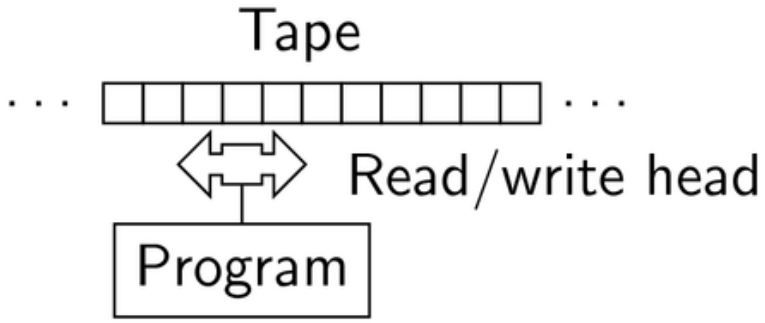
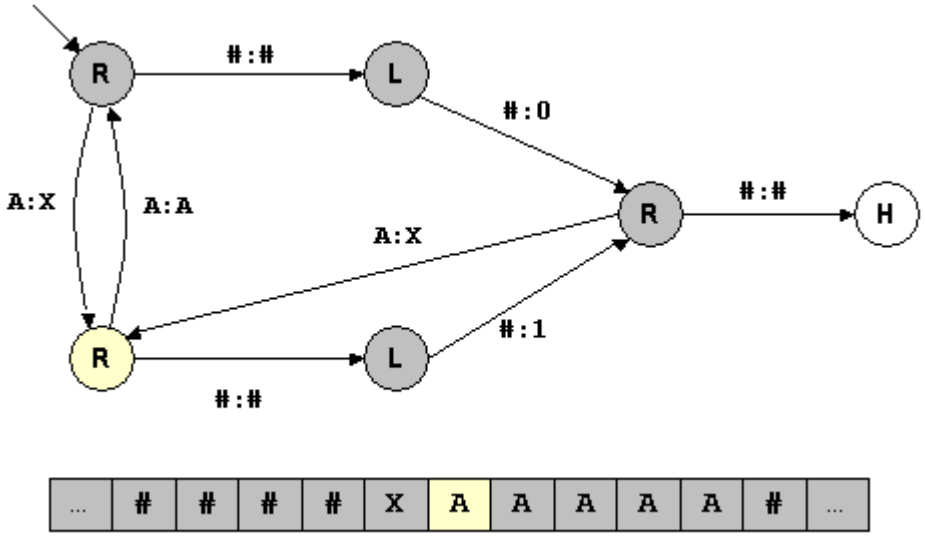
Some additional theory tidbits

- What if Python didn't have for/while loops, but had something else – e.g. "goto line i"
  - how does this affect what we can compute?
- Turing machines: important theoretical "universal computer"
  - memory: infinitely long "tape" of cells that can be blank or contain 0 or 1
  - program: a set of "states" and one fundamental operation:

    if state == q and tape cell == x, set cell to y, state to q', move

This simple computer can compute anything that is computable!





- A simulator
- A "real" one ☺

while vs. goto plus a simpler form of if statement that has no “body” – just a possible goto *linenum*

```
n = 0
while n < 100:
    sum = sum + n
    n = n + 1
print sum
```

```
1. n = 0
2. If ??? goto ???
3. sum = sum + n
4. n = n + 1
5. ???
6. print sum
```

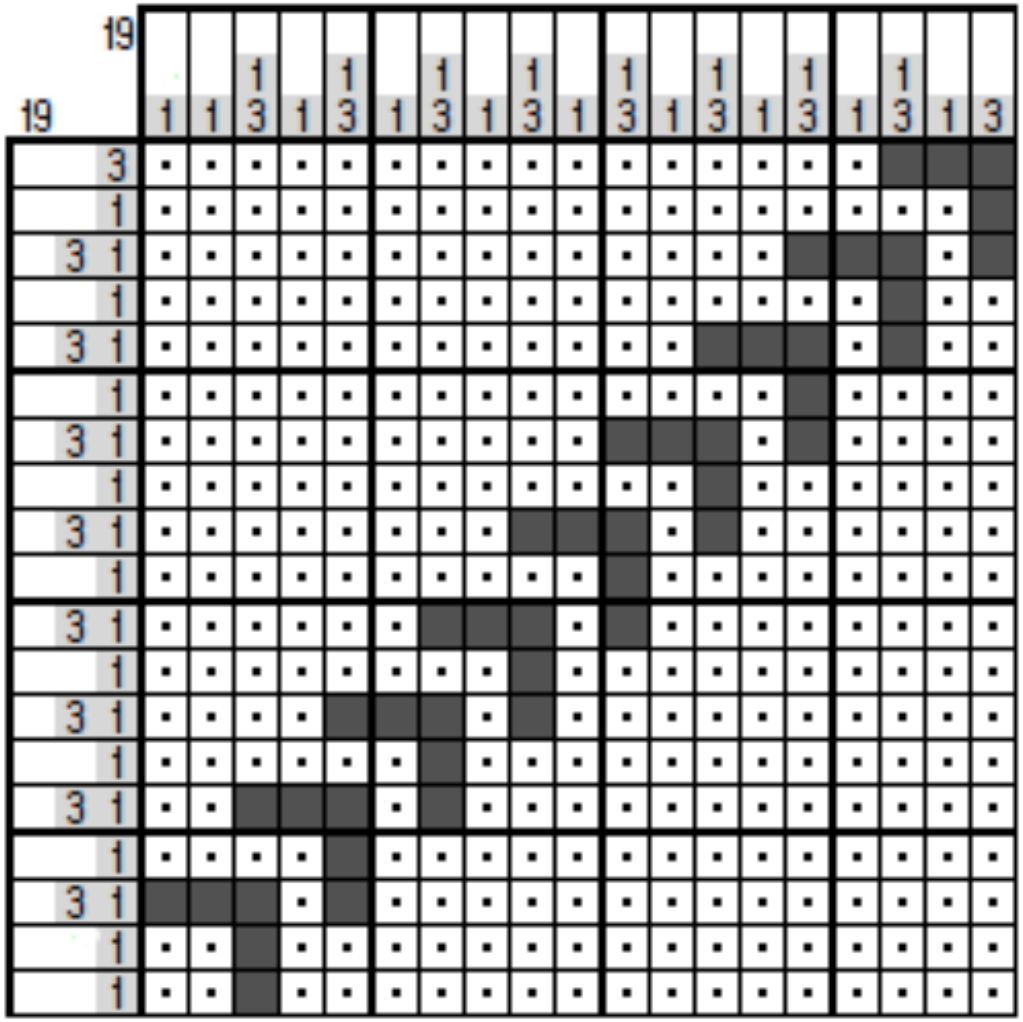
while vs. goto

```
n = 0
while n < 100:
    sum = sum + n
    n = n + 1
print sum
```

```
1. n = 0
2. if n >= 100 goto 6
3. sum = sum + n
4. n = n + 1
5. goto  2
6. print sum
```

More CS theory

- P vs. NP - the biggest unsolved problem in computer science
  - there are many problems that we don't know *efficient* algorithms for, and don't even know whether or not efficient ones even exist
  - Hundreds of real-world have been shown to be *NP-complete*
    - this means that if you solve any one of them, we can efficiently convert that solution into a solution for the rest of them. *Solve one efficiently* → *solve all efficiently!*
    - Longest path (but not shortest path), graph coloring, subset sum, many puzzles (Sudoku (!, ?, ?, solvers, hardest??, solver???, solver!, thoughts), minesweeper, etc.), satisfiability of logic formulas, nonograms (demo: nopuzzle.py)Traveling Salesperson (movie!?)
  - Solve it for extra practice this week == $1 million?
  - Demo: sudoku solver using "dancing links" algorithm sudoku.py



Human solvable via “global” thinking.
But not solvable by simple “local” thinking (single row/col constraints analysis) algorithm demonstrated In nopuzzle.py

https://webpbn.com/survey/dom.html

https://webpbn.com/survey/index.html

http://www.ijcsonline.com/IJCS/IJCS_2016_0303005.pdf

- Regarding puzzles:
  - Hundreds/thousands of logic puzzle types
    - Some cs (and other) research on puzzle design - lots of mobile games are puzzles – how are good puzzle levels made – via algorithm? by hand? Are hand-made ones better?
    - A recommended puzzle site (there are *many* good ones but this one provides an interesting variety of carefully hand-crafted ones, once per month, by CS PhD Pavel Curtis, who works at Microsoft)

      http://www.pavelspuzzles.com/aenigmas

The Halting Problem (later this week)

- it's important to know what we can and can't compute
- It turns out that we cannot create program that can check *all* other programs for infinite loops
- see, e.g., http://en.wikipedia.org/wiki/Halting_problem
- First: demonstrate that we can write programs that create and execute new programs/functions. testProgramOnInput.py
- Informal proof that we can't write doesItHalt
  - why can't we create fully correct doesItHalt function? (doesItHalt.py)
  - To see why, consider function test in doesItHaltTest.py