Program Design

Invasion Percolation: The Grid

5	3	7	2	6	1	1	3	4
8	5	6	5	7	2	3	6	2
2	5	8	7	5	5	6	5	9
5	2	6	4	9	3	9	6	5
4	6	8	8	5	9	7	3	9
7	6	4	5	1	2	6	8	5
5	4	2	5	8	5	5	5	8
5	7	5	1	5	3	8	5	5
4	5	1	9	7	8	6	5	1

Need:

- a random 2D grid

5	3	7	2	6	1	1	3	4
8	5	6	5	7	2	3	6	2
2	5	8	7	5	5	6	5	9
5	2	6	4	9	3	9	6	5
4	6	8	8	5	9	7	3	9
7	6	4	5	1	2	6	8	5
5	4	2	5	8	5	5	5	8
5	7	5	1	5	3	8	5	5
4	5	1	9	7	8	6	5	1

Need:

- a random 2D grid
- to mark cells

5	3	7	2	6	1	1	3	4
8	5	6	5	7	2	3	6	2
2	5	8	7	5	5	6	5	9
5	2	6	4	9	3	9	6	5
4	6	8	8		9	7	3	9
7	6	4	5			6	8	5
5	4	2	5	8		5	5	8
5	7	5	1	5	3	8	5	5
4	5	1	9	7	8	6	5	1

Don’t care about a cell’s value after it has been filled

5	3	7	2	6	1	1	3	4
8	5	6	5	7	2	3	6	2
2	5	8	7	5	5	6	5	9
5	2	6	4	9	3	9	6	5
4	6	8	8	−1	9	7	3	9
7	6	4	5	−1	−1	6	8	5
5	4	2	5	8	−1	5	5	8
5	7	5	1	5	3	8	5	5
4	5	1	9	7	8	6	5	1

So use any value that can't ever be a real cell value to mark filled cells

Note: we're using integers as:

Note: we're using integers as:

- Actual data values

Note: we're using integers as:

- Actual data values

- Flags to represent cell states

Note: we're using integers as:

- Actual data values

- Flags to represent cell states

It's simple to do...

Note: we're using integers as:

- Actual data values

- Flags to represent cell states

It's simple to do...

...but if we ever get data whose values are those we've been using as flags, our program will interpret them as flags

Note: we're using integers as:

- Actual data values

- Flags to represent cell states

It's simple to do...

...but if we ever get data whose values are those we've been using as flags, our program will interpret them as flags

This kind of error can be very hard to track down

5	3	7	2	6	1	1	3
8	5	6	5	7	2	3	6
2	5	8	7	5	5	6	5
5	2	6	4	9	3	9	6
4	6	8	8	−1	9	7	3
7	6	4	5	−1	−1	6	8

Are grids always square?

5	3	7	2	6	1	1	3
8	5	6	5	7	2	3	6
2	5	8	7	5	5	6	5
5	2	6	4	9	3	9	6
4	6	8	8	−1	9	7	3
7	6	4	5	−1	−1	6	8

Are grids always square?

Are they always odd × odd (so that there is a unique center square)?

5	3	7	2	6	1	1	3
8	5	6	5	7	2	3	6
2	5	8	7	5	5	6	5
5	2	6	4	9	3	9	6
4	6	8	8	−1	9	7	3
7	6	4	5	−1	−1	6	8

How general should we make the first version of our program?

5	3	7	2	6	1	1	3
8	5	6	5	7	2	3	6
2	5	8	7	5	5	6	5
5	2	6	4	9	3	9	6
4	6	8	8	−1	9	7	3
7	6	4	5	−1	−1	6	8

How general should we make the first version of our program?

"Don't build it until you need it."

5	3	7	2	6	1	1	3
8	5	6	5	7	2	3	6
2	5	8	7	5	5	6	5
5	2	6	4	9	3	9	6
4	6	8	8	−1	9	7	3
7	6	4	5	−1	−1	6	8

How general should we make the first version of our program?

"Don't build it until you need it."

vs.

"A week of hard work can sometimes save you an hour of thought."

Like many generalizations, these rules are:

Like many generalizations, these rules are:

- True

Like many generalizations, these rules are:

- True

- Not particularly useful

Like many generalizations, these rules are:

- True

- Not particularly useful

Knowing what rules to apply when comes with experience

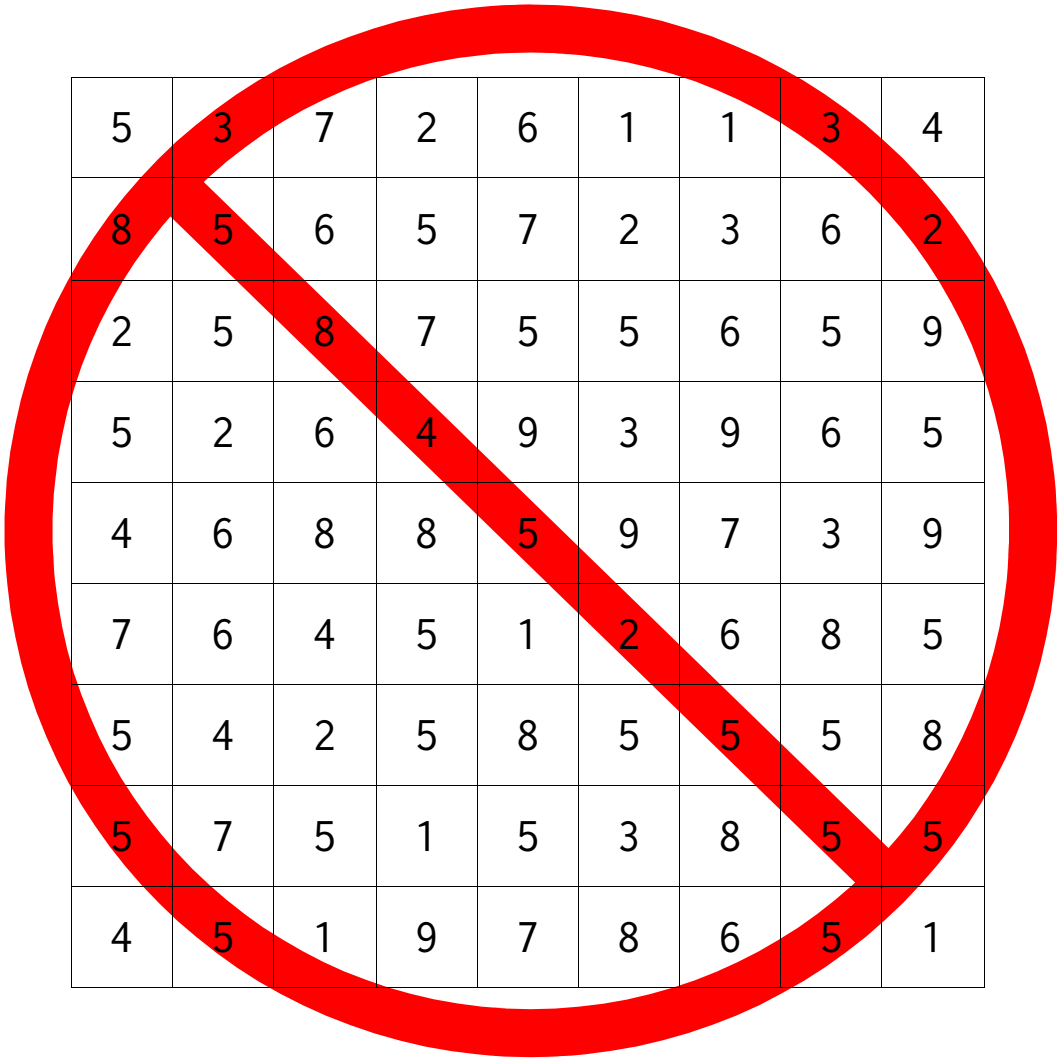
Like many generalizations, these rules are:

- True

- Not particularly useful

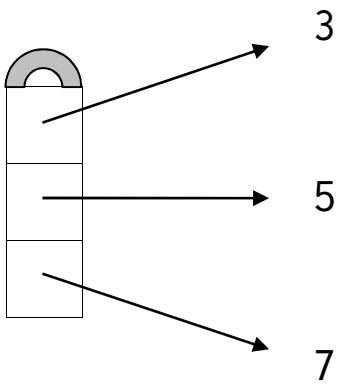
Knowing what rules to apply when comes with experience

The only way to get experience is to work through many examples



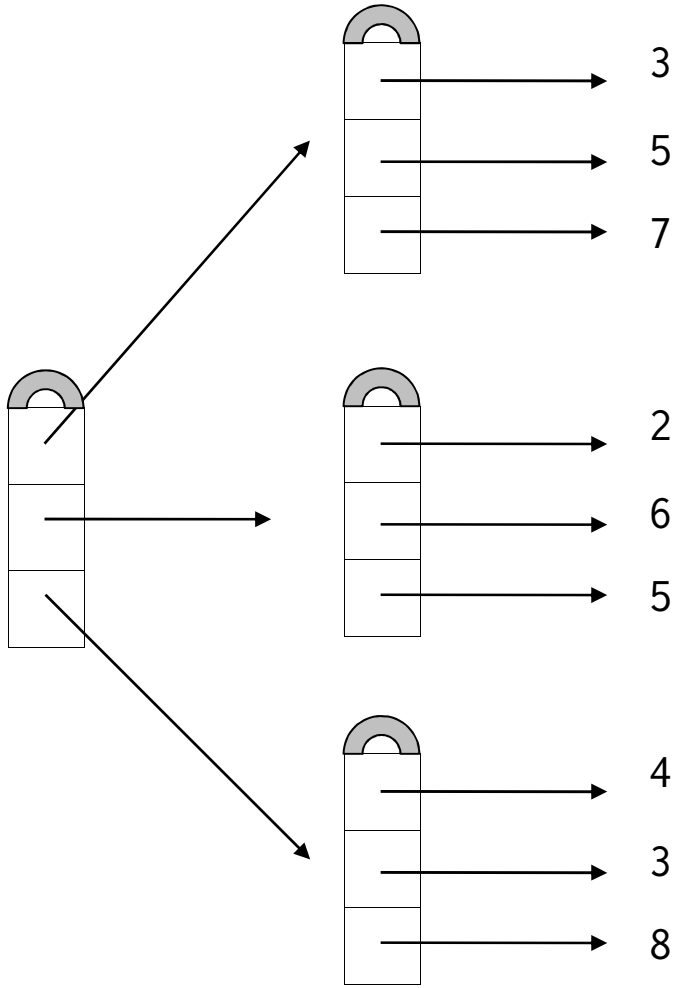
5	3	7	2	6	1	1	3	4
8	5	6	5	7	2	3	6	2
2	5	8	7	5	5	6	5	9
5	2	6	4	9	3	9	6	5
4	6	8	8	5	9	7	3	9
7	6	4	5	1	2	6	8	5
5	4	2	5	8	5	5	5	8
5	7	5	1	5	3	8	5	5
4	5	1	9	7	8	6	5	1

Problem: Python doesn't actually have 2D arrays



But it does have

1D lists



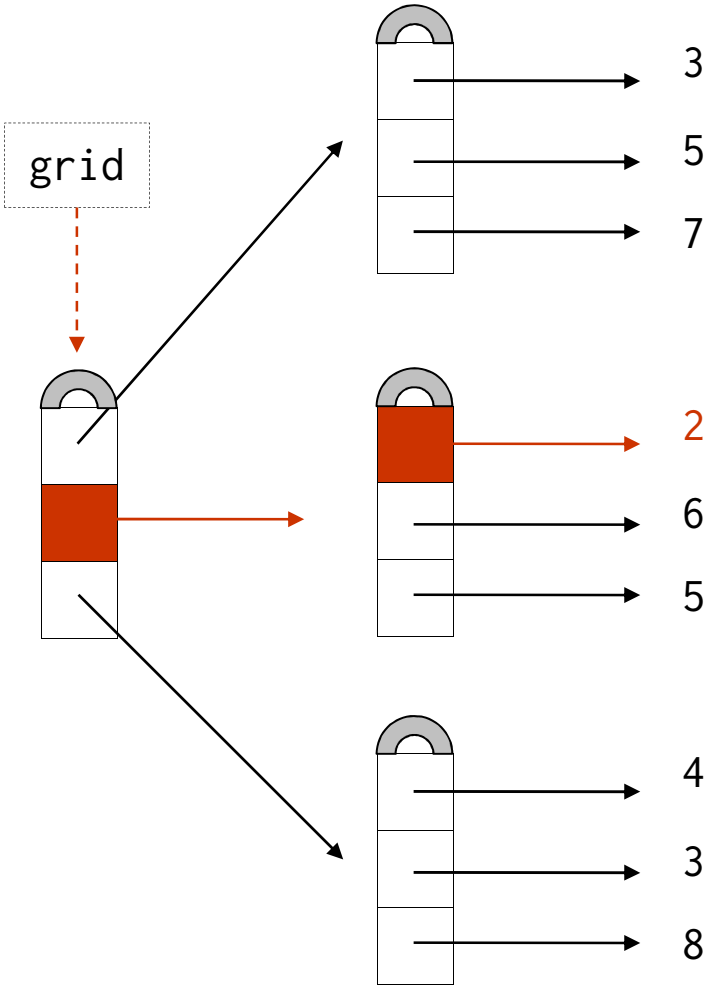
But it does have

1D lists

Which can refer to

other lists

```
grid[1][0] == 2
```



But it does have

1D lists

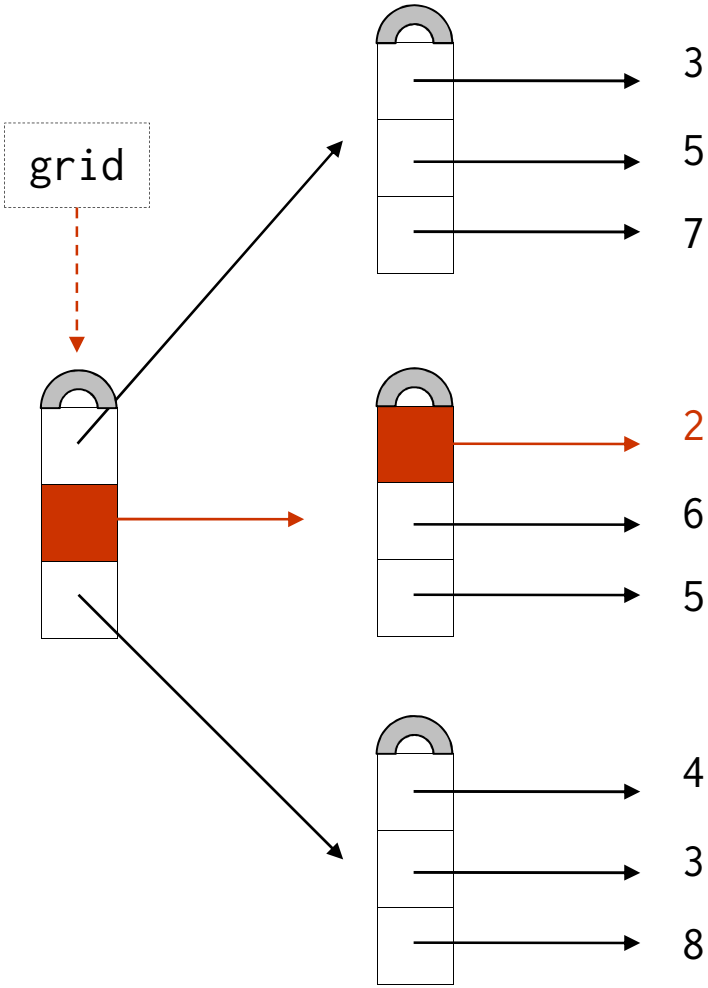
Which can refer to

other lists

This gives us

double subscripts

```
grid[1][0] == 2
```



But it does have

1D lists

Which can refer to

other lists

This gives us

double subscripts

...which is really what

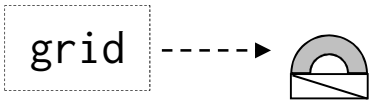
we mean by

“two-dimensional”

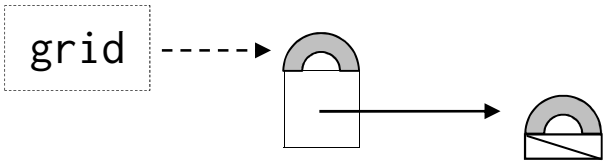
```
# Create an NxN grid of random integers in 1..Z.
assert N > 0, "Grid size must be positive"
assert N%2 == 1, "Grid size must be odd"
grid = []
for x in range(N):
  grid.append([])
  for y in range(N):
    grid[-1].append(1) # FIXME: need a random value
```

```
# Create an NxN grid of random integers in 1..Z.
assert N > 0, "Grid size must be positive"
assert N%2 == 1, "Grid size must be odd"
grid = []
for x in range(N):
  grid.append([])
  for y in range(N):
    grid[-1].append(1) # FIXME: need a random value
```

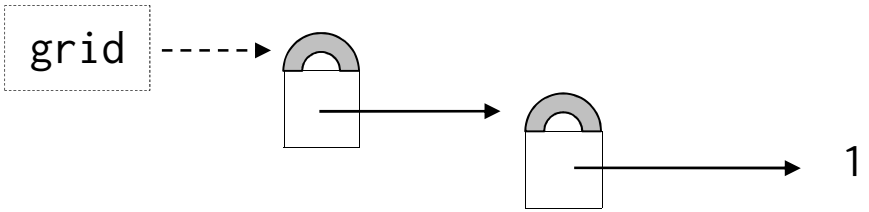
```
# Create an NxN grid of random integers in 1..Z.
assert N > 0, "Grid size must be positive"
assert N%2 == 1, "Grid size must be odd"
grid = []
for x in range(N):
  grid.append([])
  for y in range(N):
    grid[-1].append(1) # FIXME: need a random value
```



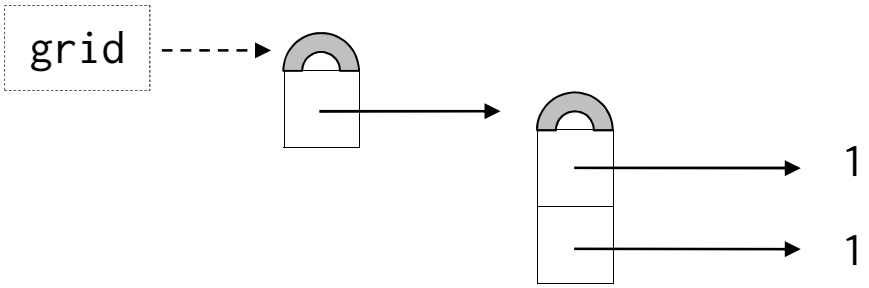
```
# Create an NxN grid of random integers in 1..Z.
assert N > 0, "Grid size must be positive"
assert N%2 == 1, "Grid size must be odd"
grid = []
for x in range(N):
  grid.append([])
  for y in range(N):
    grid[-1].append(1) # FIXME: need a random value
```



```
# Create an NxN grid of random integers in 1..Z.
assert N > 0, "Grid size must be positive"
assert N%2 == 1, "Grid size must be odd"
grid = []
for x in range(N):
  grid.append([])
  for y in range(N):
    grid[-1].append(1) # FIXME: need a random value
```



```
# Create an NxN grid of random integers in 1..Z.
assert N > 0, "Grid size must be positive"
assert N%2 == 1, "Grid size must be odd"
grid = []
for x in range(N):
  grid.append([])
  for y in range(N):
    grid[-1].append(1) # FIXME: need a random value
```



software carpentry

created by

Greg Wilson

May 2010