11. More on While and Boolean-Valued Functions

Topics:

Reasoning about While Loops

Designing Boolean-Valued Functions

Four Examples

1. Random Walk

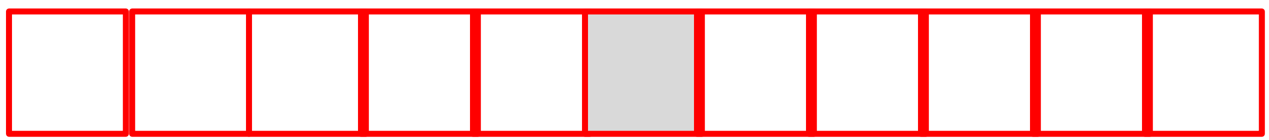
2. Fibonacci numbers and the Golden Ratio

3. A Spiral Problem

4. Detecting streaks in a coin toss sequence

Random Walk

Random Walk

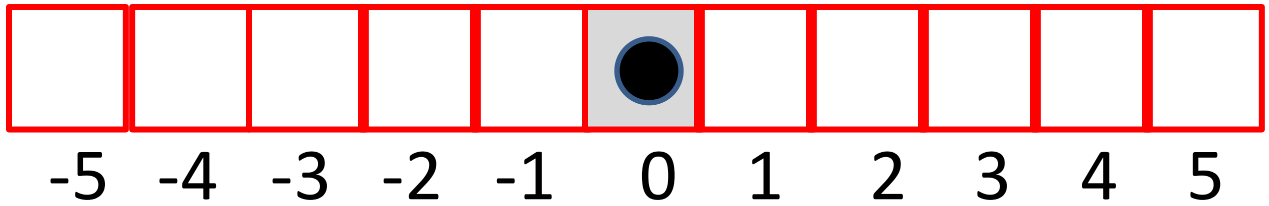


Tiles 1x1
Middle tile has center (0,0)
Robot starts at center tile
Hops according to coin flip
Heads: Hop left
Tails: Hop right
Simulation over when robot hops off runway

Random Walk

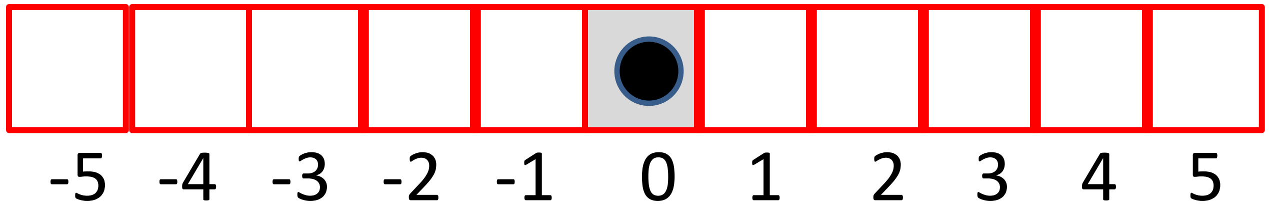
```
from random import randint as randi
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



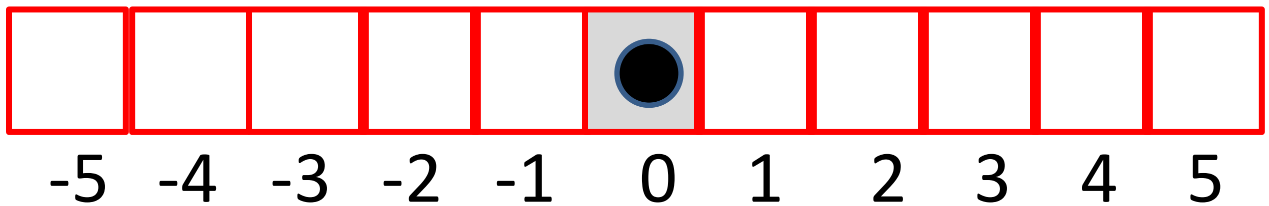

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



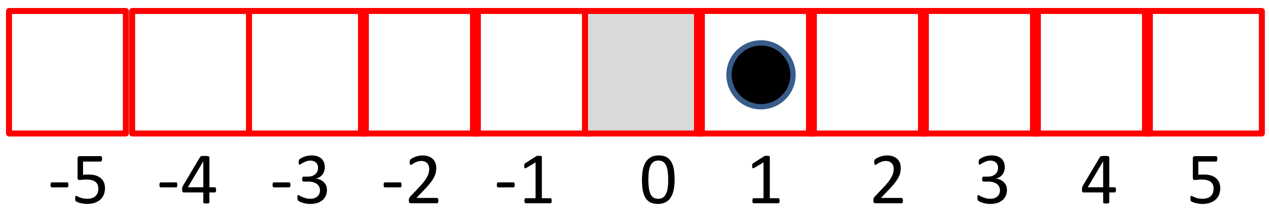

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



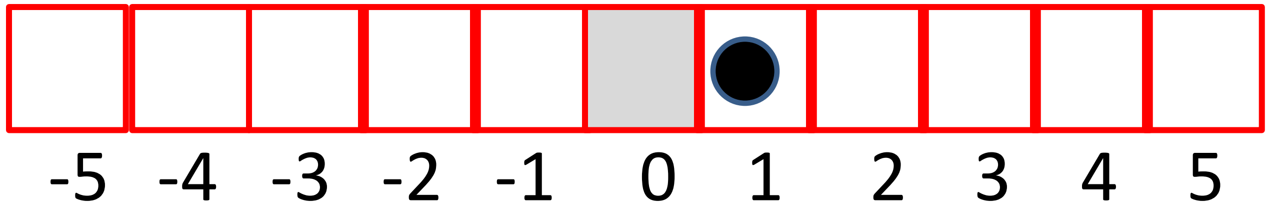

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



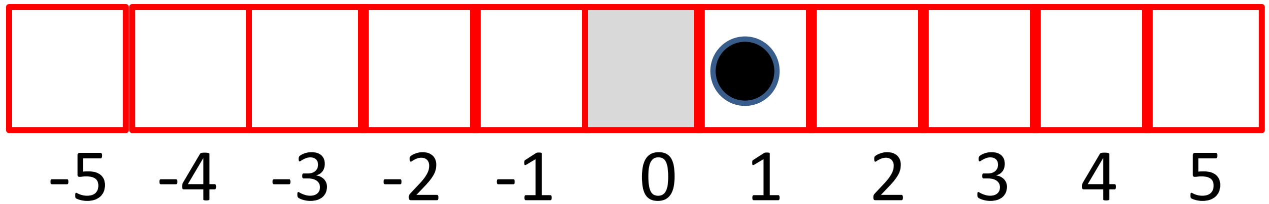

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



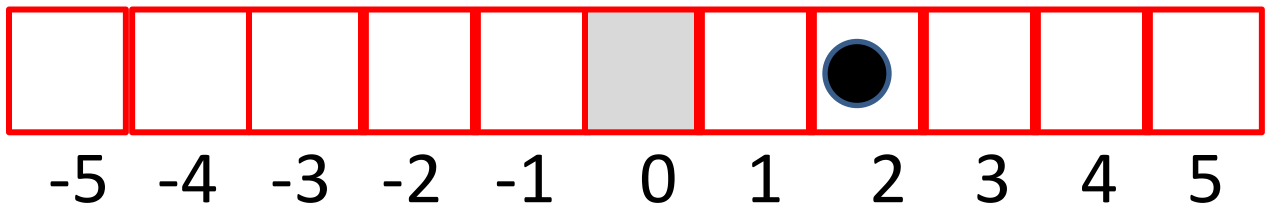

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



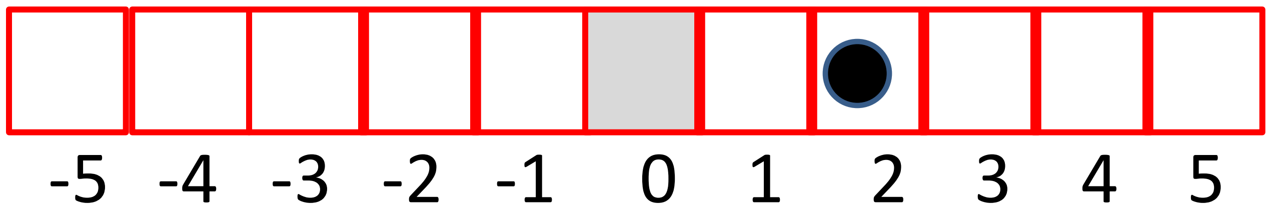

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



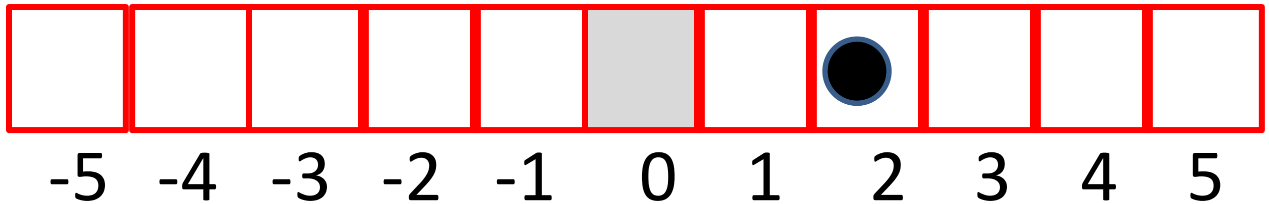

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



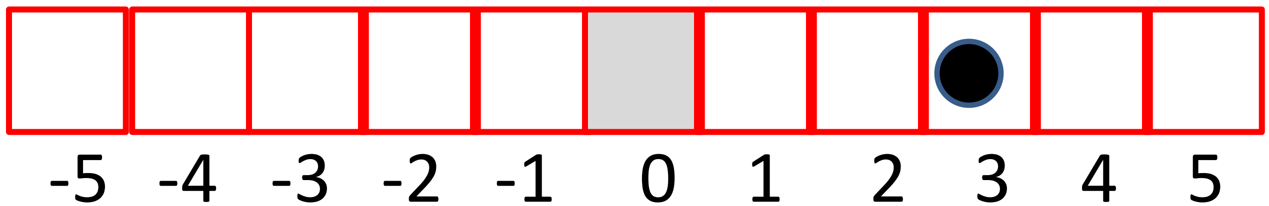

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



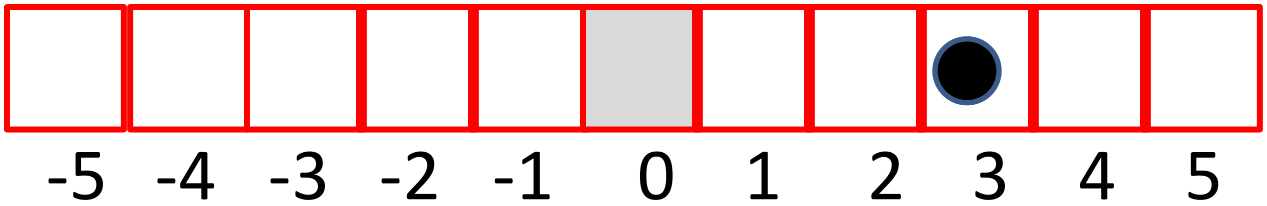

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



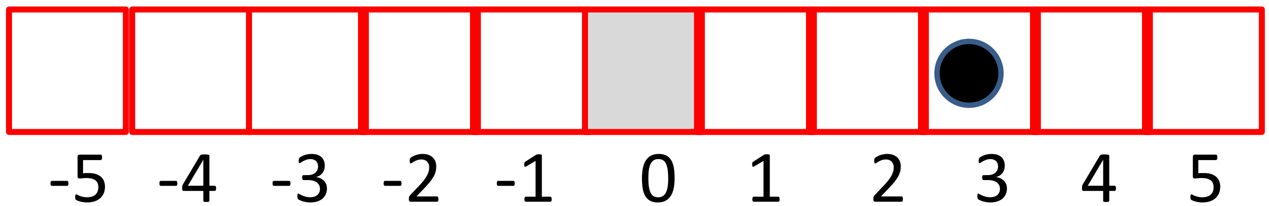

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



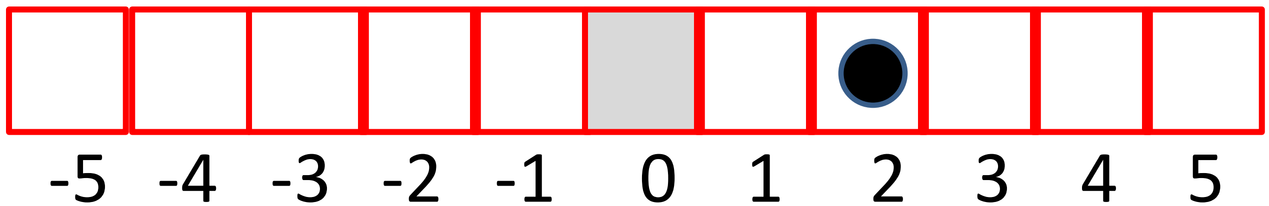

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



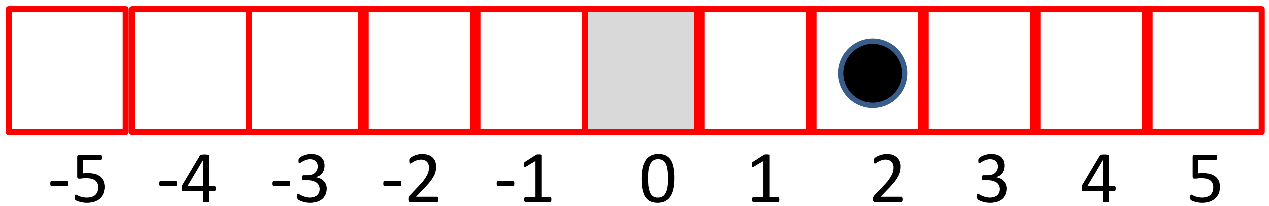

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



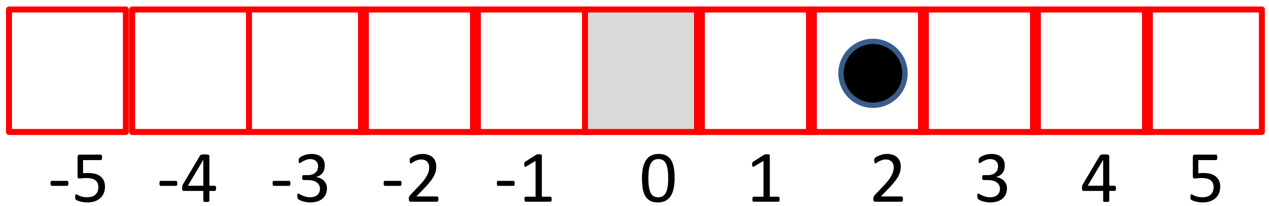

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



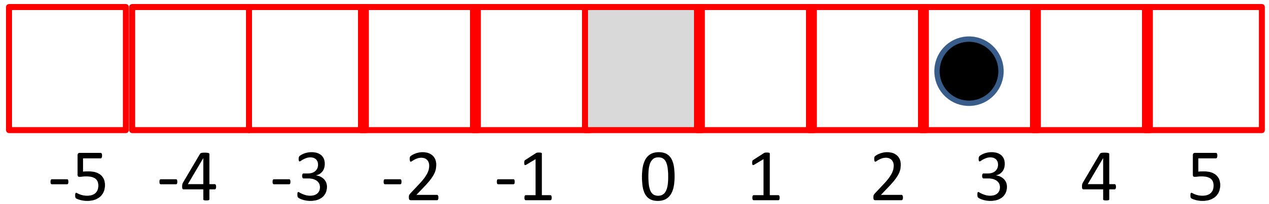

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



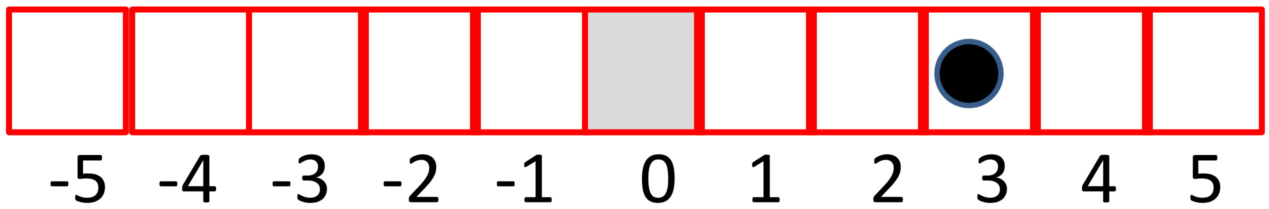

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



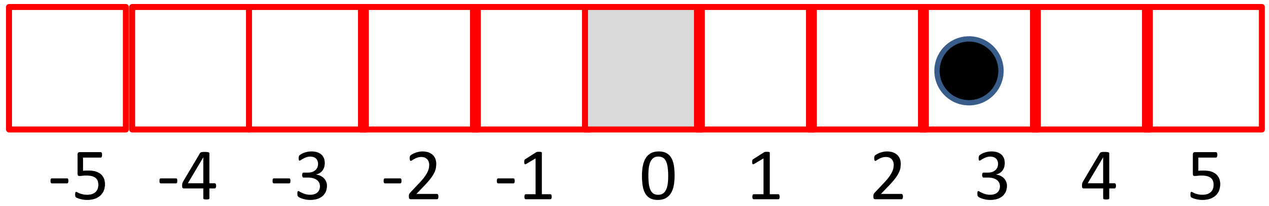

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



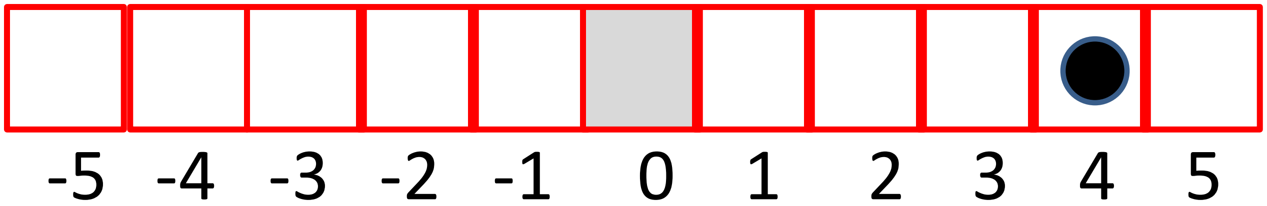

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



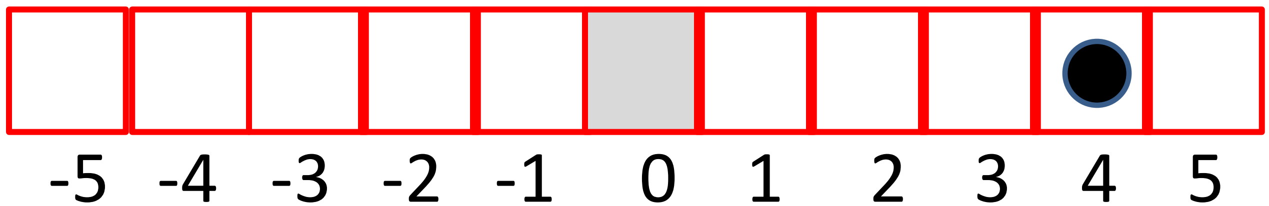

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



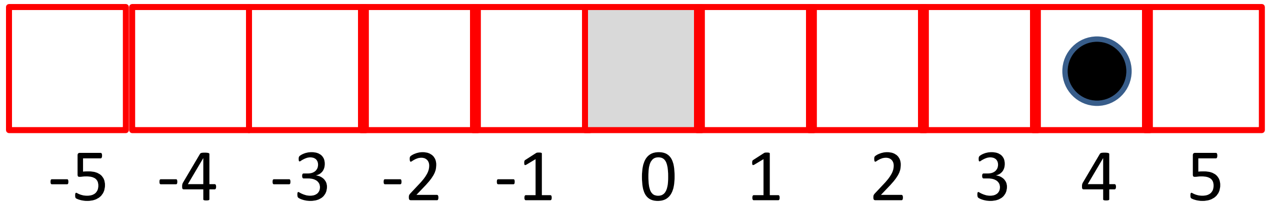

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



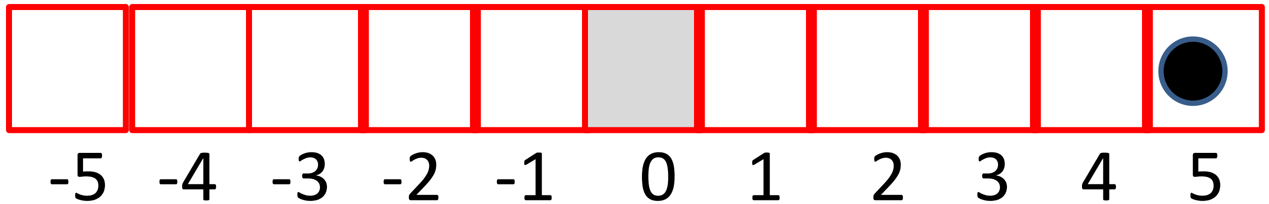

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



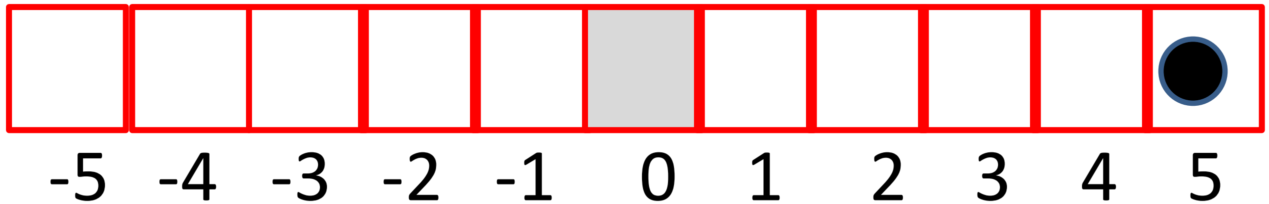

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



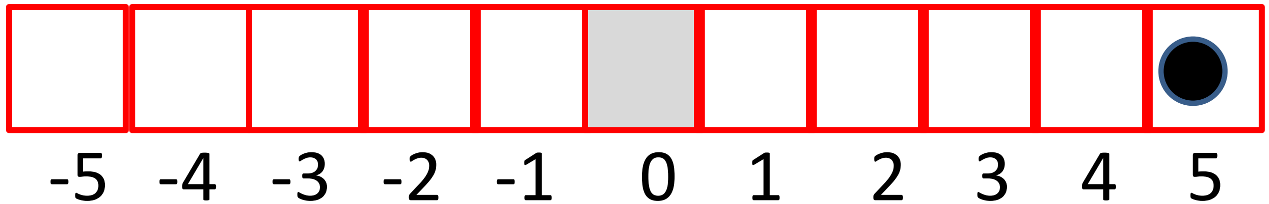

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk



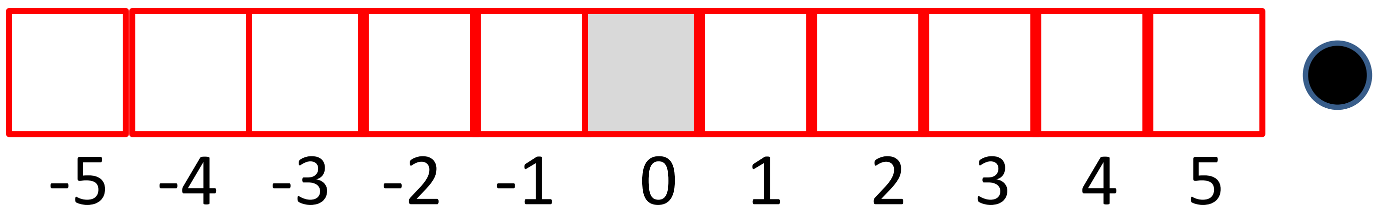

```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk




```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk




```
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

2. Fibonacci Numbers and the Golden Ratio

Fibonacci Numbers and the Golden Ratio

Here are the first 12 Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

The Fibonacci ratios 1/1, 2/1, 3/2, 5/3, 8/5 get closer and closer to the "golden ratio"

phi = (1 + sqrt(5))/2

Fibonacci Ratios 2/1, 3/2, 5/3, 8/5

1

2

2

3

3

5

5

8

Generating Fibonacci Numbers

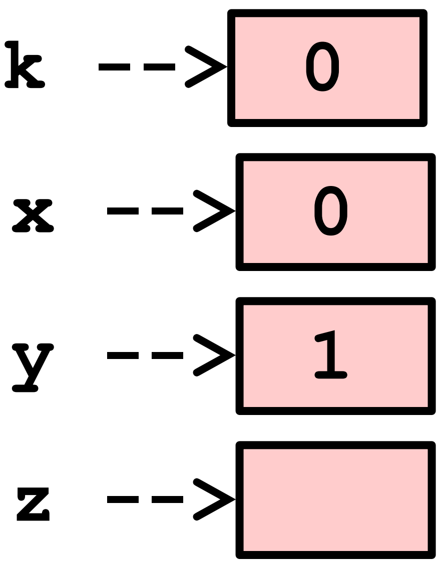
Here are the first 12 Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

Starting here, each one is the sum of its two predecessors

Generating Fibonacci Numbers

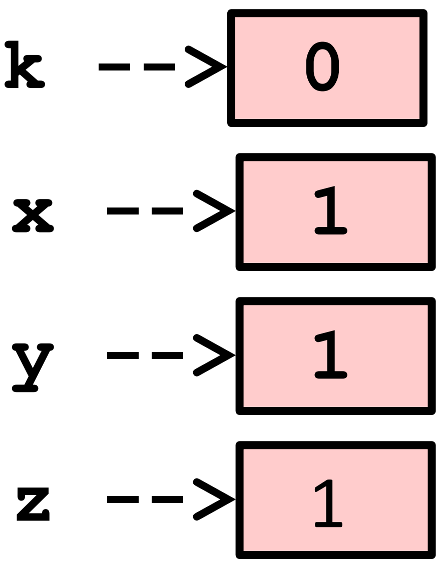
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144




```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144




```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

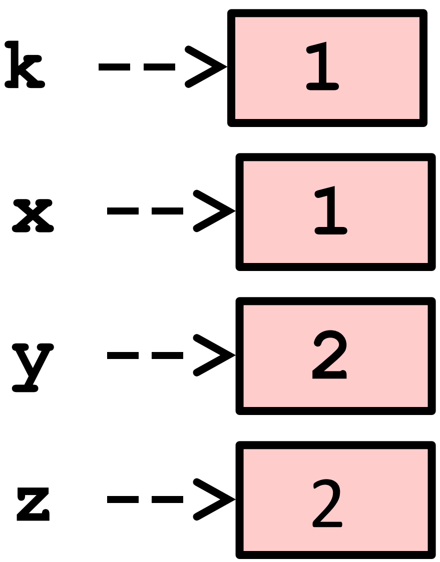
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k  --> 1
x  --> 1
y  --> 1
z  --> 1
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144




```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

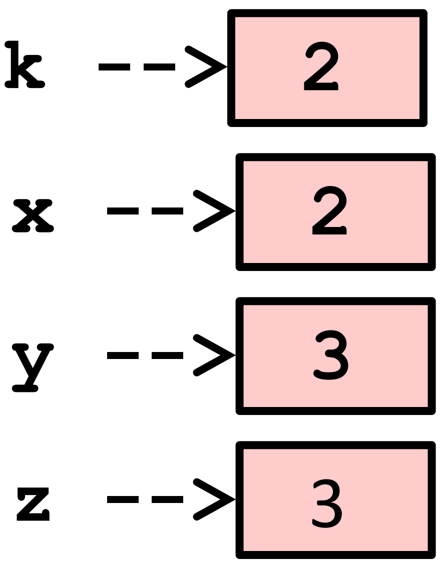
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k --> 2
x --> 1
y --> 2
z --> 2
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144




```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k  --> 3
x  --> 2
y  --> 3
z  --> 3
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k  -->  3
x  -->  3
y  -->  5
z  -->  5
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

```
x = 0
print x
y = 1
print y
for k in range(6):
    z = x+y
    x = y
    y = z
    print z
```

```
 0
 1
 1
 2
 3
 5
 8
13
```

Generating Fibonacci Numbers

```
x = 0
print x
y = 1
print y
for k in range(6):
    z = x+y
    x = y
    y = z
    print z
```

```
x = 0
print x
y = 1
print y
k = 0
while k<6:
    z = x+y
    x = y
    y = z
    print z
    k = k+1
```

Print First Fibonacci Number >= 1000000

```
x = 0
y = 1
z = x + y
while y < 1000000:
    x = y
    y = z
    z = x + y
print y
```

Print First Fibonacci Number >= 1000000

```
past = 0
current = 1
next = past + current
while current < 1000000:
    past = current
    current = next
    next = past + current
print current
```

```
1346269
```

Print First Fibonacci Number >= 1000000

```
past = 0
current = 1
next = past + current
while current < 1000000:
   past = current
   current = next
   next = past + current
print current
```

Reasoning. When the while loop terminates, it will be the first time that `current>= 1000000` is true. By print out current we see the first fib >= million

Print Largest Fibonacci Number < 1000000

```
past = 0
current = 1
next = past + current
while next <= 1000000:
    past = current
    current = next
    next = past + current
print current
```

832040

Print Largest Fibonacci Number < 1000000

```
past = 0
current = 1
next = past + current
while next < 1000000:
    past = current
    current = next
    next = past + current
print current
```

Reasoning. When the while loop terminates, it will be the first time that `next>= 1000000` is true. Current has to be < 1000000. And it is the largest fib with this property

Fibonacci Ratios

```
past = 0
current = 1
next = past + current
while next <= 1000000:
    past = current
    current = next
    next = past + current
    print next/current
```

```
1.000000000000
2.000000000000
1.500000000000
1.666666666667
1.600000000000
1.625000000000
1.615384615385
1.619047619048
1.617647058824
1.618181818182
1.617977528090
1.618055555556
1.618025751073
1.618037135279
1.618032786885
       :
```

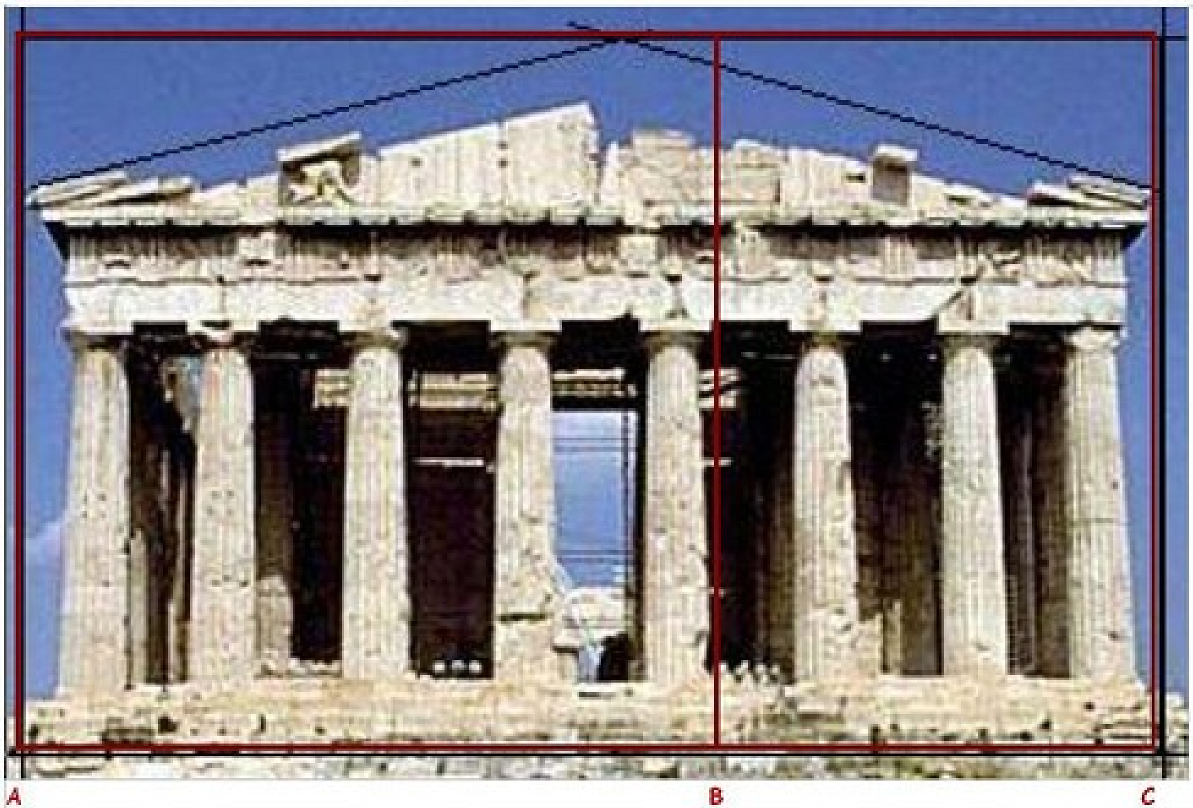
Heading towards the
Golden ratio = (1+sqrt(5))/2

Fibonacci Ratios

```
past = 0
current = 1
next = past + current
k = 1
phi = (1+math.sqrt(5))/2
while abs(next/current - phi) > 10**-9
    past = current
    current = next
    next = past + current
    k = k+1
print k,next/current
```

23 1.618033988749

Most Pleasing Rectangle



1

(1+sqrt(5))/2

3. A Spiral Problem

A Spiral Problem

Recall:

`DrawSpiral(N,t,c1,c2,c3)`

draws a spiral and

`SpiralRadius(N,t)`

computes its radius.

The Gist of SpiralRadius

```
xc = 0; yc = 0; r = 0
for k in range(N):
    theta = (k*t)*math.pi/180
    L = k+1
    # (xc,yc) = end of the kth edge
    xc = xc + L*math.cos(theta)
    yc = yc + L*math.sin(theta)
    dist = math.sqrt(xc**2+yc**2)
    r = max(r,dist)
return r
```

The Gist of SpiralRadius

```
xc = 0; yc = 0; r = 0
for k in range(N):
    theta = (k*t)*math.pi/180
    L = k+1
    # (xc,yc) = end of kth edge
    xc = xc + L*math.cos(theta)
    yc = yc + L*math.sin(theta)
    dist = math.sqrt(xc**2+yc**2)
    r = max(r,dist)
return r
```

The Heading

For the k-th edge, here is the heading in radians:

```
theta = (k*t)*math.pi/180
```

`t` is the turn angle in degrees

The Gist of SpiralRadius

```
xc = 0; yc = 0; r = 0
for k in range(N):
    theta = (k*t)*math.pi/180
    L = k+1
    # (xc,yc) = end of kth edge
    xc = xc + L*math.cos(theta)
    yc = yc + L*math.sin(theta)
    dist = math.sqrt(xc**2+yc**2)
    r = max(r,d)
return r
```

The Ending Endpoint

Before: (xc,yc) is where the kth edge starts

```
xc = xc + L*math.cos(theta)
yc = yc + L*math.sin(theta)
```

After: (xc,yc) is where the kth edge ends

The Gist of SpiralRadius

```
xc = 0; yc = 0; r = 0
for k in range(N):
    theta = (k*t)*math.pi/180
    L = k+1
    # (xc,yc) = end of the kth edge
    xc = xc + L*math.cos(theta)
    yc = yc + L*math.sin(theta)
    dist = math.sqrt(xc**2+yc**2)
    r = max(r,d)
return r
```

Computing the max Distance

Is the end of the kth edge further away from (0,0) than all previous endpoints?

```
dist = math.sqrt(xc**2+yc**2)
r = max(r,d)
```

```
dist = math.sqrt(xc**2+yc**2)
if dist > r:
        r = dist
```

A Reverse Problem

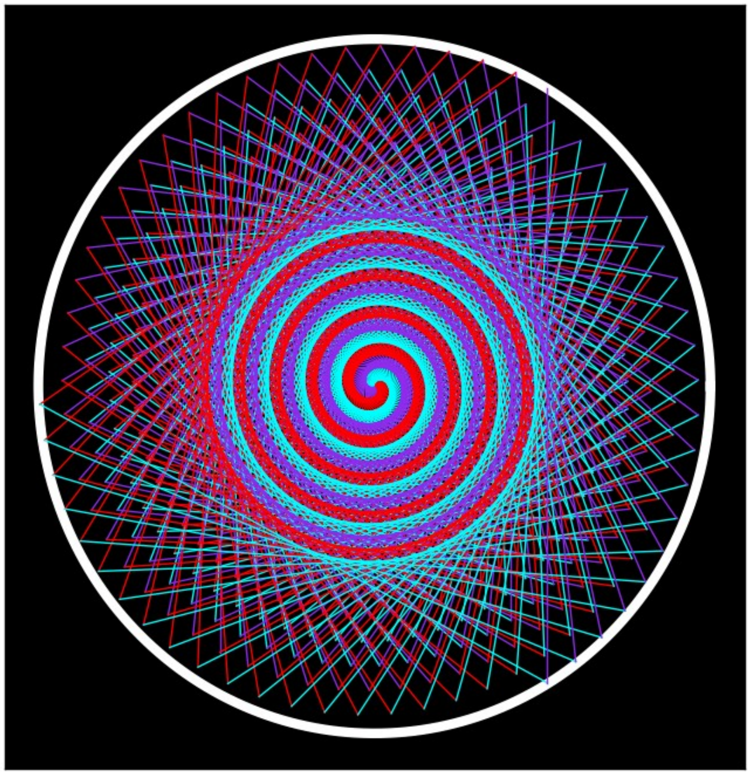
Given the turn angle **t** and a radius **r**, what is the largest **N** so that

```
DrawSpiral(N,t,c1,c2,c3)
```

fits inside the circle

x**2 + y**2 = r**2

Example

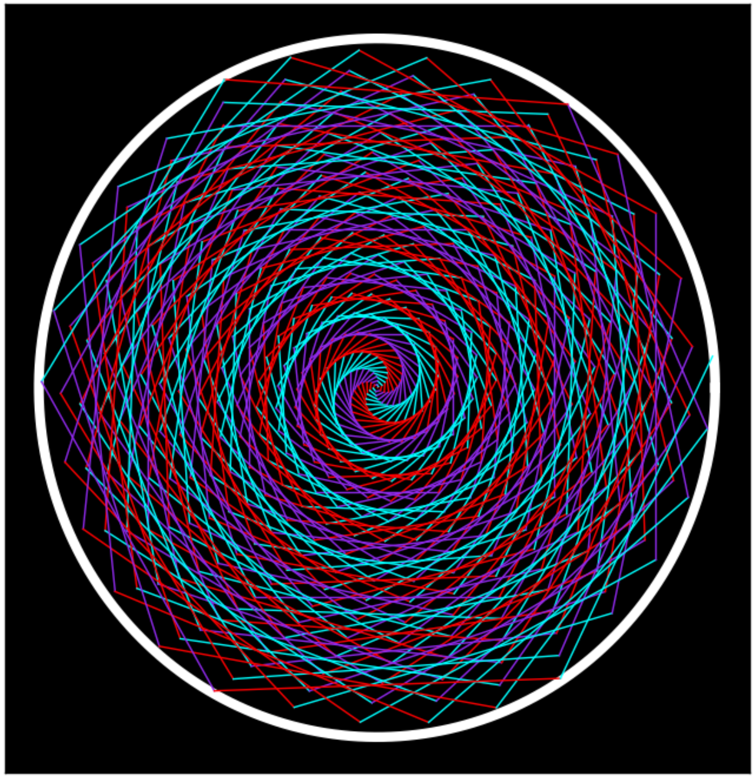


The circle has radius
r = 500.

`DrawSpiral(513,62,...)`

just fits inside

Example

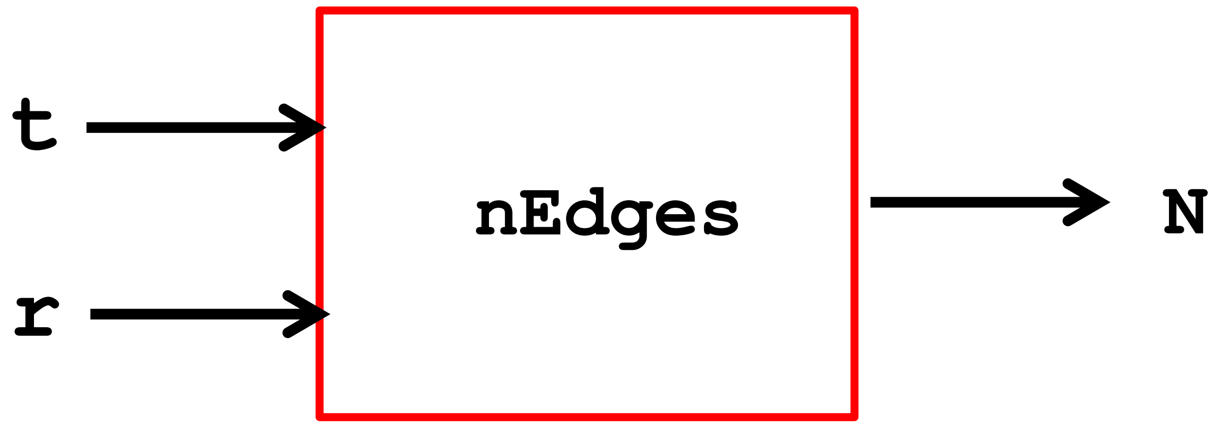


The circle has radius
r = 500.

`DrawSpiral(856,162,...)`

just fits inside

Let's Design a Function that Returns This Integer




```
t = turn angle
r = radius
N = max number edges so spiral radius <= r
```

The Body of `nEdges`

```
k = 0  # Index of current edge
Compute endpoint distance to (0,0)
while endpoint inside circle
    k = k+1
    Compute endpoint dist to (0,0)
N = k
return N
```

```
k = 0
xc = 1
yc = 0
d = math.sqrt(xc**2 + yc**2)
while d<=r:
    k = k+1
    theta = (k*t)*math.pi/180
    xc = xc + (k+1)*math.cos(theta)
    yc = yc + (k+1)*math.sin(theta)
    d = math.sqrt(xc**2 + yc**2)
return k-1
```

4. Streaks in a Coin Toss Sequence

Coin Toss Strings

S is a coin toss string if it is made up of H's and T'

`s = 'HHTHTTTHTHHTHTTT'`

Streaks

```
s = 'HHTHTTTHTHHTHTTT'
```

```
s[0:2]        a  length-2 streak
```

Streaks

`s = 'HHTHTTTHTHHTHTTT'`

`s[4:7]` a length-3 streak

Streaks

```
s = `HHTHTTTHTHHTTTTT'
```

```
s[12:17]        a length-5 streak
```

Streak Definition

`s[k:k+n]` is a length-n streak if

```
(1) k+n <= len(s)
and
(2) It is either all T's or all H's
and
(3) If there is a character before
     s[k], it is different from s[k].
and
(4) If there is a character after
     s[k+n], it is different from s[k+n].
```

Streaks

s = 'HHTHTTTHTHHTHTTT'

s[5:7]   is NOT a length-2 streak

Rule 3: If there is a character before s[k], it is different from s[k].

isStreak(s,k,n)

```
t = s[k:k+n]
if k+n>len(s):
    return False
elif t.count('H')<n and t.count('T')<n:
    return False
elif k>0 and (s[k-1]==s[k]):
    return False
elif (k+n<len(s)) and (s[k+n-1]==s[k+n]):
    return False
else:
    return True
```

A function can have more than one return

Using isStreak to Find Streaks

```
s = 'HHTHTTTHTHHHTHTT'

k     isSTreak(s,k,3)
------------------------------------------
0          False
```

Using isStreak to Find Streaks

```
s = 'HHTHTTTHTHHHTHTT'
```

```
k     isSTreak(s,k,3)
---------------------------------------
0          False
1          False
```

Using `isStreak` to Find Streaks

`s = 'HHTHTTTHTHHHTHTT'`

k	isSTreak(s,k,3)
0	False
1	False
2	False

Using isStreak to Find Streaks

s = 'HHTHTTTHTHHHTHTT'

```
k     isSTreak(s,k,3)
----------------------------------------
0          False
1          False
2          False
3          False
```

Using isStreak to Find Streaks

```
s = 'HHTHTTTHTHHHTHTT'
```

```
k     isSTreak(s,k,3)
---------------------------------------
0          False
1          False
2          False
3          False
4          True
```

Using isStreak to Find Streaks

```
def FindStreak(s,n):
   k=0
   while k<len(s) and (not isStreak(s,k,n)):
         # s[k:k+n] is not a streak
         k = k+1
   if k<len(s):
         # isStreak(s,k,n) is True
         return k
   else:
         # k==len(s) is True
         return -1
```