Списки и кортежи

С помощью списков и кортежей программист может решить задачи, связанные с организацией, обработкой, фильтрацией, сортировкой, поиском данных.

Списки — изменяемые, то есть элементы списка можно добавлять, удалять или именять. Кортежи — неизменяемые, элементы в них не могут быть изменены.

Списки

Списки — упорядоченные изменяемые коллекции объектов произвольных типов.

Как объявить список

1 Используя функцию list():

```
name_movie = list('Матрица')
print(name_movie)
# Вывод в терминал: ['М', 'а', 'т', 'р', 'и', 'ц', 'а']

# Такая запись создаст пустой список.
movies = list()
print(movies)
# Вывод в терминал: []
```

С помощью этой функции можно преобразовать в список любой итерируемый объект.

2 Литерально:

```
# Литералом для объявления списка будет
# запись элементов через запятую в квадратных скобках.
movies = ['Матрица', 'Хакеры', 'Трон']

# Литеральное объявление пустого списка.
movie_ratings = []
```

3 Используя списковое включение (от англ. list comprehension) — сокращённую форму записи программного кода, которая создаёт список на основе заданных правил и условий.

```
movie_ratings = [4.7, 5.0, 4.3, 3.8]
# Здесь используется списковое включение.
new_movie_ratings = [rating + 0.2 for rating in movie_ratings]
print(new_movie_ratings)
# Вывод в терминал: [4.9, 5.2, 4.5, 4.0]
```

Чтобы на выходе получился список — код заключается в квадратные скобки, это литералы списка.

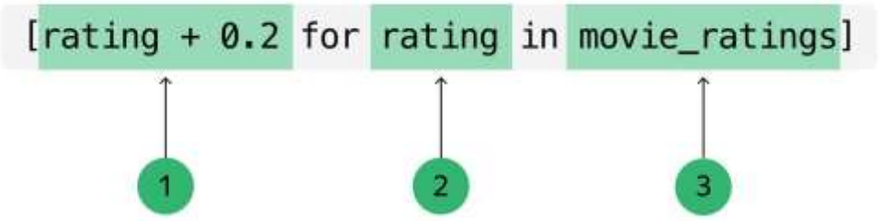
Внутри квадратных скобок можно выделить три части:

1 Значение элемента нового списка: значение из исходного списка, увеличенное на 02

2 Обрабатываемый объект: переменная, в которую по очереди, как в цикле, передаются значения всех элементов списка: rating.

3 Источник, из которого получаем значения элементов: список movie_ratings.

Части отделены друг от друга ключевыми словами for и in.



Условия в list comprehension

1 Тернарный оператор используется, когда нужно избежать превышения предельно возможного значения.

```
movie_ratings = [4.7, 5.0, 4.3, 3.8]
# Добавить в новый список rating + 0.2, если rating < 4.9; иначе добавь rating.
new_movie_ratings = [
    rating +0.2 if rating < 4.9 else rating for rating in movie_ratings
]
print(new_movie_ratings)
# Вывод в терминал: [4.9, 5.0, 4.5, 4.0]
```

2 Оператор if используется, когда нужно сформировать список только из определённых значений.

```
movie_ratings = [4.7, 5.0, 4.3, 3.8]
# Сформировать список из значений, которые больше 4.5.
new_movie_ratings = [rating for rating in movie_ratings if rating > 4.5]
print(new_movie_ratings)
# Вывод в терминал:[4.7, 5.0]
```

Распаковка списка

Операция позволяет одним выражением присвоить значения элементов списка переменным.

```
week = [
    'Понедельник', 'Втроник', 'Среда',
    'Четверг', 'Пятница', 'Суббота', 'Воскресенье'
]

mon, tue, wed, thu, fri, sat, sun = week

print(wed)
# Вывод в терминал: Среда
```


Если число переменных и распаковываемых элементов будет отличаться — возник нет ошибка.

Методы списков

list.append(element) — добавляет элемент в конец списка. Метод append() ничего не возвращает; он не создаёт новый объект, а изменяет существующий.

```
movies = ['Матрица', 'Хакеры', 'Трон']
movies.append('Тихушники')
print(movies)
# Вывод в терминал: ['Матрица', 'Хакеры', 'Трон', 'Тихушники']
```

list_1.extend(list_2) — расширяет и изменяет список list_1 добавляя в конец все элементы списка list_2

```
movie_ratings = [4.7, 5.0, 4.3, 3.8]
movies = ['Матрица', 'Хакеры', 'Трон']
print(id(movies))
# Вывод в терминал: 2217539360448

movies.extend(movie_ratings)
print(movies)
# Вывод в терминал: ['Матрица', 'Хакеры', 'Трон', 4.7, 5.0, 4.3, 3.8]
print(id(movies))
# Вывод в терминал: 2217539360448
```

list.insert(index, value) подставляет элемент со значением value на позицию index и увеличивает индекс элементов, начиная от index, на единицу.

```
movies = ['Матрица', 'Хакеры', 'Трон']
movies.insert(1, 'Тихушники')
print(movies)
# Вывод в терминал: ['Матрица', 'Тихушники', 'Хакеры', 'Трон']
```

list.remove(value) — удаляет первый элемент, значение которого совпадает с аргументом (при чтении списка слева направо).

```
movies = ['Матрица', 'Тихушники', 'Хакеры', 'Трон']
movies.remove('Хакеры')
print(movies)
# Вывод в терминал: ['Матрица', 'Тихушники', 'Трон']
```

list.pop(index) — удаляет из списка элемент с индексом index и возвращает его. Если индекс не указан, метод удаляет и возвращает последний элемент списка.

```
movies = ['Матрица', 'Тихушники', 'Хакеры', 'Трон']
movie = movies.pop(2)
print(movies)
# Вывод в терминал: ['Матрица', 'Тихушники', 'Трон']
```

list.index(value, start end) — читает список слева направо и возвращает индекс первого найденного элемента со значением value.

```
movie_ratings = [4.7, 5.0, 4.3, 3.8]
rating = movie_ratings.index(4.3)
print(rating)
# Вывод в терминал: 2
```

list.count(value) — возвращает количество элементов со значением value.

```
movie_ratings = [4.7, 5.0, 4.3, 3.8, 4.7, 4.1]
rating_count = movie_ratings.count(4.7)
print(rating_count)
# Вывод в терминал: 2
```

list.sort() — сортирует список. Необязательный параметр reverse определяет направление сортировки. По умолчанию reverse = False, элементы сортируются «по возрастанию», от меньшего к большему.

```
movie_ratings = [4.7, 5.0, 4.3, 3.8, 4.7, 4.1]
movie_ratings.sort()
print(movie_ratings)
# Вывод в терминал: [3.8, 4.1, 4.3, 4.7, 4.7, 5.0]

movies = ['Матрица', 'Хакеры', 'Трон']
movies.sort(reverse = True)
print(movies)
# Вывод в терминал: ['Хакеры', 'Трон', 'Матрица']
```

list.reverse() — инвертирует список.

```
movie_ratings = [4.7, 5.0, 4.3, 3.8, 4.7, 4.1]
print(id(movie_ratings))
# Вывод в терминал: 27337512

# Развоврот списка методом reverse()
movie_ratings.reverse()
print(id(movie_ratings))
# Вывод в терминал: 27337512

print(movie_ratings)
# Вывод в терминал: [4.1, 4.7, 3.8, 4.3, 5.0, 4.7]
```

list.copy() — возвращает новый список, независимую копию исходного списка.

```
movie_ratings = [4.7, 5.0, 4.3, 3.8, 4.7, 4.1]
copy_movie_ratings = movie_ratings.copy()
print(id(movie_ratings))
# Вывод в терминал: 38282024

print(id(copy_movie_ratings))
# Вывод в терминал: 38281640

# Применим сортировку к исходному списку
movie_ratings.sort()
print(movie_ratings)
# Вывод в терминал: [3.8, 4.1, 4.3, 4.7, 4.7, 5.0]

# Элементы списка-копии остались неотсортированными
print(copy_movie_ratings)
# Вывод в терминал: [4.7, 5.0, 4.3, 3.8, 4.7, 4.1]
```

list.clear() — очищает список, удаляет из него все элементы.

```
movie_ratings = [4.7, 5.0, 4.3, 3.8, 4.7, 4.1]
movie_ratings.clear()
print(movie_ratings)
# Вывод в терминал: []
```

Кортежи

Кортеж (tuple) — неизменяемая последовательность.

Создание кортежей 1

Литерально:

```
package_1 = ('2:00:01', 15000)
```

2 С помощью функции tuple():

```
package = tuple('object')
print(package)
# Вывод в терминал: ('o', 'b', 'j', 'e', 'c', 't')
```

3 С помощью специального синтаксиса для объявления кортежа. Такой синтаксис называется «упаковка».

```
# Значения, перечисленные через запятую и присвоеные одной переменной,
# будут упакованы в кортеж.
package_2 = '2:00:01', 15000
```

Встроенные функции кортежей

1 Значения элементов кортежа можно присвоить переменным. Такой процесс называется «распаковка».

```
package = ('2:00:01', 15000)

time, steps = package

print(steps)
# Вывод в терминал: 15000
print(time)
# Вывод в терминал: '2:00:01'
```

2 Функция sorted() отвечает за сортировку. В неё передаётся кортеж, а возвращается список.

```
srtd_tpl = sorted(3125, 25, 5, 625, 1, 125)
print(srtd_tpl)
# Вывод в терминал: [1, 5, 25, 125, 625, 3125]
```

3 Функция del() удаляет кортеж целиком.

```
week = (
    'Понедельник', 'Втроник', 'Среда',
    'Четверг', 'Пятница', 'Суббота', 'Воскресенье'
)

del week

print(week)

# Выведется:
# Traceback (most recent call last):
#   File "D:/Dev/course/examples/main.py", line 5, in <module>
#     print(week)
# NameError: name 'week' is not defined
```