

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №11**  
**дисциплины «Основы программной инженерии»**

Выполнил:  
Яблоновский Дмитрий Николаевич  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка и  
сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Руководитель практики:  
Богданов С.С., ассистент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** Работа с множествами в языке Python.

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

### Порядок выполнения работы

1. Создал репозиторий GitHub.

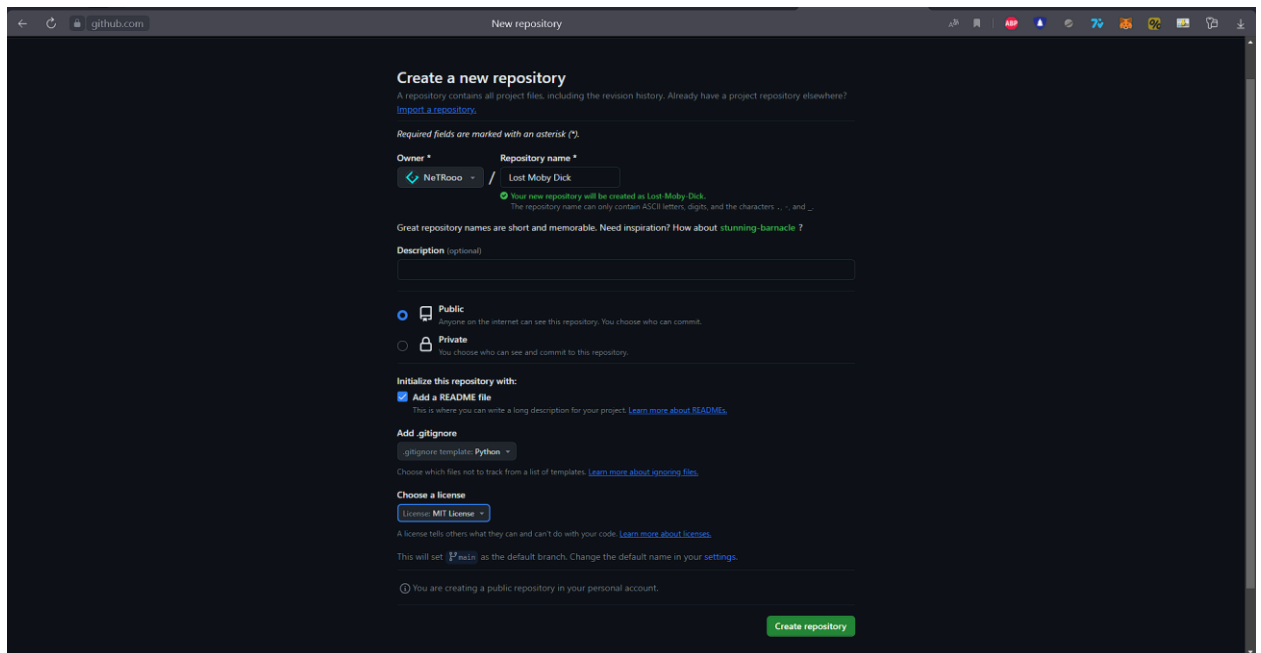


Рисунок 1- Создание репозитория

2. Проработал примеры из лабораторной работы.

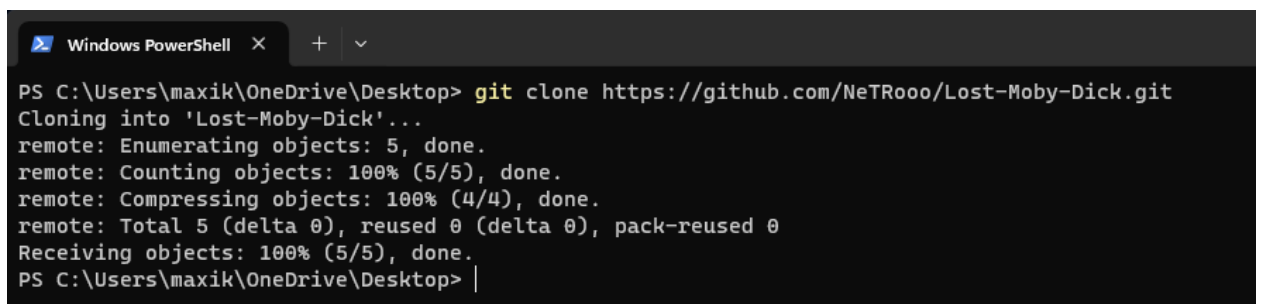


Рисунок 2 – Выполнил клонирование созданного репозитория

Код программы из примера:

```
#!/usr/bin/env python3

# -*- coding: utf-8 -*-

import sys

from datetime import date

def get_worker():
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))
    # Создать словарь.
    return {
        'name': name,
        'post': post,
        'year': year,
    }

def display_workers(staff):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
```

if staff:

# Заголовок таблицы.

line = '+--{ }--+{ }--+{ }--+{ }--+'.format(

'-' \* 4,

'-' \* 30,

'-' \* 20,

'-' \* 8

)

print(line)

print(

'| {:^4} | {:^30} | {:^20} | {:^8} |'.format(

"№",

"Ф.И.О.",

"Должность",

"Год"

)

)

print(line)

# Вывести данные о всех сотрудниках.

for idx, worker in enumerate(staff, 1):

print(

'| {:>4} | {:<30} | {:<20} | {:>8} |'.format(

idx,

```

        worker.get('name', ""),

        worker.get('post', ""),

        worker.get('year', 0)

    )

)

print(line)

else:

    print("Список работников пуст.")

def select_workers(staff, period):

    """

    Выбрать работников с заданным стажем.

    """

    # Получить текущую дату.

    today = date.today()

    # Сформировать список работников.

    result = []

    for employee in staff:

        if today.year - employee.get('year', today.year) >= period:

            result.append(employee)

    # Возвратить список выбранных работников.

    return result

```

```
def main():  
    """  
    Главная функция программы.  
    """  
    # 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.  
            worker = get_worker()  
            # 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':  
            # Отобразить всех работников.
```

```

display_workers(workers)

elif command.startswith('select '):

    # Разбить команду на части для выделения стажа.

    parts = command.split(' ', maxsplit=1)

    # Получить требуемый стаж.

    period = int(parts[1])

    # Выбрать работников с заданным стажем.

    selected = select_workers(workers, period)

    # Отобразить выбранных работников.

    display_workers(selected)

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)

if __name__ == '__main__':

    main()

```

```
ПРОБЛЕМЫ 102 Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ GITLENS

PS C:\Users\maxik\OneDrive\Desktop\Lost-Moby-Dick> & C:/Users/maxik/AppData/Local/Programs/Python/Python39/python.exe c:/Users/maxik/OneDrive/Desktop/Lost-Moby-Dick/task0.py
>>> рудз
Неизвестная команда рудз
>>> help
Список команд:

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

Рисунок 3 – Выполнение кода из примера.

```
task1.py U X
task1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  def test():
4      num = int(input('Введите целое число: '))
5      if num > 0:
6          positive()
7      elif num < 0:
8          negative()
9  def positive():
10     print('Положительное')
11 def negative():
12     print('Отрицательное')
13 if __name__ == '__main__':
14     test()

ПРОБЛЕМЫ 5 Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ GITLENS

PS C:\Users\maxik\OneDrive\Desktop\Lost-Moby-Dick> & C:/Users/maxik/AppData/Local/Programs/Python/Python39/python.exe c:/Users/maxik/OneDrive/Desktop/Lost-Moby-Dick/task1.py
Введите целое число: 13
Положительное
PS C:\Users\maxik\OneDrive\Desktop\Lost-Moby-Dick> █
```

Рисунок 4 – Код задания и вывод при выполнении. (задание №1)

```
task1.py U task2.py U X
task2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  def multiply_until_zero():
4      result = 1
5      while True:
6          num = int(input('Введите число (для остановки введите 0): '))
7          if num == 0:
8              break
9          result *= num
10     return result
11 if __name__ == '__main__':
12     result = multiply_until_zero()
13     print(f'Произведение введенных чисел: {result}')

ПРОБЛЕМЫ 12 Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ GITLENS

PS C:\Users\maxik\OneDrive\Desktop\Lost-Moby-Dick> & C:/Users/maxik/AppData/Local/Programs/Python/Python39/python.exe c:/Users/maxik/OneDrive/Desktop/Lost-Moby-Dick/task2.py
Введите число (для остановки введите 0): 5
Введите число (для остановки введите 0): 6
Введите число (для остановки введите 0): 0
Произведение введенных чисел: 30
PS C:\Users\maxik\OneDrive\Desktop\Lost-Moby-Dick> █
```

Рисунок 5 – Код задания и вывод при выполнении. (задание №2)



```
task3.py U task2.py U task3.py U X
task3.py > ...
1  #!/usr/bin/env python3
2  #-*- coding: utf-8 -*-
3  import math
4  def circle(radius):
5      return math.pi * radius ** 2
6  def cylinder():
7      radius = float(input('Введите радиус цилиндра: '))
8      height = float(input('Введите высоту цилиндра: '))
9      side_area = 2 * math.pi * radius * height
10     full_area = side_area + 2 * circle(radius)
11     choice = input('Хотите получить только боковую площадь? (да/нет): ').lower()
12     if choice == 'да':
13         return side_area
14     elif choice == 'нет':
15         return full_area
16     else:
17         return 'Неправильный выбор.'
18 if __name__ == '__main__':
19     result = cylinder()
20     print(f'Площадь цилиндра: {result}')

ПРОБЛЕМЫ (27) ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ GITLENS
PS C:\Users\maxik\OneDrive\Desktop\Lost-Moby-Dick> & C:\Users\maxik\AppData\Local\Programs\Python\Python39\python.exe c:/Users/maxik/OneDrive/Desktop/Lost-Moby-Dick/task3.py
Введите радиус цилиндра: 12
Введите высоту цилиндра: 5
Хотите получить только боковую площадь? (да/нет): нет
Площадь цилиндра: 1281.7698026646356
PS C:\Users\maxik\OneDrive\Desktop\Lost-Moby-Dick>
```

Рисунок 6 – Код задания и вывод при выполнении. (задание №3)

```
task1.py U task2.py U task3.py U task4.py X
task4.py > ...
1  #!/usr/bin/env python3
2  #-*- coding: utf-8 -*-
3  def get_input():
4      return input("Введите значение: ")
5  def test_input(value):
6      try:
7          int(value)
8          return True
9      except ValueError:
10         return False
11  def str_to_int(value):
12      return int(value)
13  def print_int(number):
14      print(number)
15  if __name__ == '__main__':
16      user_value = get_input()
17      if test_input(user_value):
18          int_value = str_to_int(user_value)
19          print_int(int_value)
20      else:
21          print("Введенное значение нельзя преобразовать в целое число.")

ПРОБЛЕМЫ (35) ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ GITLENS
PS C:\Users\maxik\OneDrive\Desktop\Lost-Moby-Dick> & C:\Users\maxik\AppData\Local\Programs\Python\Python39\python.exe c:/Users/maxik/OneDrive/Desktop/Lost-Moby-Dick/task4.py
Введите значение: 1923
1923
PS C:\Users\maxik\OneDrive\Desktop\Lost-Moby-Dick>
```

Рисунок 7 – Код задания и вывод при выполнении. (задание №4)

```
PS C:\Users\vmazik\OneDrive\Desktop\Lost-Moby-Dick> git add .
warning: in the working copy of 'individual_task1.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'lab_task1.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'task1.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'task2.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'task3.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'task4.py', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\vmazik\OneDrive\Desktop\Lost-Moby-Dick> git commit -m "last commit"
[dev 5b67615] last commit
 7 files changed, 398 insertions(+)
 create mode 180644 individual_task1.py
 create mode 180644 lab_task1.py
 create mode 180644 task0.py
 create mode 180644 task1.py
 create mode 180644 task2.py
 create mode 180644 task3.py
 create mode 180644 task4.py
PS C:\Users\vmazik\OneDrive\Desktop\Lost-Moby-Dick> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\vmazik\OneDrive\Desktop\Lost-Moby-Dick> git merge dev
Updating 64b28a1..5b67615
Fast-forward
 individual_task1.py | 95 +++++
 lab_task1.py       | 122 +++++
 task0.py           | 113 +++++
 task1.py           | 14 +++++
 task2.py           | 13 +++++
 task3.py           | 20 +++++
 task4.py           | 21 +++++
 7 files changed, 398 insertions(+)
 create mode 180644 individual_task1.py
 create mode 180644 lab_task1.py
 create mode 180644 task0.py
 create mode 180644 task1.py
 create mode 180644 task2.py
 create mode 180644 task3.py
 create mode 180644 task4.py
PS C:\Users\vmazik\OneDrive\Desktop\Lost-Moby-Dick> git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 4.40 KiB | 4.40 MiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/NeTRooo/Lost-Moby-Dick.git
 64b28a1..5b67615  main -> main
PS C:\Users\vmazik\OneDrive\Desktop\Lost-Moby-Dick> git checkout dev
Switched to branch 'dev'
Your branch is ahead of 'origin/dev' by 1 commit.
(use "git push" to publish your local commits)
PS C:\Users\vmazik\OneDrive\Desktop\Lost-Moby-Dick> git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/NeTRooo/Lost-Moby-Dick.git
```

Рисунок 8 – Создание коммита и отправка коммита.

Вывод: в ходе выполнения лабораторной работы были получены навыки работы с множествами в языке Python.

Ссылка на репозиторий: <https://github.com/NeTRooo/Lost-Moby-Dick>

## Ответы на контрольные вопросы

1. Что такое списки в языке Python?

Списки в языке Python - это упорядоченные изменяемые коллекции элементов.

2. Каково назначение кортежей в языке Python?

Кортежи в языке Python используются для создания неизменяемых упорядоченных коллекций элементов.

3. Как осуществляется создание кортежей?

Кортежи создаются с использованием круглых скобок, например, `my_tuple = (1, 2, 3)`.

4. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется по индексу, например, `element = my_tuple[0]`.

5. Зачем нужна распаковка (деструктуризация) кортежа?

Распаковка кортежа (деструктуризация) позволяет присваивать значения элементам кортежа одной строкой.

6. Какую роль играют кортежи в множественном присваивании?

Кортежи играют ключевую роль в множественном присваивании, где значения присваиваются сразу нескольким переменным.

7. Как выбрать элементы кортежа с помощью среза?

Выбор элементов кортежа с помощью среза осуществляется, например, `subset = my_tuple[1:3]`.

8. Как выполняется конкатенация и повторение кортежей?

Конкатенация кортежей выполняется оператором `+`, а повторение - оператором `*`.

9. Как выполняется обход элементов кортежа?

Обход элементов кортежа осуществляется с использованием цикла, например, `for item in my_tuple:`.

10. Как проверить принадлежность элемента кортежу.

Принадлежность элемента кортежу можно проверить с использованием оператора `in`

11. Какие методы работы с кортежами Вам известны?

Некоторые методы работы с кортежами включают `count()` для подсчета элементов и `index()` для поиска индекса элемента.

12. Допустимо ли использование функций агрегации таких как `len()`, `sum()` и т. д. при работе с кортежами?

Да, функции агрегации, такие как `len()`, `sum()`, могут использоваться с кортежами.

13. Как создать кортеж с помощью спискового включения?

Кортеж можно создать с помощью спискового включения, например, `my_tuple = tuple(x for x in my_list)`.