Python предлагает ряд составных типов данных, часто называемых последовательностями. Список является одним из наиболее часто используемых и очень универсальных типов данных, используемых в Python.

Как создать список?

В программировании на Python список создается путем помещения всех элементов (элементов) в квадратную скобку [], разделенных запятыми.

Он может иметь любое количество элементов, и они могут быть разных типов (целое число, число с плавающей запятой, строка и т. Д.).

```
# empty list
my_list = []


# list of integers
my_list = [1, 2, 3]


# list with mixed datatypes
my_list = [1, "Hello", 3.4]
```

Кроме того, список может даже иметь другой список как элемент. Это называется вложенным списком.

```
# вложенный список


my_list = ["mouse", [8, 4, 6], ['a']]
```

Как получить доступ к элементам из списка?

Существуют различные способы доступа к элементам списка.

Индекс списка

Мы можем использовать оператор индекса [] для доступа к элементу в списке. Индекс начинается с 0. Итак, список из 5 элементов будет иметь индекс от 0 до 4.

Попытка получить доступ к элементу, другому, что это вызовет IndexError. Индекс должен быть целым числом. Мы не можем использовать float или другие типы, это приведет к TypeError.

Доступ к вложенному списку осуществляется с помощью вложенной индексации. my_list = ['p','r','o','b','e']

Output: p

print(my_list[0])

Output: o

```
print(my_list[2])

# Output: e

print(my_list[4])

# Error! Only integer can be used for indexing

# my_list[4.0]

# Nested List

n_list = ["Happy", [2,0,1,5]]

# Nested indexing

# Output: a

print(n_list[0][1])

# Output: 5

print(n_list[1][3])
```

Отрицательная индексация

Python допускает отрицательную индексацию для своих последовательностей. Индекс -1 относится к последнему элементу, -2 - ко второму последнему элементу и т. Д.

```
 my_list = ['p','r','o','b','e']



# Output: e

print(my_list[-1])

# Output: p

print(my_list[-5])
```

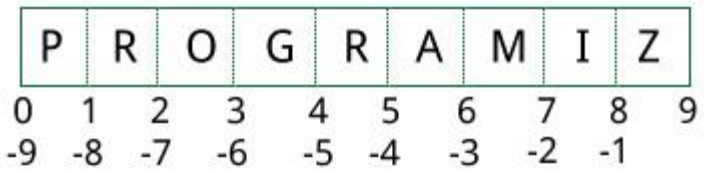
Как нарезать списки в Python?

Мы можем получить доступ к ряду элементов в списке, используя оператор среза (двоеточие). my_list = ['p','r','o','g','r','a','m','i','z']

```
# elements 3rd to 5th

print(my_list[2:5])

# elements beginning to 4th
```

```
print(my_list[:-5])

# elements 6th to end

print(my_list[5:])

# elements beginning to end

print(my_list[:])
```

Нарезку лучше всего визуализировать, считая индекс между элементами, как показано ниже. Поэтому, если мы хотим получить доступ к диапазону, нам нужны два индекса, которые будут вырезать эту часть из списка.



Как изменить или добавить элементы в список?

Список является изменяемым, то есть его элементы могут быть изменены в отличие от строки или кортежа .

Мы можем использовать оператор присваивания (=), чтобы изменить элемент или диапазон элементов. # mistake values

```
odd = [2, 4, 6, 8]

# change the 1st item

odd[0] = 1

# Output: [1, 4, 6, 8]

print(odd)

# change 2nd to 4th items

odd[1:4] = [3, 5, 7]

# Output: [1, 3, 5, 7]

print(odd)
```

Мы можем добавить один элемент в список, используя append()метод или добавить несколько элементов, используя extend()метод.

```
odd = [1, 3, 5]
odd.append(7)
```

```
# Output: [1, 3, 5, 7]
print(odd)
odd.extend([9, 11, 13])
# Output: [1, 3, 5, 7, 9, 11, 13]
print(odd)
```

Мы также можем использовать оператор + для объединения двух списков. Это также называется конкатенацией.

Оператор * повторяет список заданное количество раз.

```
odd = [1, 3, 5]
# Output: [1, 3, 5, 9, 7, 5]
print(odd + [9, 7, 5])
#Output: ["re", "re", "re"]
print(["re"] * 3)
```

Кроме того, мы можем вставить один элемент в нужное место с помощью метода insert()или вставить несколько элементов, втиснув его в пустой фрагмент списка.

```
odd = [1, 9]
odd.insert(1,3)
# Output: [1, 3, 9]
print(odd)
odd[2:2] = [5, 7]


# Output: [1, 3, 5, 7, 9]
print(odd)
```

Как удалить или удалить элементы из списка?

Мы можем удалить один или несколько элементов из списка, используя ключевое my_list = ['p','r','o','b','l','e','m']

```
# delete one item
del my_list[2]
# Output: ['p', 'r', 'b', 'l', 'e', 'm']
print(my_list)
```

```
# delete multiple items

del my_list[1:5]

// Output: ['p', 'm']

print(my_list)

# delete entire list

del my_list

# Error: List not defined

print(my_list)
```

Мы можем использовать remove()метод для удаления данного элемента или pop()метод для удаления элемента по указанному индексу.

pop()Метод удаляет и возвращает последний элемент , если индекс не предусмотрен. Это помогает нам реализовывать списки в виде стеков (структура данных «первый вошел, последний вышел»).

Мы также можем использовать clear()метод для очистки списка. my_list = ['p','r','o','b','l','e','m']

```
my_list.remove('p')

# Output: ['r', 'o', 'b', 'l', 'e', 'm']

print(my_list)

# Output: 'o'

print(my_list.pop(1))

# Output: ['r', 'b', 'l', 'e', 'm']

print(my_list)

# Output: 'm'

print(my_list.pop())

# Output: ['r', 'b', 'l', 'e']

print(my_list)

my_list.clear()

# Output: []

print(my_list)
```

Наконец, мы также можем удалить элементы в списке, назначив пустой список фрагменту элементов.

```
>>> my_list = ['p','r','o','b','l','e','m']
```

```
>>> my_list[2:3] = []
>>> my_list
['p', 'r', 'b', 'l', 'e', 'm']
>>> my_list[2:5] = []
>>> my_list
['p', 'r', 'm']
```

Методы списка Python

Методы, которые доступны со списком объектов в программировании на Python, приведены в таблице ниже.

Они доступны как list.method(). Некоторые из методов уже были использованы выше.

Методы списка Python
append () - добавляет элемент в конец списка
extend () - добавляет все элементы списка в другой список
insert () - вставить элемент по указанному индексу
remove () - удаляет элемент из списка
pop () - удаляет и возвращает элемент по указанному индексу
clear () - удаляет все элементы из списка
index () - возвращает индекс первого соответствующего элемента
count () - Возвращает количество элементов, переданных в качестве аргумента.
sort () - сортировка элементов в списке в порядке возрастания
reverse () - обратный порядок элементов в списке
copy () - возвращает поверхностную копию списка.

Некоторые примеры методов списка Python:

```
my_list = [3, 8, 1, 6, 0, 8, 4]

# Output: 1

print(my_list.index(8))

# Output: 2

print(my_list.count(8))
```

```
my_list.sort()

# Output: [0, 1, 3, 4, 6, 8, 8]

print(my_list)

my_list.reverse()

# Output: [8, 8, 6, 4, 3, 1, 0]

print(my_list)
```

Понимание списка: элегантный способ создания нового списка

Понимание списка - это элегантный и лаконичный способ создания нового списка из существующего списка в Python.

Понимание списка состоит из выражения, за которым следует оператор for в квадратных скобках.

Вот пример, чтобы составить список, в котором каждый элемент имеет возрастающую мощность 2.

```
pow2 = [2 ** x for x in range(10)]

# Output: [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
print(pow2)
# Вывод: [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

Этот код эквивалентен

```
pow2 = []
for x in range(10):
   pow2.append(2 ** x)
```

Понимание списка может содержать дополнительные операторы iffor или if . Необязательный ifоператор может отфильтровывать элементы для нового списка. Вот несколько примеров.

```
>>> pow2 = [2 ** x for x in range(10) if x > 5]
>>> pow2
[64, 128, 256, 512]
>>> odd = [x for x in range(20) if x % 2 == 1]
>>> odd
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
>>> [x+y for x in ['Python ','C '] for y in ['Language','Programming']]
['Python Language', 'Python Programming', 'C Language', 'C Programming']
```

Другие операции со списками в Python

Список участников теста

Мы можем проверить, существует ли элемент в списке, используя ключевое слово in. my_list = ['p','r','o','b','l','e','m']

Output: True

print('p' in my_list)

Output: False

print('a' in my_list)

Output: True

print('c' not in my_list)

Итерация по списку

Используя for цикл, мы можем перебирать каждый элемент списка.

for fruit in ['apple','banana','mango']:

    print("I like",fruit)

Встроенные функции со списком

Встроенные функции , такие как all(), any(), enumerate(), len(), max(), min(), list(), и sorted()т.д. , как правило , используется со списком для выполнения различных задач.

Встроенные функции со списком	
функция	Описание
all()	Верните True, если все элементы списка истинны (или если список пуст).
any ()	Верните True, если какой-либо элемент списка имеет значение true.Если список пуст, вернуть False.
enumerate ()	Вернуть перечисляемый объект. Он содержит индекс и значение всех элементов списка в виде кортежа.
LEN ()	Вернуть длину (количество элементов) в списке.
list()	Преобразуйте итерируемый (кортеж, строка, набор, словарь) в список.
max()	Вернуть самый большой элемент в списке.
min ()	Вернуть самый маленький элемент в списке
sorted()	Возвращает новый отсортированный список (не сортирует сам список).
sum ()	Вернуть сумму всех элементов в списке.