

Списки, кортежи

Повторение

- Что такое циклический алгоритм?
- Что возвращает функция range()?
- Приведите пример цикла for
- Приведите пример цикла while
- Для чего нужен оператор Break?
- Для чего нужен оператор Continue?

Списки

- С каким видом «списка» мы уже знакомы? Строки и `range()` возвращает список.
- Т.е. список – это последовательность, набор каких-то данных. В случае со строками это набор однотипных символов, в случае с функцией `range()` это набор чисел.
- Но когда нам нужно хранить другие объекты? Несколько строк например, или разные, неупорядоченные числа. В этом нам поможет такой тип данных как список.

Списки

- Список (list) представляет собой изменяемый тип данных, который хранит набор или последовательность элементов. Для создания списка в квадратных скобках ([]) через запятую перечисляются все его элементы

```
>>> numbers = [1,2,3,4,5]
>>> numbers
[1, 2, 3, 4, 5]
```

- Также для создания списка можно использовать конструктор **list()**:
- Оба этих определения списка аналогичны - они создают список.

```
>>> numbers1 = []
>>> numbers2 = list()
>>> numbers1
[]
>>> numbers2
[]
```



Списки

- Конструктор list для создания списка может принимать другой список.
- Для обращения к элементам списка надо использовать индексы, которые представляют номер элемента в списка. Индексы начинаются с нуля. То есть второй элемент будет иметь индекс 1. \Для обращения к элементам с конца можно использовать отрицательные индексы, начиная с -1. То есть у последнего

```
numbers = [1, 2, 3, 4, 5]
print(numbers[0]) # 1
print(numbers[2]) # 3
print(numbers[-3]) # 3
```

```
numbers[0] = 125 # изменяем первый элемент списка
print(numbers[0]) # 125
```

```
чего - -? и так далее
= [1, 2, 3, 4, 5, 6, 7, 8, 9]
= list(numbers)
```



Списки

- Список необязательно должен содержать только однотипные объекты. Мы можем поместить в один и тот же список одновременно строки, числа, объекты других типов

```
objects = [1, 2.6, "Hello", True]
```

- Для перебора элементов можно использовать как цикл for, так и цикл while. Перебор с помощью цикла for:

```
companies = ["Microsoft", "Google", "Oracle", "Apple"]  
for item in companies:  
    print(item)
```

Практика

- Измените под цикл `while`. Чтобы узнать количество элементов в списке, необходимо воспользоваться функцией `len(переменная-список)`.

```
companies = ["Microsoft", "Google", "Oracle", "Apple"]
i = 0
while i < len(companies):
    print(companies[i])
    i += 1
```

Методы и функции по работе со списками

- Для управления элементами списки имеют целый ряд методов. Некоторые из них:
- **append(item)**: добавляет элемент `item` в конец списка
- **insert(index, item)**: добавляет элемент `item` в список по индексу `index`
- **remove(item)**: удаляет элемент `item`. Удаляется только первое вхождение элемента. Если элемент не найден, генерирует исключение `ValueError`
- **clear()**: удаление всех элементов из списка
- **index(item)**: возвращает индекс элемента `item`. Если элемент не найден, генерирует исключение `ValueError`
- **pop([index])**: удаляет и возвращает элемент по индексу `index`. Если индекс не передан, то просто удаляет последний элемент.
- **count(item)**: возвращает количество вхождений элемента `item` в список
- **sort([key])**: сортирует элементы. По умолчанию сортирует по возрастанию. Но с помощью параметра `key` мы можем передать функцию сортировки.
- **reverse()**: расставляет все элементы в списке в обратном порядке



Проверка наличия элемента

- Если определенный элемент не найден, то методы `remove` и `index` генерируют исключение. Чтобы избежать подобной ситуации, перед операцией с элементом можно проверять его наличие с помощью ключевого слова `in`:

```
companies = ["Microsoft", "Google", "Oracle", "Apple"]
item = "Oracle" # элемент для удаления
if item in companies:
    companies.remove(item)

print(companies)
```

Списки списков

- Списки кроме стандартных данных типа строк, чисел, также могут содержать другие списки. Подобные списки можно ассоциировать с таблицами, где вложенные списки выполняют

```
users = [  
    ["Tom", 29],  
    ["Alice", 33],  
    ["Bob", 27]  
]  
  
print(users[0])           # ["Tom", 29]  
print(users[0][0])       # Tom  
print(users[0][1])       # 29
```

- Чтобы обратиться к элементу вложенного списка, необходимо использовать пару индексов: `users[0][1]` обращение ко второму элементу первого вложенного списка.



Кортежи

- Кортеж (tuple) представляет последовательность элементов, которая во многом похожа на список за тем исключением, что кортеж является неизменяемым (immutable) типом. Поэтому мы не можем добавлять или удалять элементы в кортеже, изменять его.

- Для создания кортежа используются круглые скобки, в которые вносятся значения, разделенные запятыми:

```
user = ("Tom", 23)
print(user)
```

```
user = ("Tom",)
```

- Если вдруг кортеж состоит из одного элемента, то после единственного элемента кортежа необходимо поставить запятую.



Кортежи

- Для создания кортежа из списка можно передать список в функцию `tuple()`, которая возвратит кортеж:

```
users_list = ["Tom", "Bob", "Kate"]
users_tuple = tuple(users_list)
print(users_tuple)      # ("Tom", "Bob", "Kate")
```

- Обращение к элементам в кортеже происходит также, как и в списке по индексу. Индексация начинается также с нуля при получении элементов с начала списка и с -1 при получении элементов с конца списка. Попробуйте получить элементы списка сами.

Перебор кортежей

- Для перебора кортежа можно использовать стандартные циклы **for** и **while**. С помощью цикла **for**:

```
user = ("Tom", 22, False)
for item in user:
    print(item)
```

- С помощью цикла **while**:

```
user = ("Tom", 22, False)

i = 0
while i < len(user):
    print(user[i])
    i += 1
```



Кортежи

- Как для списка с помощью выражения *элемент in кортеж* можно проверить наличие элемента в кортеже.
- Один кортеж может содержать другие кортежи в виде элементов.

```
countries = (  
    ("Germany", 80.2, (("Berlin", 3.326), ("Hamburg", 1.718))),  
    ("France", 66, (("Paris", 2.2), ("Marsel", 1.6)))  
)  
  
for country in countries:  
    countryName, countryPopulation, cities = country  
    print("\nCountry: {} population: {}".format(countryName, countryPopulation))  
    for city in cities:  
        cityName, cityPopulation = city  
        print("City: {} population: {}".format(cityName, cityPopulation))
```



Практика

- Есть список `a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]`. Выведите все элементы, которые меньше 5.

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
for elem in a:
    if elem < 5:
        print(elem)
```

Практика

- Даны списки:
- `a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]`;
- `b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]`.
- Нужно вернуть список, который состоит из элементов, общих для этих двух списков.

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
result = []
for elem in a:
    if elem in b:
        result.append(elem)
print(result)
```


ДЗ

- Выведите первый и последний элемент списка.
- Напишите программу, которая выводит чётные числа из заданного списка и останавливается, если встречает число 237.
- Удаление элементов списка по условию. Из списка чисел удалить элементы, значения которых больше 35 и меньше 65. При этом удаляемые числа сохранить в другом списке.