Списки (list)

9.3.3.1 создавать программы на языке программирования Python (пайтон) с использованием одномерных массивов; использовать различные методы работы со списками

Определение

Список (list) представляет тип данных, который хранит набор или последовательность элементов.

Для создания списка в квадратных скобках ([ ]) через запятую перечисляются все его элементы. Во многих языках программирования есть аналогичная структура данных, которая называется массив.

Например, определим список чисел: **numbers = [1, 2, 3, 4, 5]**

Конструктор list():

Также для создания списка можно использовать конструктор **list()**:

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

Оба этих определения списка аналогичны - они создают пустой список.

Конструктор list для создания списка может принимать другой список:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers2 = list(numbers)
```

Обращение к элементам

Для обращения к элементам списка надо использовать индексы, которые представляют номер элемента в списка.

Например, задан список:

massiv=[“Alua”, “Dima”, “Aldiar”, “Danial”, “Sofia”]

Alua	Dima	Aldiar	Danial	Sofia	*Элементы:*
0	1	2	3	4	*Индексы:*
-5	-4	-3	-2	-1	*Индексы в обратном порядке:*

Как видно, 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 элемента будет индекс -1, у предпоследнего -2 и так

```
File Edit Format Run Options Window Help
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
```

Повторение элементов

Для создания списка, в котором повторяется одно и то же значение несколько раз используем звездочку *.

Например, определим список из шести пятерок:

```
File Edit Format Run Options Window Help
numbers = [5] * 6  # [5, 5, 5, 5, 5, 5]
print(numbers)
```

Функция range

При необходимости создания последовательного списка чисел удобно использовать функцию ***range***, которая имеет три формы:

range(end): создается набор чисел от 0 до числа end

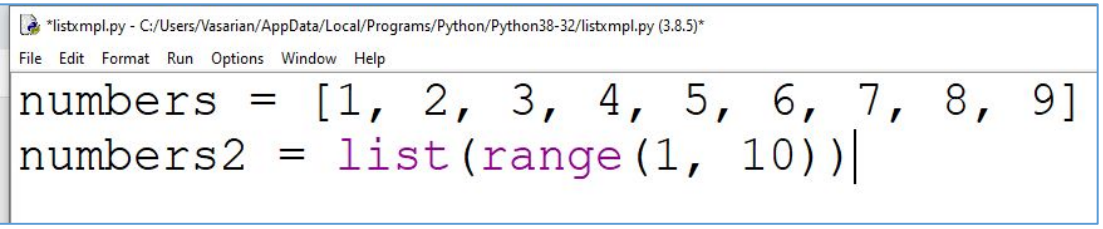
range(start, end): создается набор чисел от числа start до числа end

range(start, end, step): создается набор чисел от числа start до числа end с шагом step

```
numbers = list(range(10))
print(numbers)      # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

numbers = list(range(2, 10))
print(numbers)      # [2, 3, 4, 5, 6, 7, 8, 9]

numbers = list(range(10, 2, -2))
print(numbers)      # [10, 8, 6, 4]
```



К примеру, эти два определения списка аналогичны, но за счет функции **range** сокращается объем кода:

Методы и функции по работе со списками

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
len(list)	возвращает длину списка
sorted(list, [key])	возвращает отсортированный список
min(list)	возвращает наименьший элемент списка
max(list)	возвращает наибольший элемент списка

Методы и функции по работе со списками

Перебор элементов: цикл for, while

Перебор с помощью цикла for:

```
File  Edit  Format  Run  Options  Window  Help
companies = ["Microsoft", "Google", "Oracle", "Apple"]
for item in companies:
    print(item)
```

Перебор с помощью цикла while:

```
File  Edit  Format  Run  Options  Window  Help
companies = ["Microsoft", "Google", "Oracle", "Apple"]
i = 0
while i < len(companies):
    print(companies[i])
    i += 1
```

Добавление и удаление элементов

Для добавления элемента применяются методы **append()** и **insert**, а для удаления - методы **remove()**, **pop()** и **clear()**.

Использование методов: append, insert

File Edit Format Run Options Window Help

```
users = ["Tom", "Bob"]

# добавляем в конец списка
users.append("Alice")
# ["Tom", "Bob", "Alice"]


# добавляем на вторую позицию
users.insert(1, "Bill")
# ["Tom", "Bill", "Bob", "Alice"]
```

Использование методов: index, pop

```
File Edit Format Run Options Window Help
users = ["Tom", "Bill", "Bob", "Alice"]

# получаем индекс элемента
i = users.index("Tom")

# удаляем по этому индексу
removed_item = users.pop(i)

# ["Bill", "Bob", "Alice"]
```

Использование методов: remove, clear

```
File Edit Format Run Options Window Help
users = ["Bill", "Bob", "Alice"]

#определяем последний элемент
last_user = users[-1]

# удаляем последний элемент
users.remove(last_user)

# ["Bill", "Bob"]

# удаляем все элементы
users.clear()
```

Задачки:

1. Есть списки: a = [1, 2, 3]; b = [4, 5]. Найдите значение **a+b, b*3, a + [7, 8] + [9], b+[0, 1]* 3**

2. Создайте список и выведите все элементы списка с четными индексами (то есть **A[0], A[2], A[4],** ...).

3. Создайте список состоящий из чисел. Выведите все четные элементы списка. При этом используйте цикл **for**, перебирающий элементы списка, а не их индексы!

4. Задан список команды баскетболистов из 5 спортсменов. Перед началом соревнования один заболел. Удалите его из списка.

5. Создайте список состоящий из чисел. Замените все четные элементы списка на 0.