**Step 1. Basic operations with lists**

LISTS

To store several values, lists are used.

To create a list, use the syntax of the form:

myList = ['one', 'two', 'three']

Any data types can be used as list elements:

list\_of\_some\_values = [123, 'python', True]

Lists are very convenient to work with when you need to filter, transform, and combine data.

Let's see the basic operations for working with lists using the example of a set of values:

just\_numbers = [2, 1, 5, 4, 3]

**Basic list operations**

Basic simple operations with lists:

Number of items in a list

len( just\_numbers )

What will be the result?

Sum of list elements

sum( just\_numbers )

What will be the result?

**Step 2. Sorting lists**

Sort Ascending

Sort list items in ascending order

sorted( just\_numbers )

What will be the result?

SORT BY DESCENT

If the list elements are words, then sorting will occur alphabetically.

To sort in descending order, add the **reverse = True** parameter:

sorted( just\_numbers, reverse = True )

[5, 4, 3, 2, 1]

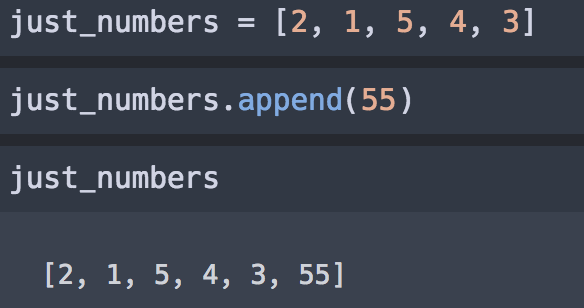
**Step 3. Adding an item to the list**

**ADDING AN ITEM TO THE LIST**

The *append* method is used to add an item to the list:

just\_numbers = [**2**, **1**, **5**, **4**, **3**]

just\_numbers.append(**55**)



Exercise

Create a list of three words: 'one', 'two', 'three'.

Display this list sorted alphabetically (use the following format: ['...', '...', '...'], where list elements should be used instead of ellipsis).

**Step 4. Lists and strings**

**HOW TO GET A SHEET FROM A LINE**

In most cases, the data will come to you as lists, not rows. For example, when reading a table with data from a file. If the table has a separator (for example, like a comma in our task), then the split function can be used to transfer a string to a list. In parentheses, we indicate the separator of elements (in our example, this is a comma):

queries\_string = "watch TV series online, sports news, movie poster, dollar rate, series this summer, python course, sports series"

Apply split

Dial in a new cell

**print**( queries\_string.split(',') )

What will be the result?

**Step 5. Verification task**

You are given the following line with the names of the files that are out of order:

"003\_logs\_2017-11-03; 001\_logs\_2017-11-01; 005\_logs\_2017-11-05; 002\_logs\_2017-11-02; 004\_logs\_2017-11-04"

Write this string to the file\_string variable.

You need:

1. Convert this string to a list (call the sheet file\_list). Use a semicolon as a separator.

What function can you use to do this?

append

strip

split

2. Sort the list in ascending order of dates (that is, the file 001\_logs\_2017-11-01 is the first, and 005\_logs\_2017-11-05 is the last). Write the result to the file\_list\_sorted variable.

What code will allow you to do this conversion?

file\_list\_sorted = sorted(file\_list, reverse = True)

file\_list\_sorted = sorted(file\_list)

';'.join(file\_list)

3. Add the following item to the sorted list: 006\_logs\_2017-11-06.

What result should you get in the end?

['001\_logs\_2017-11-01', '002\_logs\_2017-11-02', '003\_logs\_2017-11-03', '004\_logs\_2017-11-04', '005\_logs\_2017-11-05', '006\_logs\_2017-11-06']

['006\_logs\_2017-11-06', '005\_logs\_2017-11-05', '004\_logs\_2017-11-04', '003\_logs\_2017-11-03', '002\_logs\_2017-11-02', '001\_logs\_2017-11-01']

['003\_logs\_2017-11-03', '001\_logs\_2017-11-01', '005\_logs\_2017-11-05', '002\_logs\_2017-11-02', '004\_logs\_2017-11-04', '006\_logs\_2017-11-06']

**Exporting Calculation Results**

**Step 1. We translate the results of calculations into a string**

**TRANSLATING A LIST TO A LINE**

The *join* function is used to convert the list to a string. Specify the desired separator in quotes. For example, given a list with search queries:

queriesList = ['watch TV series online', 'sports news', 'movie poster', 'dollar exchange rate', 'series this summer', 'python course', 'sports series']

You can convert the list back to a semicolon string as a separator.

**print**( ','.join( queriesList ) )

Translation to string

What text is displayed as a result of this command?

**Step 2. Using a different separator**

**SEPARATOR TABULATION**

This method is very convenient when writing results to a file, especially if the amount of data to be recorded is not known in advance. For example, after filtering, the number of remaining requests can be any. If you need to use the tabulation character as a separator (this is convenient for further export to Excel), then use the \ t service character:

**print**( '**\t**'.join( queriesList ) )

Exercise

In the last exercise you got the list

results = ['001\_logs\_2017-11-01', '002\_logs\_2017-11-02', '003\_logs\_2017-11-03', '004\_logs\_2017-11-04', '005\_logs\_2017-11-05', '006\_logs\_2017-11-06']

Convert this list to a string using two vertical bars || as separators.

What will be the result?

**Step 3. Selecting sheet elements**

**SELECTING LIST ITEMS**

In working with data, it is very often necessary to select list items based on their position. For example, the second from the right. Or select only even ones, starting with the third element. There are simple and convenient filtering methods for such list operations in python.

Let's take another example of a list:

sequence = [ 'Google Adwords', 'Yandex Direct', 'Facebook', 'VK', 'Partner' ]

To get the needed item, enter its number:

sequence[**1**]

**Step 4. Index**

Please note that instead of the first 'Google Adwords' element, for some reason we received the 'Yandex Direct' element. This example shows a very important feature of the numbering of elements in all major programming languages: the numbers of all elements start from zero. That is, the 'Google Adwords' element has a serial number 0:

sequence[**0**]

Result:

'Google Adwords'

Exercise

If we want to get elements from the first to the third, then we write like this:

sequence[**0**:**3**]

What result?

**Step 5. Borders are optional**

Due to the fact that all numbering is shifted 1 to the left, Python will take all elements from zero to the third minus 1, which we see in the example. This algorithm may seem strange, but later we will see that it is very convenient when working with different intervals.

In the previous example, you can omit 0, that is, just write

sequence[:**3**]

Result:

['Google Adwords', 'Yandex Direct', 'Facebook']

Exercise

How can you get the following result (several correct options are possible)?

['Google Adwords', 'Yandex Direct', 'Facebook']

sequence[0:2]

sequence[0:3]

sequence[:2]

sequence[:3]

**Step 6. Advanced filtering of items**

Filtering Items

Display all elements from the third to the last in the list

sequence = [ 'Google Adwords', 'Yandex Direct', 'Facebook', 'VK', 'Partner' ]

sequence[**2**:]

What result will you get?

**NEGATIVE INDEX**

As you saw in the last step, negative indices can be used in lists. That is, the element number will be counted from the end. For example, to display the last item in the list, enter sequence [-1]:

sequence[-**1**]

'Partner'

Exercise

Which command can be used to display all elements from the third to the last but one? That is, get ['Facebook', 'VK']. Several correct answers are possible.

sequence[2:-1]

sequence[-3:-1]

sequence[-3:4]

sequence[2:4]

**Step 7. Filtering with steps**

You can also set the interval with which we go through the list. For example, to get only even numbers from a sequence 1 through 10, use the following syntax:

justNumbers = [**1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**]

justNumbers[**1**::**2**]

Exercise

What will the system give out after such a command?