Практическая работа №3
Разработка программ с использованием словарей, кортежей и множеств.

Цель работы: закрепить знания принципов работы с кортежами и множествами, научиться разрабатывать и отлаживать программы обработки структурированных типов данных; рассмотреть принципы работы со словарями, научиться использовать методы словарей, овладеть практическими навыками разработки и отладки программ обработки словарей

Задание на лабораторную работу:

1. Изучить теоретические сведения.
2. Написать программу в соответствии с вариантом.

Теоретические сведения

Множества (set) – это неупорядоченные коллекции уникальных элементов. В отличие от последовательностей, множества не поддерживают операции индексирования и получения срезов. Элементы, помещаемые в множество, должны быть неизменяемыми (Например, числа, строки, кортежи. Изменяемые типы данных не могут быть элементами множества, в частности, нельзя сделать элементом множества список (но можно сделать кортеж) или другое множество).

Создание пустого множества:

`>>> a = set()`
`>>> a`
`set()`
	`>>> a = {}  # А так нельзя!`
	`будет создан пустой словарь`

Множества удобно использовать для удаления повторяющихся элементов:

```
>>> words = ['hello', 'daddy', 'hello', 'mum']
>>> set(words)
{'hello', 'daddy', 'mum'}
```

Существует два типа множеств: set – изменяемое множество, и frozenset – неизменяемое множество. Оба типа множеств создаются с помощью пары встроенных функций:

```
s = set([1,5,10,15])
f = frozenset(['a',37,'hello'])
```

Обе функции, set() и frozenset(), заполняют создаваемое множество, выполняя итерации по значениям, хранящимся в аргументе. Оба типа множеств поддерживают методы, перечисленные в таблице:

Методы и операторы, поддерживаемые множествами

Метод	Описание
len(s)	Возвращает количество элементов в множестве s.
s.copy()	Создает копию множества s.
s.difference(t)	Разность множеств. Возвращает все элементы из множества s, отсутствующие в t.
s.intersection(t)	Пересечение множеств. Возвращает все элементы, присутствующие в обоих множествах s и t.
s.isdisjoint(t)	Возвращает True, если множества s и t не имеют общих элементов.
s.issubset(t)	Возвращает True, если множество s является подмножеством t.

s.issuperset(t)	Возвращает True, если множество s является надмножеством t.
s.symmetric_difference(t)	Симметричная разность множеств. Возвращает все элементы, которые присутствуют в множестве s или t, но не в обоих сразу.
s.union(t)	Объединение множеств. Возвращает все элементы, присутствующие в множестве s или t.

Возвращаемое значение имеет тот же тип, что и множество s (set или frozenset). Аргумент t может быть любым объектом языка Python, поддерживающим итерации.

```
>>> a = set([1,2,3,4])
>>> b = set([3,4,5,6])
>>> a | b                    # Объединение
{1, 2, 3, 4, 5, 6}
>>> a & b                    # Пересечение
{3, 4}
>>> a < b                    # Подмножества
False
>>> a - b                    # Разность (элементы, присутствующие в t,
                            но отсутствующие в s)
{1, 2}
>>> a ^ b                    # Симметричная разность (элементы,
                            присутствующие в a или b,  но не в двух
                            множествах сразу)
{1, 2, 5, 6}
```

Дополнительно изменяемые множества (тип set) поддерживают методы, перечисленные в таблице:

Методы, поддерживаемые изменяемыми множествами

Метод	Описание
s.add(item)	Добавляет элемент item в s. Ничего не делает, если этот элемент уже имеется в множестве.
s.clear()	Удаляет все элементы из множества s.
s.difference_update(t)	Удаляет все элементы из множества s, которые присутствуют в t.
s.discard(item)	Удаляет элемент item из множества s. Ничего не делает, если этот элемент отсутствует в множестве.
s.intersection_update(t)	Находит пересечение s и t и оставляет результат в s.
s.pop()	Возвращает произвольный элемент множества и удаляет его из s.
s.remove(item)	Удаляет элемент item из s. Если элемент item отсутствует в множестве, возбуждается исключение KeyError.
s.discard (item)	Удаляет элемент item из s. Если элемент item отсутствует в множестве, не делает ничего
s.symmetric_difference_ update(t)	Находит симметричную разность s и t и оставляет результат в s.
s.update(t)	Добавляет все элементы t в множество s. t может быть другим множеством, последовательностью или любым другим объектом, поддерживающим итерации.

Все эти операции изменяют само множество s. Аргумент t может быть любым объектом, поддерживающим итерации.

Мощность множества показывает количество элементов в нем. По факту, это аналог функции len() для других коллекций.

Кортежи (tuple) – неизменяемый тип данных

!!!! Обратите внимание: кортежи не обладают методами, которые имеются у списков (например, кортежи не имеют метода append). Зато кортежи поддерживают обычные операции над последовательностями, которые применяются к строкам и к спискам:

```
>>> (1, 2) + (3, 4)          # Конкатенация (1, 2, 3, 4)
>>> (1, 2) * 4               # Повторение (1, 2, 1, 2, 1, 2, 1, 2)
>>> T = (1, 2, 3, 4)         # Индексирование, извлечение среза
>>> T[0], T[1:3]
(1, (2, 3))
```

Операции +, * и извлечения среза при применении к кортежам возвращают новые кортежи.

Операции над кортежами:

Операция	Интерпретация
()	Пустой кортеж
T = (0,)	Кортеж из одного элемента
T = (0, ‘Ni’, 1.2, 3) или T = 0, ‘Ni’, 1.2, 3	Кортеж из четырех элементов
T = (‘abc’, (‘def’, ‘ghi’))	Вложенные кортежи
T = tuple(‘spam’)	Создание кортежа из итерируемого объекта
T[i] T[i][j]	Индексы
T[i:j]	Срез
len(T)	Длина
T1 + T2	Конкатенация
T * 3	Повторение
for x in T: print(x)	Обход в цикле
‘spam’ in t2	проверка вхождения
T.count(‘Ni’)	подсчет вхождений (в версии 3.0)
T.index(‘Ni’)	поиск (в версии 3.0)

list и ***tuple*** – это встроенные функции, которые используются для преобразования в список и затем обратно в кортеж. В действительности обе функции создают новые объекты, но благодаря им создается эффект преобразования.

Пример: необходимо отсортировать содержимое кортежа. Для этого его сначала можно преобразовать в список, чтобы превратить в изменяемый объект и получить доступ к методу сортировки или задействовать функцию sorted, которая принимает объекты любых типов последовательностей (и не только):

```
>>> T = (‘cc’, ‘aa’, ‘dd’, ‘bb’)
>>> tmp = list(T)            # Создать список из элементов кортежа
>>> tmp.sort()               # Отсортировать списка
>>> tmp
[‘aa’, ‘bb’, ‘cc’, ‘dd’]
>>> T = tuple(tmp)           # Создать кортеж из элементов списка
>>> T
```

```
('aa', 'bb', 'cc', 'dd')
>>> sorted(T)              # Или использовать встроенную функцию
sorted
```

Кортежи могут содержать списки:

```
>>> nested = (1, "do", ["param", 10, 20])
```

Можно изменить список:

```
>>> nested[2][1] = 15
>>> nested
(1, 'do', ['param', 15, 20])
```

!!!! В кортеже содержится не сам список, а ссылка на него. Ее изменить нельзя. Но менять сам список можно.

Именованные кортежи

Пример: Создайте кортеж из 7-ми именованных кортежей учащихся. В именованном кортеже будут присутствовать следующие поля: имя студента, возраст, оценка за семестр, город проживания. Функция good_students() будет принимать этот кортеж, вычислять среднюю оценку по всем учащимся и выводить на печать следующее сообщение: “Ученики {список имен студентов через запятую} в этом семестре хорошо учатся!”. В список студентов, которые выводятся по результатам работы функции, попадут лишь те, у которых оценка за семестр равна или выше средней.

```
from collections import namedtuple
Student = namedtuple('Student', 'name age mark city')
students = (
   Student('Елена', '13', 7.1, 'Москва'),
   Student('Ольга', '11', 7.9, 'Иваново'),
   Student('Елизавета', '14', 9.1, 'Тверь'),
   Student('Дмитрий', '12', 5.2, 'Челябинск'),
   Student('Максим', '15', 6.1, 'Самара'),
   Student('Николай', '11', 8.7, 'Владивосток'),
   Student('Артур', '13', 5.8, 'Екатеринбург')
)

def good_students(students):
   total_mark = 0

   for student in students:
       total_mark += student.mark
   avg_mark = total_mark / len(students)

   good_mark_students = [student.name for student in students if
student.mark >= avg_mark]
   print('Ученики ', ', '.join(good_mark_students), ' в этом семестре
хорошо учатся!')

good_students(students)
```

Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному, называется **словарем** *(*dict) или *ассоциативным массивом*. Она позволяет хранить пары «ключ:значение».

!!!! *В словаре не может быть двух элементов с одинаковыми ключами. Могут быть одинаковые значения у разных ключей.*

Ключом может быть любой неизменяемый тип данных. Значением – любой тип данных.

```
>>> nums = {1: 'one', 2: 'two', 3: 'three'}
>>> person = {'name': 'Tom', 1: [30, 16], 2: 2.34, ('ab', 100): 'no'}
```

Добавление и изменение элемента имеет одинаковый синтаксис:
словарь[ключ] = значение.
Ключ может быть как уже существующим (тогда происходит изменение значения), так и новым (происходит добавление элемента словаря).
Пример: англо-русский словарь

```
# создание словаря
>>> a = {'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse':
'мышь'}
>>> a['cat']                    # вывод элемента словаря с ключом cat
'кошка'
>>> a                           # вывод всего словаря
{'dog': 'собака', 'cat': 'кошка', 'bird': 'птица', 'mouse': 'мышь'}
>>> a['elephant'] = 'бегемот'   # добавление нового элемента
>>> a['table'] = 'стол'         # добавление нового элемента
>>> a
{'dog': 'собака', 'cat': 'кошка', 'mouse': 'мышь', 'bird': 'птица',
'table': 'стол', 'elephant': 'бегемот'}
>>> a['elephant'] = 'слон' # изменение значения уже существующего
ключа
>>> del a['table']         # удаление элемента с ключом table
>>> a
{'dog': 'собака', 'cat': 'кошка', 'mouse': 'мышь', 'bird': 'птица',
'elephant': 'слон'}
```

Для ***удаления элемента*** из словаря можно использовать операцию del A[key] (операция возбуждает исключение KeyError, если такого ключа в словаре нет).

```
Пример безопасных способов удаления элемента из словаря.
A = {'ab' : 'ba', 'aa' : 'aa', 'bb' : 'bb', 'ba' : 'ab'}
key = 'ac'
```

# предварительно проверяем наличие элемента 	
# способ 1 	
if key in A:
  del A[key]
	# перехватываем и обрабатываем исключение
	# способ 2
	try:
	  del A[key]
	except KeyError:
	  print('Нет такого элемента')
	print(A)

Еще один способ удалить элемент из словаря: использование метода pop: A.pop(key). Этот метод возвращает значение удаляемого элемента, если элемент с данным ключом отсутствует в словаре, то возбуждается исключение. Если методу pop передать второй параметр, то если элемент в словаре отсутствует, то метод pop возвратит значение этого параметра. Это позволяет проще всего организовать безопасное удаление элемента из словаря: A.pop(key, None).

Перебор элементов словаря в цикле for

```
nums = {1: 'one', 2: 'two', 3: 'three'}
```

# простой перебор в цикле 	
for i in nums:
  print(i, nums[i])
	# с помощью метода items()
	for key, value in nums.items():
	  print(key, value)

```
#получим:
1 one
2 two
3 three
```

Методы словарей

dict.clear()	очищает словарь (удаляет все элементы словаря, но не удаляет сам словарь).
dict.copy()	возвращает копию словаря.
dict.get(key[, default])	возвращает значение ключа, но если его нет, не вызывает исключение, а возвращает default (по умолчанию None).
dict.items()	возвращает пары (ключ, значение).
dict.keys()	возвращает ключи в словаре.
dict.values()	возвращает значения в словаре.
dict.pop(key[, default])	удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение).
dict.popitem()	удаляет и возвращает пару (ключ, значение). Если словарь пуст, вызывает исключение KeyError. Помните, что словари неупорядочены.
dict.setdefault(key[, default])	возвращает значение ключа, но если его нет, не вызывает исключение, а создает ключ с значением default (по умолчанию None).
dict.update([other])	обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).
dict.fromkeys(seq[, value])	создает словарь с ключами из seq и значением value (по умолчанию None).

Пример создания копии словаря:

```
>>> nums2 = nums.copy()
>>> nums2[4] = 'four'
>>> nums
{1: 'one', 2: 'two', 3: 'three'}
>>> nums2
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

Пример: можно добавить в словарь другой словарь с помощью update():

```
>>>nums = {1: 'one', 2: 'two', 3: 'three'}
>>> nums.update({6: 'six', 7: 'seven'})
>>> nums
{1: 'one', 2: 'two', 3: 'three', 6: 'six', 7: 'seven'}
```

Пример: телефонный справочник

```
>>> tel = {'jack': 4098, 'sape': 4139}     #создание словаря из 2
эл-тов
>>> tel['guido'] = 4127                    #добавление нового
элемента
>>> tel                                    # вывод всего словаря
{'jack': 4098, 'sape': 4139, 'guido': 4127}
>>> tel['jack']                            # вывод элемента с ключом
jack
4098
>>> del tel['sape']                        #удаление элемента с ключом
sape
>>> tel['irv'] = 4127                      #добавление нового
элемента
>>> tel
{'jack': 4098, 'guido': 4127, 'irv': 4127}
>>> list(tel)                              #получение списка ключей
```

```
['jack', 'guido', 'irv']
>>> sorted(tel)                                #сортировка ключей словаря
['guido', 'irv', 'jack']
>>> 'guido' in tel                             #проверка наличия ключа в
словаре
True
```

Пример: функция biggest_dict принимает неограниченное количество параметров «ключ: значение» и обновляет созданный словарь my_dict, состоящий из одного элемента «first_one» со значением «we can do it».

```
my_dict = {'first_one': 'we can do it'}

def biggest_dict(**kwargs):
    my_dict.update(**kwargs)

biggest_dict(k1=22, k2=31, k3=11, k4=91)
biggest_dict(name='Елена', age=31, weight=61, eyes_color='grey')
print(my_dict)
Получим:
{'first_one': 'we can do it', 'k1': 22, 'k2': 31, 'k3': 11, 'k4': 91,
'name': 'Елена', 'age': 31, 'weight': 61, 'eyes_color': 'grey'}
```

Задания

Задание 1

Создайте кортеж из 7-ми именованных кортежей товаров. В именованном кортеже будут присутствовать следующие поля: наименование товара, производитель, вес, цена.

В1-4: Пользователь вводит вес. Функция принимает этот кортеж в качестве параметра, выводить на печать наименование товаров, вес которых равен или больше, чем ввел пользователь.

В5-8: Пользователь вводит цену. Функция принимает этот кортеж в качестве параметра, выводить на печать наименование товаров и производителей, цена которых меньше или равна, чем ввел пользователь.

В9-12: Пользователь вводит букву. Функция принимает этот кортеж в качестве параметра, выводить на печать наименование товаров, производители которых начинаются на заданную букву.

Задание 2

В1-В2: Заполните один кортеж десятью случайными целыми числами от 0 до 5 включительно. Также заполните второй кортеж числами от -5 до 0. Объедините два кортежа с помощью оператора +, создав тем самым третий кортеж. С помощью метода кортежа count() определите в нем количество нулей. Выведите на экран третий кортеж и количество нулей в нем.

В3-В4: Пользователь вводит строку слов чисел через пробел. Для каждого слова выведите в отдельной строке сообщение YES, если это слово уже встречалось в строке или NO, если не встречалось.

```
Пример: дана строка – кот компот мармелад компот шоколад
шоколад
получили –
no
no
no
yes
no
yes
```

В5-В6: Дан кортеж и элемент X. Создать новый кортеж, начинающийся с первого появления элемента X в нем и заканчивающийся вторым его появлением включительно.
Если элемента нет – создать пустой кортеж. Если элемент встречается только один раз, то создать кортеж, который начинается с него и идет до конца исходного.
```
Пример:  кортеж=(1, 8, 3, 4, 8, 8, 9, 2), X=8
получили (8, 3, 4, 8)
```

В7-В8: Даны два списка слов. Посчитайте, сколько слов содержится одновременно как в первом списке, так и во втором.
```
Пример: дан 1 список ['кот', 'компот', 'мармелад', 'шоколад']
дан 2 список ['мармелад', 'сироп', 'шоколад']
получили 2 слова
```

В9-В10: Дан список целых чисел. Написать программу создания кортежа из уникальных элементов списка в обратном порядке. Для получения обратного порядка можно использовать встроенную функцию reversed().
```
Пример: дан список [2, 1, 3, 1, 2, 5, 5, 9, 2, 0, 0]
получили кортеж (0, 2, 9, 5, 1, 3)
```

В11-В12: Дан список чисел. Определите, сколько в нем встречается различных чисел. Создайте множество, состоящее из половины различных чисел списка.
```
Пример: дан список [1, 6, 1, 2, 5, 5, 3, 1, 4,  2]
получили 6 различных чисел в списке, новое множество {1, 6, 2}
```

Задание 3

В1: Дан словарь `{'a':11, 'e':22, 'i':33, 'o':44, 'u':55, 'y':66}`. Пользователь вводит строку. Напишите функцию, которая заменяет все гласные в строке на числа из словаря.

В2: Дан словарь `tel={'Иванов':25689500, 'Петров':51245874, 'Сидоров': 56112780,'Соколов':40125870, 'Петренко':70015784, 'Данилов': 21124130}`. Пользователь вводит фамилию. Функция находит есть ли такой человек в справочнике и выводит его телефонный номер.

В3: Пользователь вводит фамилии (ключи словаря) и адрес (значения). Вывести информацию о тех, чья фамилия начинается на букву, заданную пользователем, с помощью функции.

В4: Написать функцию, которая находит ключ с самыми высокими и самыми малыми значениями в словаре `my_dict = {'a':500, 'b':5874, 'c': 560,'d':400, 'e':74, 'f': 20}`

В5: Пользователь вводит 5 названий товаров (ключи словаря) и их стоимость (значения). Вывести все данные в табличном виде с помощью функции.

В6: Перевести дату, введенную в виде «Название_месяца, число» в числовой вид с помощью функции. Например, ввели «Январь, 15», получили «01.15»

В7: Пользователь вводит строку. Напишите функцию, которая считает сколько раз встречается каждая гласная буква (a, e, i, o, u, y).

В8: Пользователь вводит строку цифр. Дан словарь, в котором ключами являются цифры, а значениями – коды графически видоизмененных цифр из таблицы Unicode. Написать функцию перевода цифр из обычных в видоизмененные.
```
D={0: 9471, 1:10102, 2:10103, 3: 10104, 4:10105, 5:10106,
6:10107, 7:10108,  8:10109, 9:10110}
```
❶❷❸❹❺❻❼❽❾❿

В9: Дан словарь, содержащий англо-русские слова (например, `{'cat': 'кошка', 'dog': 'собака'}`). Пользователь вводит английское слово. Напишите функцию, которая переводит слова на русский язык. Если такого слова нет в словаре, вместо него вывести ???

В10: Дана строка текста. Посчитать сколько раз каждое слово встречается в тексте. Написать функцию, которая выводит слова, встречающиеся только 3 раза в том порядке, как они идут в тексте.

В11: Пользователь вводит N слов, которые будут ключами словаря. Значения формируются случайным образом от 1 до 10. Вывести те элементы словаря, значения которых больше 5. Для решения использовать функцию.

В12: Дана строка чисел (от 0 до 9). Написать функцию, которая считает сколько раз каждое число встречается в строке, вывести наиболее часто встречающееся число. Например, дана строка '111411511111222'. Получим 1

В13: Задайте словарь для перевода слов в цифры (например, `{'one':1, 'two':2, 'three':3 …}` Пользователь вводит слова в строку через пробел. Напишите функцию, которая заменяет слова цифрами (например, ввели «two five nine», получили 259)

Задание 4

Создайте словарь, который будет хранить информацию о количестве учащихся в разных классах школы (1а, 1б, 2б, 6а, 7в и т. п.). Пользователь должен иметь возможность вносить изменения:

а) изменять количество учащихся в классе,

б) в школе появился новый класс (добавлять класс),

в) в школе был расформирован класс (удалять класс),

г) вычислить общее количество учащихся в школе.

Предусмотреть возможность многократного выполнения перечисленных действий.

Задание 5

Пользователь вводит количество стран N. Затем для каждой страны названия городов этой страны (страна и города через пробел). Затем вводит число M – количество запросов. Затем M названий городов. Для каждого города укажите, в какой стране он находится.

Например,

ввели:	получили:
2 Беларусь Гродно Гомель Минск Берестовица Россия Москва Воронеж 3 Воронеж Берестовица Минск	Россия Беларусь Беларусь

Контрольные вопросы

1. Что означает термин «последовательность», и какие три типа относятся к этой категории?
2. Чем кортежи отличаются от списков?
3. Как определить размер кортежа?
4. Как задать кортеж, содержащий единственное значение в виде целого числа 42?
5. Как преобразовать список в кортеж? Как преобразовать кортеж в список?
6. Прокомментируйте следующий код (чему равна переменная T):

```
T = (1, 2, 3)
T[2] = 4
T = T[:2] + (4,)
```

7. Напишите выражение, которое изменит первый элемент в кортеже. Кортеж со значением (4,5,6) должен стать кортежем со значением (1,5,6).
8. Как удалить элемент множества не вызывая ошибку, если его там нет?
9. Что такое словарь?
10. Какие действия можно выполнять со словарем?
11. Перечислите разные способы создания словаря.
12. Как быстро проверить, есть ли некоторое значение среди всех значений элементов словаря?
13. Как определить длину словаря?