The class bidicommand is a case of the class TextCommand in the module Sublime\_plugin.

It has a method run that takes edit.

The sublime api defines edit

Edit objects have no functions, they exist to group buffer modifications.

region = sublime.Region(0, self.view.size())

first we need to find self.view.size() of from TextCommand

when we create a text command we give it a view as an input.

View is a sublime class. The api defines it:

Represents a view into a text buffer

It has the function size()

def size(self):

return sublime\_api.view\_size(self.view\_id)

now, view\_id is in turn inputed for the view object.

The api doesn’t have view\_size but it has size for view

int Returns the number of character in the file.

So view.size gives has the number of charecters in a file

Region is another sublime class. The API defines it

Represents an area of the buffer

Essentialy it take two text pointers and represent the text between them. The direction could be positive or negative. In this case from 0 to size is the whole of a file.

next line:

bidiRegion(region, self.view, edit)

we essentialy know all three, region and view are the whole text, and edit is the input.

txt = view.substr(region)

we are asked:

if isinstance(region, Region)

the answer is yes

return sublime\_api.view\_cached\_substr(self.view\_id, x.a, x.b)

this is not documented...

next line

reshaped\_text = reshape(txt)

this is mostly for arabic so we’ll skip it for now

bdiText = get\_display(reshaped\_text)

get\_display(unicode\_or\_str):

storage = get\_empty\_storage()

get\_empty\_storage():

return {

'base\_level': None,

'base\_dir' : None,

'chars': [],

'runs' : deque(),

}

Deque is a list-like object which easily could be extended in both directions.

if isinstance(unicode\_or\_str, str)

no it’s unicode

text = unicode\_or\_str.decode(utf-8)

decoded = True

ok

if base\_dir is None:

yes

base\_level = get\_base\_level(text, upper\_is\_rtl)

our input was upper\_is\_rtl=False