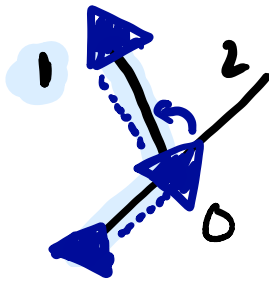


Branching when drawing trees

Let's say we're drawing a tree branch 1



we're stuck!

how do we get back to the main branch and continue onto branch 2?

goto!

But goto where? At what heading?

Before we leave 0 and go up to 1, we should save our state (position + heading).

Then after drawing 1, we can restore the state, going to 0, before we hit a dead end.

then continue drawing branch 2.

In Summary:

Before embarking down a branch, save position + heading, then restore this state after drawing the branch

Implement with 2 new L-system symbols:

- [Save current state (heading + position)
-] Restore previous saved state (heading + position)

L system string for

"F[+F]F"

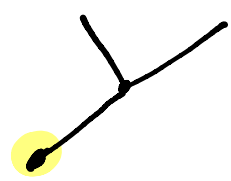
↑ ↑
save restore
go along 1 go along 2



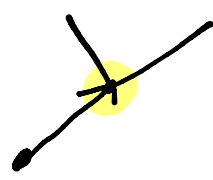
How to implement [and] symbols to save state?

Stack the information in a list:

Stack = []



Stack.append(headng)
Stack.append(position)



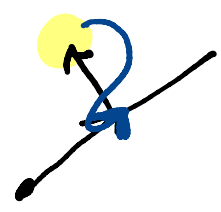
Stack: [headng, position]

prevPos = Stack.pop()

Stack: [headng]

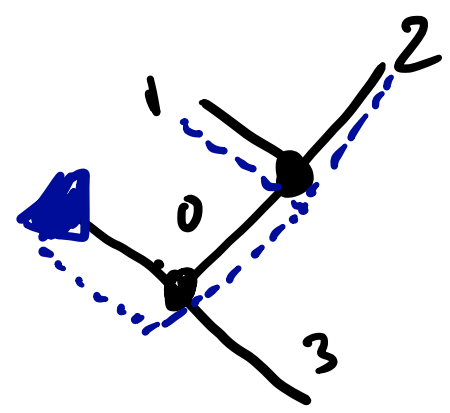
prevHead = Stack.pop()

Stack = []



* If we append a position after headng,
we need to pop position before headng

Practice: What is the L-system string to draw this?



" F [+ F [+ F] F] F
 ↑ ↑ ↑ ↑ ↑
 Saul 0 Saul 1 2 3

L-system project note:

Why do we **loop char-by-char** to apply replacement rules — and not use **replace String method?**

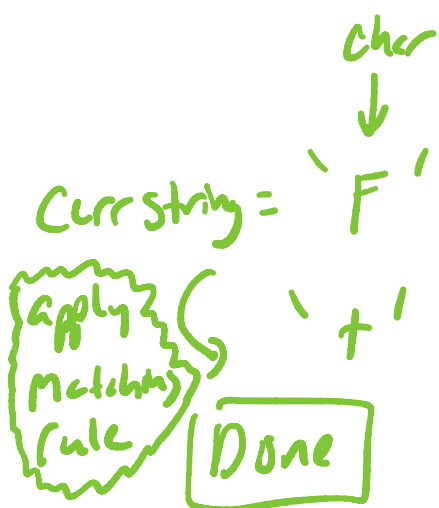
⇒ If there are > 1 rule, results will be different [in undesirable way]

Simple example

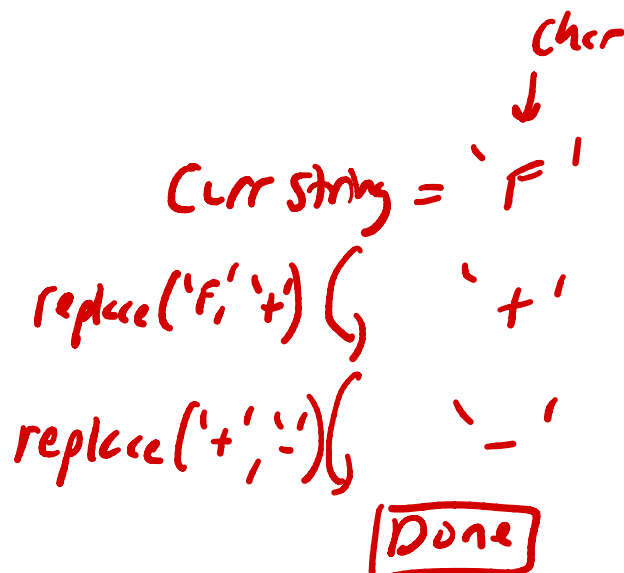
base = 'F'

self.rules = [['F', '+'], ['+', '-']]

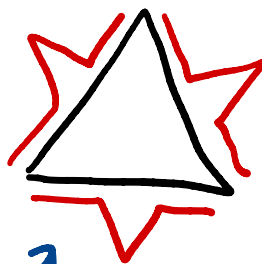
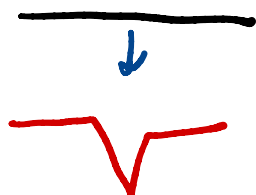
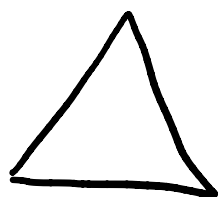
Loop char-by-char



String replace() on rules one-by-one



★ Key: replace may change the result of
previously applied rules mid-iteration
⇒ Not what we want!



Don't want to modify 
mid-iteration — let it be!