества? `for a in {0, 1, 2}:
print(a)`

5. Что такое set comprehension?

Чтобы узнать, является ли множество `a` надмножеством `b`, необходимо вызвать метод `issuperset` и вывести результат его работы на экран.

1. Каково назначение множеств `frozenset` ? `frozenset` в Python - это неизменяемая (`immutable`) версия типа данных "множество" (`set`). Основное назначение `frozenset` заключается в том, что оно может использоваться в

ситуациях, где требуется неизменяемое множество, то есть множество, элементы которого нельзя изменить после его создания. Вот некоторые случаи, когда `frozenset` может быть полезным:

- Ключи в словаре: Поскольку словари Python могут использовать только неизменяемые объекты в качестве ключей, `frozenset` может быть использован в качестве ключа для словаря.
- Элементы множества в другом множестве: Вы можете создать множество, содержащее `frozenset`, чтобы использовать его в качестве элемента другого множества, так как `frozenset` является неизменяемым и поэтому может быть элементом множества.
- Защита от изменений: Если вам нужно гарантировать, что набор элементов останется неизменным и не будет изменен случайно или намеренно, вы можете использовать `frozenset` вместо `set`.

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

Для преобразования множества в строку используется конкатенация текстовых значений, которую обеспечивает функция `join`. В этом случае ее аргументом является набор данных в виде нескольких строк. Запятая в кавычках выступает в качестве символа, разделяющего значения.

Чтобы получить из множества словарь, следует передать функции `dict` набор из нескольких пар значений, в каждом из которых будет находиться ключ.

По аналогии с предыдущими преобразованиями можно получить список неких объектов. На этот раз используется вызов `list`.

