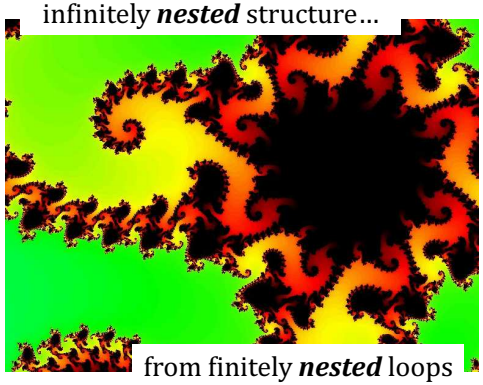
CS 5 Today

infinitely ***nested*** structure…



from finitely ***nested*** loops

Homework 9 ***Loops!***

Reading:
Section 5.4

Thinking in *loops*

for

```
for x in range(42):
    print(x)
```

while

```
x = 1
while x < 42:
    print(x)
    x *= 2
```

What are the design differences between these two types of Python loops?

Thinking in *loops*

for

definite iteration

For a **known** list or # of iterations

while

indefinite iteration

For an **unknown** number of iterations

Loops: `for` or `while`?

`pi_one(e)` — *e == how close to π we need to get*

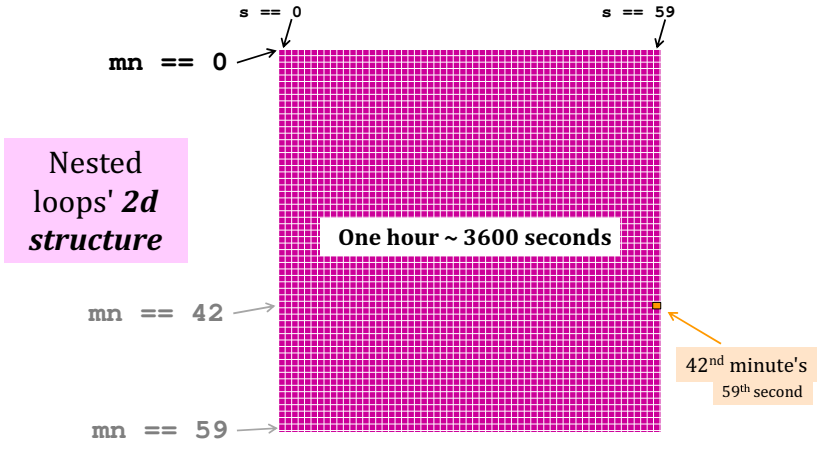
`pi_two(n)` — *n == number of darts to throw*

Which function will use which kind of loop?

Nested loops

Life clock


```
for y in range(84):
  for m in range(12):
    for d in range(f(m,y)):
      for h in range(24):
        for min in range(60):
          for s in range(60):
            tick()
```




```
for min in range(60):
  for s in range(60):
    tick()
```

Creating 2d structure

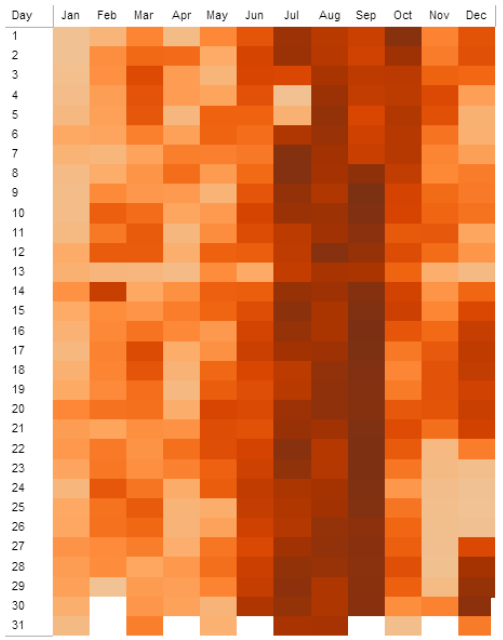
```
for row in range(3):
    for col in range(4):
        print('#', end = '')
    print()
```

row \ col	0	1	2	3
0				
1				
2				

```
row =
    col =
    col =
    col =
    col =

row =
    col =
    col =
    col =
    col =

row =
    col =
    col =
    col =
    col =
```



Nested loops' *2d structure*

```
for d in range(f(m)):
  for m in range(1,13):
    num_bdays(m,d)
```

What trends appear in this birthday data?

How might we be suspicious of the ***fairness*** of this data?!

Data represents the # of babies born in the United States between 1973 and 1999

Source: Amitabh Chandra Harvard University via NYTimes.com

viewz.blogspot.com/2012/05/how-common-is-your-birthday-find-out.html

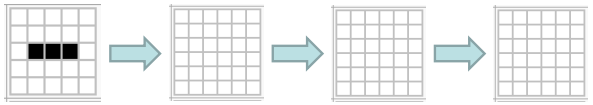
The Game of Life

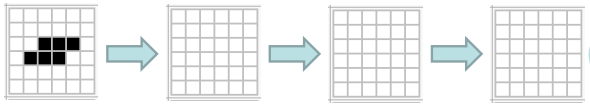
1. Any live cell with fewer than two neighbors dies of loneliness.

2. Any live cell with more than three neighbors dies of overcrowding.

3. Any live cell with two or three neighbors lives, unchanged, to the next generation.

4. Any dead cell with exactly three neighbors comes to life.





The inventor did this by hand. It's a pain! You try! ☺

Never start coding before you can do "it" by hand!

2-D "Arrays"

```
>>> A = [[0, 0, 0, 1], [1, 1, 0, 0], [0, 0, 0, 1]]
>>> A = [[0, 0, 0, 1],
         [1, 1, 0, 0],
         [0, 0, 0, 1]]
>>> A[0][3]
???
```

Shallow Copy

```
>>> A = [1, 2, 3, 4]
>>> B = A
>>> B[0] = 42
>>> A[0]
_____ ←

def foo():
  L = [1, 2, 3, 4]
  bar(L)
  return L

def bar(List):
    List[0] = 42

>>> print(foo())
_____ ←
```

Deep Copy

```
def foo2():
  L = [1, 2, 3, 4]
  M = bar2(L)
  print(L)
  print(M)

def bar2(List):
    return list(map(lambda X: X+1, List))

>>> foo2()
[1, 2, 3, 4]
_____ ←
```

Match!

What code creates the fourth one?

* and ** are extra!

Name(s): _____

Quiz

A

```
for r in range(3):          [0,1,2]
  for c in range(6):
    if c > r:               [0,1,2,3,4,5]
      print('#', end='')
    else:
      print(' ', end='')
  print()
```

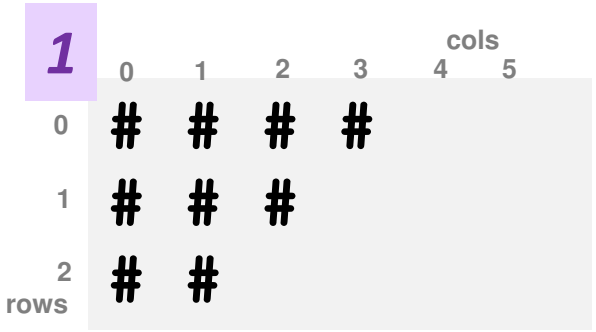
B

```
for r in range(3):
  for c in range(6):
    if c%2 == 1:
      print('#', end='')
    else:
      print(' ', end='')
  print()
```

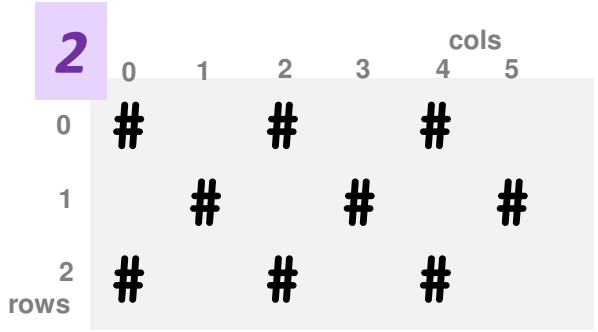
C

```
for r in range(3):
  for c in range(6):
    if c%2 == r%2:
      print('#', end='')
    else:
      print(' ', end='')
  print()
```

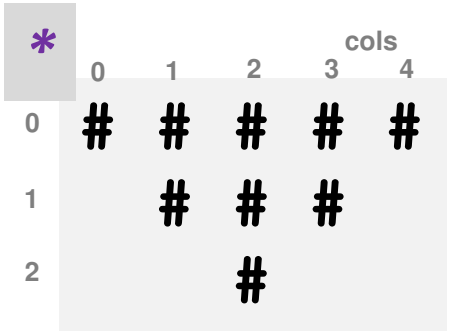
1



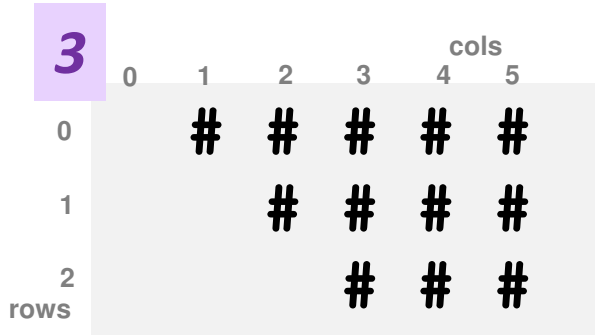
2



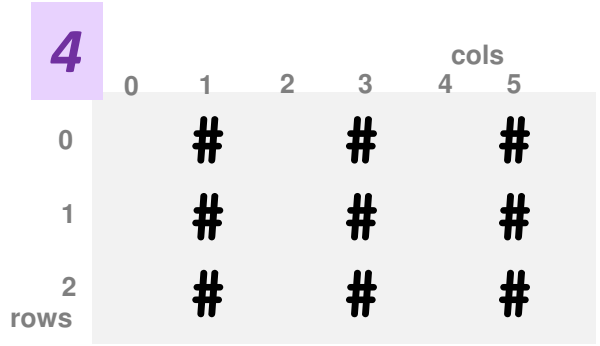
***



3



4



****