Практическая работа № 4

Коллекции (продолжение)

3. Словари

Позволяют хранить данные в формате ключ-значение. Изменяемые; неупорядоченные

In [111]:

```
empty_dict = {}
empty_dict = dict()
collections_map = {
'mutable': ['list', 'set', 'dict'],
'immutable': ['tuple', 'frozenset']
}
```

In [112]:

```
print(collections_map['immutable'])
```

```
['tuple', 'frozenset']
```

In [113]:

```
print(collections_map['irresistible'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12640\100508803.py in <cell line: 1>()
----> 1 print(collections_map['irresistible'])

KeyError: 'irresistible'
```

часто бывает полезно попытаться достать значение по ключу из словаря, а в случае отсутствия ключа вернуть какое-то стандартное значение. Для этого есть встроенный метод get

In [114]:

```
print(collections_map.get('irresistible', 'not found'))
```

```
not found
```

Проверка на вхождения ключа в словарь так же осуществляется за константное время и выполняется с помощью ключевого слова in:

In [115]:

```
'mutable' in collections_map
```

Out[115]:

```
True
```

Словари, как и все коллекции, поддерживают протокол итерации. С помощью цикла for можно итерироваться по ключам словаря:

In [121]:

```
print(collections_map)
for key in collections_map:
    print(key)
```

```
{'mutable': ['list', 'set', 'dict'], 'immutable': ['tuple', 'frozenset']}
mutable
immutable
```

Если нам нужно итерироваться не по ключам, а по ключам и значениям сразу, можно использовать метод словаря items, который возвращает ключи и значения.

In [122]:

```
for key, value in collections_map.items():
    print('{} — {}'.format(key, value))
```

```
mutable — ['list', 'set', 'dict']
immutable — ['tuple', 'frozenset']
```

Если нужно итерироваться по значениям, используйте логично метод values, который возвращает именно значения. Также существует симметричный метод keys, который воз вращает итератор ключей.

In [124]:

```
for value in collections_map.values():
    print(value)
```

```
['list', 'set', 'dict']
['tuple', 'frozenset']
```

Упражнение

In [15]:

```
states = {'Россия': 'ru', 'Германия': 'de', 'Узбекистан': 'uz', 'Зимбабве': 'zw','Турция':
states
```

Out[15]:

```
{'Россия': 'ru',
 'Германия': 'de',
 'Узбекистан': 'uz',
 'Зимбабве': 'zw',
 'Турция': 'tr'}
```

In [16]:

```
cities = {'uz':'Ташкент', 'de':'Мюнхен', 'zw':'Harare','tr':'Мармарис'}
cities
```

Out[16]:

```
{'uz': 'Ташкент', 'de': 'Мюнхен', 'zw': 'Harare', 'tr': 'Мармарис'}
```

In [17]:

```
cities['ru'] = 'Таганрог'
cities
```

Out[17]:

```
{'uz': 'Ташкент',
 'de': 'Мюнхен',
 'zw': 'Harare',
 'tr': 'Мармарис',
 'ru': 'Таганрог'}
```

А теперь посмотрим, сколько есть различных способов обращаться к данным в словаре Вывод некоторых городов:

In [18]:

```
print('В стране zw есть город ', cities['zw'])
print('В стране ru есть город ', cities['ru'])
```

```
В стране zw есть город  Harare
В стране ru есть город  Таганрог
```

Вывод некоторых стран:

In [19]:

```
print('Аббревиатура Турции ', states['Турция'])
print('Аббревиатура Германии ', states['Германия'])
```

```
Аббревиатура Турции  tr
Аббревиатура Германии  de
```

Совместное использование двух словарей:

In [20]:

```
print('В России есть город ', cities[states['Россия']])
```

```
В России есть город  Таганрог
```

Вывести аббревиатуры все стран:

In [21]:

```
for state, abbrev in list(states.items()):
    print(f'{state} имеет аббревиатуру {abbrev}')
```

```
Россия имеет аббревиатуру ru
Германия имеет аббревиатуру de
Узбекистан имеет аббревиатуру uz
Зимбабве имеет аббревиатуру zw
Турция имеет аббревиатуру tr
```

Вывод всех городов в странах

In [23]:

```
for abbrev, city in list(cities.items()):
    print(f'В стране {abbrev} есть городу {city}')
```

```
В стране uz есть городу Ташкент
И есть город Ташкент
В стране de есть городу Мюнхен
И есть город Мюнхен
В стране zw есть городу Harare
И есть город Harare
В стране tr есть городу Мармарис
И есть город Мармарис
В стране ru есть городу Таганрог
И есть город Таганрог
```

In [24]:

```
for state, abbrev in list(states.items()):
    print(f'В стране {state} используется аббревиатура {abbrev}')
    print(f'И есть город {cities[abbrev]}')
```

```
В стране Россия используется аббревиатура ru
И есть город Таганрог
В стране Германия используется аббревиатура de
И есть город Мюнхен
В стране Узбекистан используется аббревиатура uz
И есть город Ташкент
В стране Зимбабве используется аббревиатура zw
И есть город Harare
В стране Турция используется аббревиатура tr
И есть город Мармарис
```

Безопасное получение аббревиатуры страны, даже если ее нет в словаре

In [27]:

```
var = 'США'
state = states.get(var)
print(state)
```

```
None
```

In [28]:

```
if not state:
    print(f'Простите, но {var} не существует...')
```

```
Простите, но США не существует...
```

Получение города со значением по умолчанию

In [31]:

```
city = cities.get('US','не существует')
print(f'В стране "US" есть город {city}')
```

```
В стране "US" есть город не существует
```

Словари - неупорядоченный тип данных. Однако есть специальный тип OrderedDict (содержится в модуле collections), который гарантирует вам, что ключи хранятся именно в том порядке, в каком вы их добавили в словарь.

In [125]:

```
from collections import OrderedDict

ordered = OrderedDict()

for number in range(10):
    ordered[number] = str(number)
for key in ordered:
    print(key)
```

```
0
1
2
3
4
5
6
7
8
9
```

4. Множества

Множество в питоне — это неупорядоченный набор уникальных объектов. Множества изменяемы и чаще всего используются для удаления дубликатов и всевозможных проверок на вхождение. Чтобы объявить пустое множество, можно воспользоваться литералом set или использовать фигурные скобки, чтобы объявить множество и одновременно добавить туда какие-то элементы.

In [2]:

```
empty_set = set()
number_set = {1, 2, 3, 3, 4, 5}
print(number_set)
```

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

Чтобы добавить элемент в множество, используется метод add. Также множества в Python поддерживают стандартные операции над множествами --- такие как объединение, разность, пересечение и симметрическая разность.

In [3]:

```
odd_set = set()
even_set = set()
for number in range(10):
    if number % 2:
        odd_set.add(number)
    else:
        even_set.add(number)
print(odd_set)
print(even_set)
```

```
{1, 3, 5, 7, 9}
{0, 2, 4, 6, 8}
```

Теперь найдём объединение и пересечение этих множеств:

In [4]:

```
union_set = odd_set | even_set
print(union_set)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [5]:

```
general_set = odd_set & even_set
general_set
```

Out[5]:

```
set()
```

Разность двух множеств – это множество, в которое входят все элементы первого множества, не входящие во второе множество.

In [9]:

```
difference_set = general_set - odd_set
difference_set
```

Out[9]:

```
set()
```

In [134]:

```
even_set.remove(2)
print(even_set)
```

```
{0, 4, 6, 8}
```

In [135]:

```
dir(set)
```

Out[135]:

```
['__and__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
```

```
 'union',
 'update']
```

In [138]:

```
even_set.add(2)
even_set
```

Out[138]:

```
{0, 2, 4, 6, 8}
```

Также в питоне существует неизменяемый аналог типа set --- тип frozenset.

In [139]:

```
dir(frozenset)
```

Out[139]:

```
['__and__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'copy',
 'difference',
 'intersection',
 'isdisjoint',
 'issubset',
 'issuperset',
 'symmetric_difference',
 'union']
```

List comprehensions (списочные выражения)

Лаконичная конструкция для создания списков и других коллекций, когда цикл пишут прямо в скобках.
Раньше мы делали так:

In [54]:

```
square_list = []
for number in range(10):
    square_list.append(number ** 2)
print(square_list)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Это же можно сделать в 1 строку:

In [55]:

```
square_list = [number ** 2 for number in range(10)]
print(square_list)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

В списочных выражениях можно использовать условия:

In [56]:

```
even_list = [num for num in range(10) if num % 2 == 0]
print(even_list)
```

```
[0, 2, 4, 6, 8]
```

Для словарей:

In [57]:

```
square_map = {number: number ** 2 for number in range(5)}
print(square_map)
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

Если применять list comprehensions с фигурными скобками, но без двоеточий, получим set:

In [58]:

```
reminders_set = {num % 10 for num in range(100)}
print(reminders_set)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

5. Самостоятельно

Дана строка с некоторым текстом.

Требуется создать словарь, который в качестве ключей будет принимать буквы, а в качестве значений – количество этих букв в тексте. Для построения словаря создайте функцию count_it(sequence), принимающую строку с текстом. Функция должна возвратить словарь из 3-х самых часто встречаемых букв.

In [35]:

```
# Решение
def count_it(sequence):
    # При помощи генератора создаем словарь, где ключом выступает уникальный элемент строки
    num_frequency = {item: sequence.count(item) for item in sequence}

    # Сортируем словарь по значениям в порядке возрастания. Для этого методом items() форми
    sorted_num_frequency = sorted(num_frequency.items(), key=lambda element: element[1])

    # Возвращаем последние 3 элемента списка, т. е. кортежи с самыми большими значениями вт
    return dict(sorted_num_frequency[-3:])

# Тесты
print(count_it('фывралвралофравврапыорпаоывпаовпа'))
print(count_it('ывфравдыарыолрафдлваылвпаврпарпаорпаорп'))
print(count_it('dfgjgjhgdjfgjfgfsd'))
```

```
{'р': 5, 'в': 6, 'а': 7}
{'п': 5, 'р': 7, 'а': 8}
{'f': 4, 'j': 4, 'g': 5}
```

Посчитать, через сколько итераций функция random.randint(1, 10) выдаст повтор. Будем добавлять неповторяющиеся случайные числа в множество random_set. Если очередное число уже есть в random_set --- выйдем из цикла. Затем посчитаем длину множества (и прибавим 1, т.к. не учли последнее число)

In [2]:

```
# Решение
import random
random_set = set()
while True:
    new_number = random.randint(1, 10)
    if new_number in random_set:
        break
    random_set.add(new_number)
print(len(random_set) + 1)
```

```
5
```

Файлы

1. Текстовые файлы

Для открытия файлов используется встроенный метод open, которому нужно передать путь к файлу. Функция open возвращает файловый объект, с которым мы потом можем работать. Файлы можно открывать в следующих режимах:

- w - запись,
- r - чтение,
- r+ - чтение и запись,

- a - дозапись. Чтобы записать что-то в файл, применяем к файловому объекту метод write, и передаем ему строку. Метод write возвращает количество символов, которые мы записали.

Стоит также указывать кодировку. Значение по умолчаню 'utf8', но если его не указать явно, могут возникнуть трудности с русской раскладкой.

In [20]:

```
f = open('new_file.txt','w', encoding = 'utf8')
#f = open('new_file.txt','w')
#f.write('The world is changed.\nI taste it in the water.\n')z
#f.write('The world is changed')
f.write('Измученный дорогой, я выбился из сил.\nИ в доме лесника я ночлега попросил.\n')
```

Out[20]:

```
75
```

Хороший тон - всегда закрывать за собой файлы.

In [21]:

```
f.close()
```

Давайте откроем этот файл для чтения и записи.

In [22]:

```
f = open('new_file.txt','r+', encoding = 'utf8')
f.read()
```

Out[22]:

```
'Измученный дорогой, я выбился из сил.\nИ в доме лесника я ночлега попроси
л.\n'
```

Метод tell() указывает положение каретки

In [23]:

```
f.tell()
```

Out[23]:

```
136
```

Если попытаемся прочитать еще раз, то ничего не найдем

In [24]:

```
f.read()
```

Out[24]:

```
''
```

Чтобы прочитать файл заново, нужно использовать метод seek() и перенести указатель на начало файла

In [25]:

```
f.seek(0)
f.tell()
```

Out[25]:

```
0
```

In [26]:

```
print(f.read())
f.close()
```

```
Измученный дорогой, я выбился из сил.
И в доме лесника я ночлега попросил.
```

Можно еще читать файлы построчно, и даже формировать из строк список

In [49]:

```
f = open('new_file.txt', 'r+', encoding = 'utf8')
print(f.readline())
f.close()
```

```
Измученный дорогой я выбился из сил.
```

In [52]:

```
f = open('new_file.txt', 'r+', encoding = 'utf8')
print(f.readlines())
f.close()
```

```
['Измученный дорогой я выбился из сил.\n', 'И в доме лесника я ночлега попро
сил.\n']
```

Можно открывать файлы при помощи контекстного менеджера, тогда не нужно заботиться о закрытии файла. Вы можете открыть файл с помощью оператора with, записать файловый объект в переменную f и потом работать с файлом внутри этого контекстного блока. После выхода из блока интерпретатор Python закроет файл.

In [53]:

```
with open('new_file.txt', 'r+', encoding = 'utf8') as f:
    print(f.read())
```

```
Измученный дорогой я выбился из сил.
И в доме лесника я ночлега попросил.
```

Как думаете, что будет, если переместить каретку в начало файла, открытого для записи, а потом его просто закрыть?

In [29]:

```
f = open('new_file.txt','w', encoding = 'utf8')
f.seek(0)
f.close()
```

In [33]:

```
print(f.name)
print(f.closed)
```

```
new_file.txt
True
```

2. CSV (Comma Separated Values)

По сути, он является обычным текстовым файлом, в котором каждый новый элемент отделен от предыдущего запятой или другим разделителем. Обычно каждая запись начинается с новой строки. Данные CSV можно легко экспортировать в электронные таблицы или базы данных. В Python есть специальная библиотека для работы с такими файлами. Давайте прочитаем данные из отчета об анкетировании на первом занятии.

In [10]:

```
import csv
```

In [19]:

```
with open('Entering_quiz_2.csv', encoding = 'utf8') as csvfile:
    reader_object = csv.reader(csvfile, delimiter = ",")
    for row in reader_object:
        print(row)
```

```
['Отметка времени', 'Баллы', 'Укажите Вашу основную специальность', 'На ка
ком курсе учитесь?', 'Оцените уровень Вашей подготовки в программировании
вообще', 'Оцените уровень Вашей подготовки в программировании на Python',
'Как у Вас с математикой?', 'А что насчет машинного обучения?', 'Чего ждет
е от этого курса?', 'Напишите свой вариант', 'Как с английским?', 'Какую о
перационную систему сейчас используете?']
['04.10.2022 12:41:12', '', 'ccccc', '3', 'ниндзя', 'средний', 'Отлично!',
'Люблю, умею, практикую', 'Сам не знаю, просто все побежали - и я побежал,
Хочу делать нейронные сети, NLP (работа с естественным языком), Техническо
е зрение', 'тралала', '', '']
['04.10.2022 14:30:49', '', 'Промышленная электроника', 'работаю', 'ниндз
я', 'начальный', 'Терпимо', 'Впервые слышу', 'Основы ООП, Функциональное п
рограммирование', '', '', '']
['08.10.2022 11:59:02', '', 'Прикладная математика и информатика', '4', 'н
ачальный', 'начальный', 'Отлично!', 'Интересовался, проходил курсы / учили
этому в универе', 'Типы данных, базовые конструкции и основы Python, Позна
комиться с библиотеками для обработки данных, Хочу делать нейронные сети,
NLP (работа с естественным языком), DeepLearning!!! (что бы это ни значил
о))), Машинное обучение', ':)', 'Достаточно для чтения технической докумен
```

Здорово. Но ничего непонятно. Давайте попробуем разобрать данные

In [53]:

```
with open('Entering_quiz_2.csv', encoding = 'utf8') as csvfile:
    reader_object = csv.reader(csvfile, delimiter = ",")
    count = 0
    ans = []
    for row in reader_object:
        if count == 0:
            print(f'Файл содержит столбцы: {", ".join(row)}')
            header = row
            count += 1
        else:
            ans.append(row)
            count += 1
```

```
Файл содержит столбцы: Отметка времени, Баллы, Укажите Вашу основную специал
ьность, На каком курсе учитесь?, Оцените уровень Вашей подготовки в программ
ировании вообще, Оцените уровень Вашей подготовки в программировании на Pyth
on, Как у Вас с математикой?, А что насчет машинного обучения?, Чего ждете о
т этого курса?, Напишите свой вариант, Как с английским?, Какую операционную
систему сейчас используете?
```

In [54]:

```
header
```

Out[54]:

```
['Отметка времени',
 'Баллы',
 'Укажите Вашу основную специальность',
 'На каком курсе учитесь?',
 'Оцените уровень Вашей подготовки в программировании вообще',
 'Оцените уровень Вашей подготовки в программировании на Python',
 'Как у Вас с математикой?',
 'А что насчет машинного обучения?',
 'Чего ждете от этого курса?',
 'Напишите свой вариант',
 'Как с английским?',
 'Какую операционную систему сейчас используете?']
```

In [55]:

```
len(ans)
```

Out[55]:

```
79
```

In [61]:

```
print(ans[37])
```

```
['нулевой', 'нулевой', 'Терпимо', 'Что-то слышал, но не интересовался', 'Тип
ы данных, базовые конструкции и основы Python, Основы ООП, Функциональное пр
ограммирование, Научиться манипулировать данными, строить красивые графики и
извлекать из них полезную информацию, Хочу делать нейронные сети, NLP (работ
а с естественным языком), Техническое зрение, DeepLearning!!! (что бы это ни
значило))), Машинное обучение, Распознавание объектов на видео', '', 'Достат
очно для чтения технической документации', 'Windows']
```

Видим, что у нас есть пара бессодержательных столбцов. Давайте их удалим

In [63]:

```
for item in ans:
    del item[0:2]
ans[37]
```

Out[63]:

```
['Терпимо',
 'Что-то слышал, но не интересовался',
 'Типы данных, базовые конструкции и основы Python, Основы ООП, Функциональн
ое программирование, Научиться манипулировать данными, строить красивые граф
ики и извлекать из них полезную информацию, Хочу делать нейронные сети, NLP
(работа с естественным языком), Техническое зрение, DeepLearning!!! (что бы
это ни значило))), Машинное обучение, Распознавание объектов на видео',
 '',
 'Достаточно для чтения технической документации',
 'Windows']
```

In [65]:

```
header
```

Out[65]:

```
['Отметка времени',
 'Баллы',
 'Укажите Вашу основную специальность',
 'На каком курсе учитесь?',
 'Оцените уровень Вашей подготовки в программировании вообще',
 'Оцените уровень Вашей подготовки в программировании на Python',
 'Как у Вас с математикой?',
 'А что насчет машинного обучения?',
 'Чего ждете от этого курса?',
 'Напишите свой вариант',
 'Как с английским?',
 'Какую операционную систему сейчас используете?']
```

Итак. У нас имеется два списка с заголовками и данными столбцов. Мы можем собрать их в словарь. Однако в библиотеке csv есть инструмент DictReader, который позволяет, во-первых, задать именна стобцов, если они неопределены или нам не нравятся, во-вторых, обращаться к данным не по индексу, а по имени.

In [107]:

```
with open('Entering_quiz_2.csv', encoding = 'utf8') as csvfile:
    names = ['t','s', 'spec','class','Pr_lvl','Py_lvl','Mth_lvl','ML_lvl','Wait','Own','Eng
    reader_object = csv.DictReader(csvfile, delimiter = ",", fieldnames = names)
    count = 0
    ans = [[],[]]
    for row in reader_object:
        if count == 0:
            header = row
            count += 1
        else:
            ans[0].append(row['Py_lvl'])
            ans[1].append(row['class'])
            count += 1

ans
```

Out[107]:

```
[['средний',
  'начальный',
  'начальный',
  'средний',
  'нулевой',
  'средний',
  'начальный',
  'нулевой',
  'начальный',
  'нулевой',
  'средний',
  'нулевой',
  'средний',
  'нулевой',
  'начальный',
  'нулевой',
  'нулевой',
  'начальный'.
```

Как сохранить данные в csv-файл?

In [118]:

```
with open('new_csv.csv', mode = 'w', encoding = 'utf-8') as w_file:
    file_writer = csv.writer(w_file, delimiter = ",",lineterminator="\r")
    file_writer.writerow(['Py_lvl'])
    file_writer.writerow(ans[0])
```

Самостоятельно

Сохранить в отдельный csv-файл данные о специальностях, уровне знания математики и ангийского Затем считайте этот файл и выведете его содержание.

In [ ]:

```

```