

## Отчёт по практической работе №4

1. Создан файл «lab4-1.py». Скопирован текст из файла «4.1.txt». Выполнены задания из комментариев.

### 1) Задание №1

```
squares = [1, 4, 9, 16, 25] # Создание нового списка
print(squares)
# Выведите на экран подсписок [4, 9, 16].
squares_2 = []
squares_2 = squares[1:4]
print(squares_2)
```

```
C:\Users\297\AppData\Local\Programs\Python\Python39\Scripts\python.exe lab4-1.py
[1, 4, 9, 16, 25]
[4, 9, 16]
```

### 2) Задание №2

```
animals = ['elephant', 'lion', 'tiger', 'giraffe'] # Создание нового списка
print(animals)
animals += ['monkey', 'dog'] # Добавление новых элементов в список с помощью операции конкатенации
print(animals)
animals.append('dino') # Добавление нового элемента с помощью метода append()
print(animals)
# Замените элемент 'dino' на 'dinosaur'
animals.remove('dino') # Удаление 'dino'
animals.append('dinosaur')
print(animals)
```

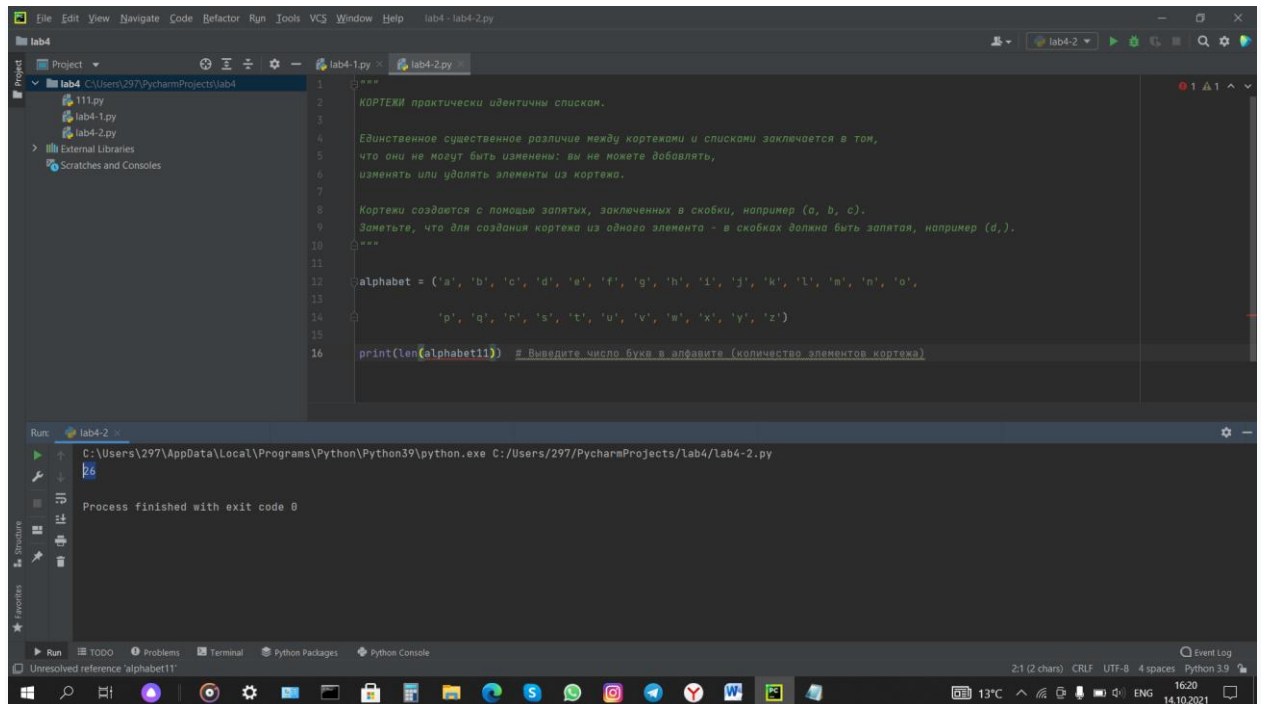
```
['elephant', 'lion', 'tiger', 'giraffe']
['elephant', 'lion', 'tiger', 'giraffe', 'monkey', 'dog']
['elephant', 'lion', 'tiger', 'giraffe', 'monkey', 'dog', 'dino']
['elephant', 'lion', 'tiger', 'giraffe', 'monkey', 'dog', 'dinosaur']
```

### 3) Задание №3

```
animals = ['elephant', 'lion', 'tiger', 'giraffe', 'monkey', 'dog'] # Создание нового списка
print(animals)
animals[1:3] = ['cat'] # Замена сразу двух элементов - 'lion' и 'tiger' - на один элемент - 'cat'
print(animals)
animals[1:3] = [] # Удаление элементов 'cat' и 'giraffe' из списка
print(animals)
# Теперь полностью очистите список
animals[0:3] = []
print(animals)
```

```
['elephant', 'lion', 'tiger', 'giraffe', 'monkey', 'dog']
['elephant', 'cat', 'giraffe', 'monkey', 'dog']
['elephant', 'monkey', 'dog']
[]
```

2. Создан файл «lab4-2.py». Скопирован текст из файла «4.2.txt». Выполнены задания из комментариев.

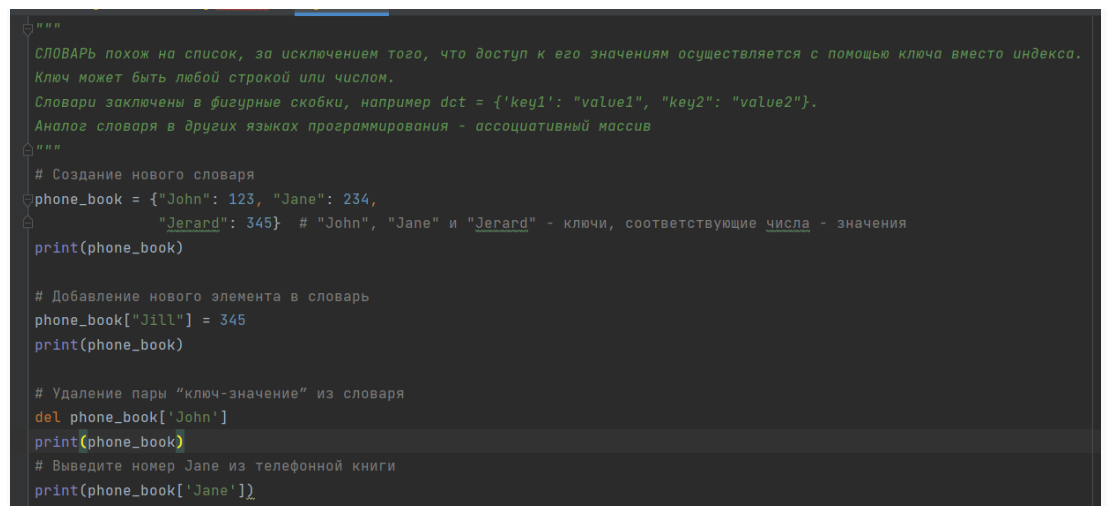


```
1 """
2 КОРТЕЖИ практически идентичны спискам.
3
4 Единственное существенное различие между кортежами и списками заключается в том,
5 что они не могут быть изменены: вы не можете добавлять,
6 изменять или удалять элементы из кортежа.
7
8 Кортежи создаются с помощью запяток, заключенных в скобки, например (a, b, c).
9 Заметьте, что для создания кортежа из одного элемента - в скобках должна быть запятая, например (d,).
10 """
11
12 alphabet = ('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
13            'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')
14
15
16 print(len(alphabet)) # Выведите число букв в алфавите (количество элементов кортежа)
```

Run: lab4-2 -  
C:\Users\297\AppData\Local\Programs\Python\Python39\python.exe C:/Users/297/PycharmProjects/Lab4/Lab4-2.py  
26  
Process finished with exit code 0

3. Создан файл «lab4-3.py». Скопирован текст из файла «4.3.txt». Выполнены задания из комментариев.

### 1) Задание №1



```
1 """
2 СЛОВАРЬ похож на список, за исключением того, что доступ к его значениям осуществляется с помощью ключа вместо индекса.
3 Ключ может быть любой строкой или числом.
4 Словари заключены в фигурные скобки, например dct = {'key1': "value1", "key2": "value2"}.
5 Аналог словаря в других языках программирования - ассоциативный массив
6 """
7
8 # Создание нового словаря
9 phone_book = {"John": 123, "Jane": 234,
10              "Jerard": 345} # "John", "Jane" и "Jerard" - ключи, соответствующие числам - значения
11
12 print(phone_book)
13
14 # Добавление нового элемента в словарь
15 phone_book["Jill"] = 345
16 print(phone_book)
17
18 # Удаление пары "ключ-значение" из словаря
19 del phone_book["John"]
20 print(phone_book)
21
22 # Выведите номер Jane из телефонной книги
23 print(phone_book["Jane"])
```

```
{'John': 123, 'Jane': 234, 'Jerard': 345}
{'John': 123, 'Jane': 234, 'Jerard': 345, 'Jill': 345}
{'Jane': 234, 'Jerard': 345, 'Jill': 345}
234
```

## 2) Задание №2

```
"""
В словарях есть много полезных методов, например keys() и values().
Вы можете изучить остальные, используя Ctrl + Пробел после имени dict_name с точкой.
"""
phone_book = {"John": 123, "Jane": 234, "Jerard": 345} # Создание нового словаря
print(phone_book)

# Добавление элемента с существующим ключом
phone_book["Jill"] = 456
print(phone_book) # Объясните результат

print(phone_book.keys())
# Выведите все номера из телефонной книги
print(phone_book.values())
```

```
{'John': 123, 'Jane': 234, 'Jerard': 345}
{'John': 123, 'Jane': 234, 'Jerard': 345, 'Jill': 456}
dict_keys(['John', 'Jane', 'Jerard', 'Jill'])
dict_values([123, 234, 345, 456])
```

## 3) Задание №3

```
"""
Ключевое слово in используется для проверки наличия в списке или словаре определенного элемента.
Вы можете применять к спискам или словарям так же, как со строками.
"""
grocery_list = ["fish", "tomato", 'apples'] # Создание нового списка
print("tomato" in grocery_list) # Проверка наличия элемента "tomato" в списке
grocery_dict = {"fish": 1, "tomato": 6, 'apples': 3} # Создание нового словаря
# Проверьте, есть ли в словаре запись с ключом "fish" и "potato"
print("fish" in grocery_dict)
print("potato" in grocery_dict)
```

```
True
True
False
```

4. Создан файл «lab4-4.py». Скопирован текст из файла «4.4.txt». Выполнены задания из комментариев.

### 1) Задание №1

```
"""
МНОЖЕСТВО в Python - это структура данных, содержащая не повторяющиеся элементы в случайном порядке.
"""
a = set() # Создание пустого множества
a.add(1) # Добавление элемента в множество
a.add(2)
a.add(1)
print(a) # вывелось два числа 1 и 2, несмотря на то, что мы дважды добавили 1 (строки 5 и 7)
b = set('hello') # Преобразование строки в множество
print(b) # выводятся буквы в случайном порядке, буква "l" выводится 1 раз
```

```
{1, 2}
{'e', 'o', 'l', 'h'}
```

### 2) Задание №2

```
"""
Множества удобно использовать для удаления повторяющихся элементов из списка
"""
a = ['aa', 'ab', 'aa', 'ba']
print(set(a))
"""
Множества поддерживают стандартные операции других структур данных - len, in, clear и т.п.
"""
# Вставьте пропущенную строку, чтобы оператор print выводил True
b = ['aa', 'ab', 'aa', 'ba']
b = set(b)
print(len(b) == 3)
```

```
{'ba', 'ab', 'aa'}
True
```

### 3) Задание №3

```
"""
Помимо базовых операций, множества в Python поддерживают операции математических множеств
"""
a = {1, 2, 3, 4} # Создание множества
b = {3, 4, 5, 6}
c = a.union(b) # Объединение множества
print(c)
# Вставьте пропущенное действие, чтобы в консоль вывелось пересечение множеств a и b
d = a.intersection(b)
print(d)
```

```
{1, 2, 3, 4, 5, 6}
{3, 4}
```

5. Создан файл «lab4-5.py». Написана программа, которая считывает с клавиатуры данные о 5 призывниках (Фамилия, Имя, Отчество, Год рождения, Заболевание) и выводит результат в виде таблицы (считывание и сохранение данных производится через словарь)

```
n_soldier = int(input("Введите количество призывников:"))

from prettytable import PrettyTable
table_s = PrettyTable()

table_s.field_names = ["Номер призывника", "Фамилия", "Имя", "Отчество", "Год рождения", "Заболевание"]
for i in range(1, n_soldier + 1):
    surname={}
    name = {}
    patronymic = {}
    birthday = {}
    ill = {}

    surname["Фамилия"] = input("Введите фамилию:")
    name["Имя"] = input("Введите имя:")
    patronymic["Отчество"] = input("Введите отчество:")
    birthday["Год рождения"] = input("Введите год рождения:")
    ill["Заболевание"] = input("Введите заболевание:")

    table_s.add_row([i, surname["Фамилия"], name["Имя"], patronymic["Отчество"], birthday["Год рождения"], ill["Заболевание"]])
print(table_s)
```

```
lab4-5 x
Введите имя:Александр
Введите отчество:Михайлович
Введите год рождения:2001
Введите заболевание:Стенокардия

+-----+-----+-----+-----+-----+-----+
| Номер призывника | Фамилия | Имя | Отчество | Год рождения | Заболевание |
+-----+-----+-----+-----+-----+-----+
| 1 | Иванов | Иван | Иванович | 1999 | Слепота |
| 2 | Петров | Николай | Сергеевич | 2001 | Нет заболеваний |
| 3 | Хомяков | Павел | Дмитриевич | 1998 | Нет заболеваний |
| 4 | Григоренко | Михаил | Андреевич | 2000 | Кардиомиопатия |
| 5 | Зиновьев | Александр | Михайлович | 2001 | Стенокардия |
+-----+-----+-----+-----+-----+-----+

Process finished with exit code 0
```