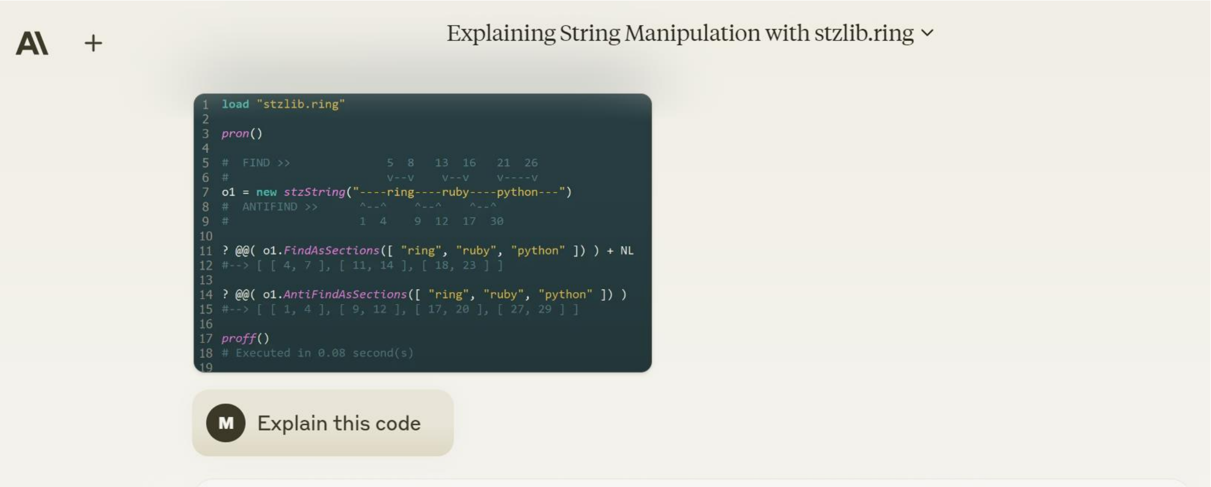
BY CLAUDE AI – FROM ANTHROPIC



```
load "stzlib.ring"

pron()

#  FIND >>              5  8   13  16   21  26
#                       v--v    v--v    v----v
o1 = new stzString("----ring----ruby----python---")
#  ANTIFIND >>      ^--^    ^--^    ^--^
#                   1  4    9  12  17  30

? @@( o1.FindAsSections([ "ring", "ruby", "python" ]) ) + NL
#--> [ [ 4, 7 ], [ 11, 14 ], [ 18, 23 ] ]

? @@( o1.AntiFindAsSections([ "ring", "ruby", "python" ]) )
#--> [ [ 1, 4 ], [ 9, 12 ], [ 17, 20 ], [ 27, 29 ] ]

proff()
# Executed in 0.08 second(s)
```

This code demonstrates the usage of the "stzlib.ring" library, specifically focusing on string manipulation using the stzString class. Here's a breakdown of what's happening:

1. The code loads the "stzlib.ring" library.

2. It creates a new stzString object `o1` with the value "----ring----ruby----python----".

3. Two main operations are performed on this string:

a. FindAsSections():

This method finds the positions of specified substrings ("ring", "ruby", "python") within the main string. The result shows the start and end indices of each found substring:

- "ring" is found at positions [4, 7]
- "ruby" is found at positions [11, 14]
- "python" is found at positions [18, 23]

b. AntiFindAsSections():

This method finds the sections of the string that are not the specified substrings. It returns:

- [1, 4]: the section before "ring"
- [9, 12]: the section between "ring" and "ruby"
- [17, 20]: the section between "ruby" and "python"
- [27, 29]: the section after "python"

The code also includes some visual aids:

- FIND >> shows the positions of the found substrings

- ANTIFIND >> shows the positions of the sections not matching the substrings

Finally, the code execution time is measured, showing it took 0.08 seconds to run.

Would you like me to explain any specific part in more detail?