

Операции со списками

Ранее было изучено, как создавать списки и добавлять в них элементы.

В данном занятии будет изучено:

- Функция получения длины списка.
- Выбор подмножеств списка (называется срез).
- Возможность расчета, сколько раз элемент появляется в списке.
- Сортировка список предметов

Длина списка

Часто нам нужно найти количество элементов в списке, обычно называемое его длиной.

Мы можем сделать это с помощью функции `len`. Когда мы применяем `len` к списку, мы получаем количество элементов в этом списке:

```
>>my_list = [1, 2, 3, 4, 5]  
>>print(len(my_list))
```

Результатом данного кода будет::

```
5
```

Задание

1. Вычислите длину `list1` и сохраните ее в переменной `list1_len`.

```
list1 = range (2, 20, 2)
```

2. Используйте `print` для проверки `list1_len`.

3. Измените команду `range`, которая генерирует `list1`, чтобы она пропускала 3 вместо 2 элемента.

Как это изменит `list1_len`?

Выбор элементов списка I

Крис проводит собеседование с кандидатами на работу. Он будет называть каждого кандидата по порядку, представленному списком Python:

```
>>calls = ['Алиса', 'Иван', 'Сергей', 'Денис', 'Анна']
```

Сначала он позвонит «Алисе», затем «Иван» и т. д

В Python порядок элемента в списке называют его индексом.

Списки Python начинаются с нулевого индекса. Это означает, что первый элемент в списке имеет индекс 0, а не 1.

Вот порядковые номера этого списка:

Element	Index
'Алиса'	0
'Иван'	1
'Сергей'	2
'Денис'	3
'Анна'	4

В этом примере элемент с индексом 2 - «Сергей».

Мы можем выбрать один элемент из списка, используя квадратные скобки ([]) и индекс элемента списка. Например, если бы мы хотели выбрать третий элемент из списка, мы бы использовали вызовы [2]:

```
>>print(calls[2])
'Сергей'
```

- Используйте квадратные скобки ([]), чтобы выбрать элемент с индексом 4 из списка сотрудников. Сохраните его в переменной index4.

employees = ['Майкл', 'Дуайт', 'Джим', 'Пэм', 'Райан', 'Энди', 'Роберт']

- Используйте print и len, чтобы отобразить, сколько элементов находится у сотрудников.
- Вставьте в код следующую команду:

```
print(employees[8])
```

- При выборе несуществующего элемента возникает ошибка IndexError.

В строке кода, которую вы вставили, измените 8 на другое число, чтобы не получить IndexError.

Выбор элементов списка II

Что, если мы хотим выбрать последний элемент списка?

Для этого в python существуют отрицательные индексы. Можно использовать индекс -1, чтобы выбрать последний элемент списка.

Рассмотрим следующий список из 5 элементов:

```
>>list1 = ['a', 'b', 'c', 'd', 'e']
```

Если мы выберем элемент -1, мы получим последний элемент, 'e':

```
>>print(list1[-1])
```

```
'e'
```

Это то же самое, что и выбор элемента с индексом 4:

```
>>> print(list1[4])
```

```
'e'
```

Задание

- Используйте print и len, чтобы отобразить длину shopping_list.

```
shopping_list = ['яйца', 'масло', 'молоко', 'огурцы', 'сок', 'хлопья']
```

2. Получите последний элемент `shopping_list`, используя индекс -1. Сохраните этот элемент в переменной `last_element`.

3. Теперь выберите элемент с индексом 5 и сохраните его в переменной `element5`.

4. Используйте `print` для отображения как `element5`, так и `last_element`.

Срезы списков

Допустим, у нас есть список букв:

```
>>letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

Предположим, мы хотим выбрать от b до f.

Мы можем сделать это, используя следующий синтаксис: буквы [начало: конец], где:

- `start` - это индекс первого элемента, который мы хотим включить в наш выбор. В этом случае мы хотим начать с b, у которого индекс 1.

- `end` - это индекс на один больше, чем последний индекс, который мы хотим включить. Последний элемент, который нам нужен, - это f с индексом 5, поэтому `end` должно быть 6.

```
>>sublist = letters[1:6]
```

```
>>print(sublist)
```

Этот код выведет:

```
['b', 'c', 'd', 'e', 'f']
```

Обратите внимание, что элемент с индексом 6 (то есть g) не включен в наш выбор.

Задание

1. Используйте `print`, чтобы проверить переменную `beginning`.

```
suitcase = ['рубашка', 'рубашка', 'брюки', 'брюки', 'пижамы', 'книги']
```

```
beginning = suitcase [0: 2]
```

Сколько элементов в списке?

2. Измените `beginning`, чтобы оно выделяло первые 4 элемента чемодана.

3. Создайте новый список под названием `middle`, содержащий два средних элемента из чемодана.

Срез списков II

Если мы хотим выбрать первые 3 элемента списка, мы могли бы использовать следующий код:

```
>>fruits = ['яблоко', 'банан', 'вишня', 'дыня']
```

```
>>print(fruits[0:3])
```

```
['яблоко', 'банан', 'вишня']
```

Начиная с начала списка, также допустимо опускать 0:

```
>>print(fruits[:3])
```

```
['яблоко', 'банан', 'вишня']
```

Мы можем сделать нечто подобное при выборе нескольких последних элементов списка:

```
>>print(fruits[2:])
```

```
['вишня', 'дыня']
```

Можно опустить последний индекс при выборе последних элементов из списка.

Если необходимо выбрать последние 3 элемента фруктов, мы также можем использовать этот синтаксис:

```
>>print(fruits[-3:])  
['банан', 'вишня', 'дыня']
```

Допустимо использовать отрицательные индексы для обратного отсчета от последнего элемента.

Задание

Создайте новый список под названием start, содержащий первые 3 элемента чемодана.

```
suitcase = ['рубашка', 'футболка', 'носки', 'очки', 'пижама', 'книги']
```

Подсчет элементов в списке

Предположим, у нас есть список букв, который представляет буквы в слове «Миссисипи»:

```
>>letters = ['м', 'и', 'с', 'с', 'и', 'с', 'и', 'п', 'и']
```

Если мы хотим знать, сколько раз «и» встречается в этом слове, мы можем использовать функцию count:

```
>>num_i = letters.count('и')  
>>print(num_i)
```

В результате исполнения этого кода будет выведено:

```
4
```

Задание

Класс миссис Уилсон голосует за президента класса. Она сохранила голос каждого студента в списке голосов.

Используйте счетчик, чтобы определить, сколько студентов проголосовало за «Jake». Сохраните свой ответ как jake_votes.

```
votes = ['Jake', 'Jake', 'Laurie', 'Laurie', 'Jake', 'Jake', 'Jake', 'Laurie',  
'Cassie', 'Cassie', 'Jake', 'Jake', 'Cassie', 'Laurie', 'Cassie', 'Jake',  
'Jake', 'Cassie', 'Laurie']
```

Используйте print для проверки jake_votes.

Сортировка списков I

Иногда нам нужно отсортировать список по возрастанию (1, 2, 3,...) или в алфавитном (a, b, c,...) порядке.

Мы можем отсортировать список, используя .sort(). Предположим, у нас есть список имен:

```
>>names = ['Xander', 'Buffy', 'Angel', 'Willow', 'Giles']  
>>print(names)
```

Это выведет:

```
['Xander', 'Buffy', 'Angel', 'Willow', 'Giles']
```

Теперь применим .sort ():

```
>>names.sort()  
>>print(names)
```

И получаем:

```
[ 'Angel', 'Buffy', 'Giles', 'Willow', 'Xander' ]
```

Обратите внимание, что функция сортировки пишется после списка имен. Если попробовать sort (names), мы получим NameError.

sort ничего не возвращает. Итак, если мы попытаемся присвоить переменной names.sort (), наша новая переменная будет None:

```
>>sorted_names = names.sort()  
>>print(sorted_names)
```

это вернет:

```
None
```

Хотя sorted_names имеет значение None, строка sorted_names = names.sort () по-прежнему редактирует имена:

```
>>print(names)
```

```
[ 'Angel', 'Buffy', 'Giles', 'Willow', 'Xander' ]
```

Задание

Отсортируйте адреса.

```
addresses = ['221 B Baker St.', '42 Wallaby Way', '12 Grimmauld Place', '742 Evergreen Terrace',  
, '1600 Pennsylvania Ave', '10 Downing St.]
```

Используйте print, чтобы увидеть, как менялись addresses

Сортировка списков II

Второй способ сортировки списка - использовать sorted. sorted отличается от .sort () следующим:

1. Оно идет перед списком, а не после него.
2. Создает новый список.

Вернемся к нашему списку имен:

```
>>names = ['Xander', 'Buffy', 'Angel', 'Willow', 'Giles']
```

Используя sorted, мы можем создать новый список с именем sorted_names:

```
>>sorted_names = sorted(names)  
>>print(sorted_names)
```

Это дает список, отсортированный по алфавиту:

```
[ 'Angel', 'Buffy', 'Giles', 'Willow', 'Xander' ]
```

Обратите внимание, что использование sorted не изменило names:

```
>>print(names)  
[ 'Xander', 'Buffy', 'Angel', 'Willow', 'Giles' ]
```

Задание

1. Используйте сортировку, чтобы упорядочить игры и создать новый список с именем games_sorted.

```
games = ['Portal', 'Minecraft', 'Pacman', 'Tetris', 'The Sims', 'Pokemon']
```

2. Используйте print для проверки игр и games_sorted.

Выводы

Было изучено:

- Как получить длину списка
- Выбор подмножеств списка (называется срезом).
- Расчет того, сколько раз элемент появляется в списке.
- Сортировка списка элементов

Задание

1. inventory - это список предметов, которые есть на складе мебели. Сколько товаров на складе?

Сохраните свой ответ в inventory_len.

```
inventory = ['двуспальная кровать', 'двуспальная кровать', 'изголовье', 'двуспальная кровать',  
'двуспальная кровать', 'комод', 'комод', 'стол', 'стол', 'тумбочка', 'тумбочка', 'королевский  
кровать', 'двуспальная кровать', 'две односпальные кровати', 'две односпальные кровати',  
'простыни', 'простыни', 'подушка', 'подушка']
```

2. Выберите первый элемент в инвентаре. Сначала сохраните его в переменной first.

3. Выберите последний элемент из инвентаря и сохраните его в переменной last.

4. Выберите предметы из инвентаря, начиная с индекса 2 и до индекса 6, но не включая его.
Сохраните свой ответ в inventory_2_6.

5. Выберите первые 3 предмета инвентаря и сохраните их в переменной first_3.

6. Сколько односпальных кроватей в инвентаре? Сохраните свой ответ в twin_beds.

7. Сортировка инвентаря с помощью .sort () .