This is CS50

Think.
Pair.
Share.

- Why are we using **C**?
- How can we **read** and **write** code that includes **variables**, **conditionals**, and **loops**?
- Why do we care about **data types**?
- What does it mean to **compile** a C program?

- Why are we using **C**?
- How can we **read** and **write** code that includes **variables**, **conditionals**, and **loops**?
- Why do we care about **data types**?
- What does it mean to **compile** a C program?
- How many years will it take to double our llamas?

Part 1

Variables and Types
Input and Printing

Variables

`calls`

Variables

```
int calls = 4;
```

Variables

```
int calls = 4;
```

name

calls

Variables

```
int calls = 4;
```

type

calls

Variables

```
int calls = 4;
```

value

calls

Variables

```
int calls = 4;
```

assignment operator

calls

Variables

```
int calls = 4;
```

type name | value

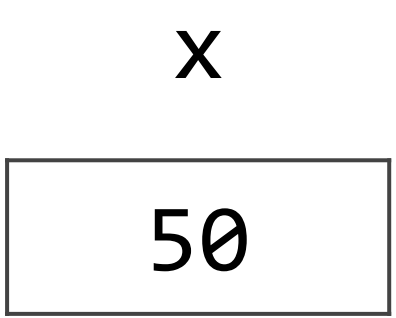
assignment operator

calls

"Create an **integer** variable named **calls** that **gets** the **value 4**."

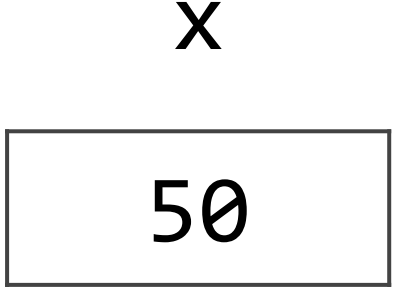
Variables

```
int x = 50;
```



Variables

```
int x = 50;
```



"Create an **integer** variable named **x** that **gets** the **value 50**."

Think.
Pair.
Share.

Why does C care about data types?

01000001

int

65

01000001

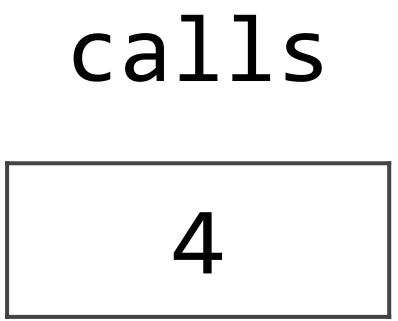
char

'A'

01000001

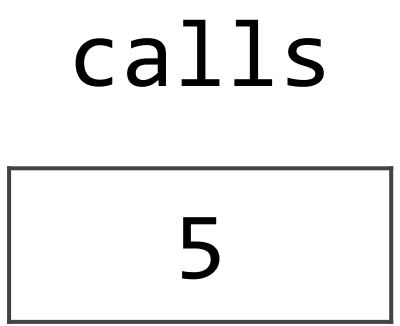
Variables

```
int calls = 4;
calls = 5;
```



Variables

```
int calls = 4;
calls = 5;
```



Variables

```
int calls = 4;
calls = 5;
```

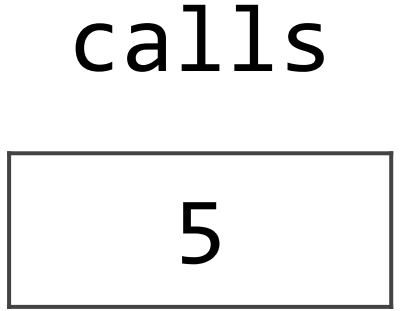
name | value

assignment operator

"**calls gets 5**."

Operators

```
int calls = 4;
calls = calls + 1;
```

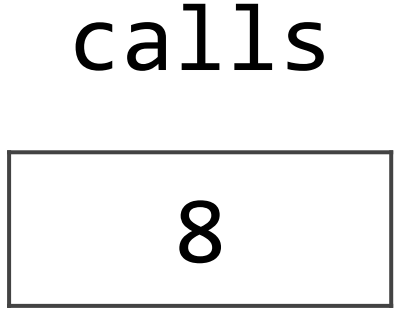


Operators

```
int calls = 4;
calls = calls - 1;
```

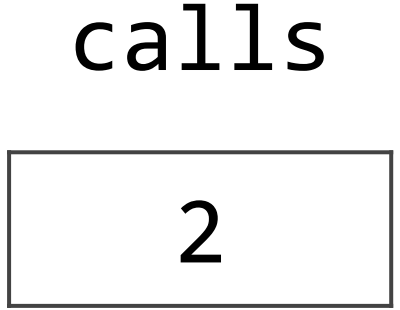
Operators

```
int calls = 4;
calls = calls * 2;
```



Operators

```
int calls = 4;
calls = calls / 2;
```



Getting input

```
int calls = get_int("Calls: ");
```

type

name

assignment operator

function call

Functions

```
int calls = get_int("Calls: ");
```

function call

Functions

```
int calls = get_int("Calls: ");
```

function name

Functions

```
int calls = get_int("Calls: ");
```

arguments

Functions

```
int calls = get_int("Calls: ");
```

function call

Return values

```
int calls = 4;
```

value

Storing return values

```
int calls = 4;
```

type name | value

assignment operator

calls

"Create an **integer** variable named **calls** that **gets** the **value 4**."

Printing values

```
int calls = 4;
printf("calls is %i\n", calls);
```

Printing values

```
int calls = 4;
printf("calls is %i\n", calls);
```

format code

Printing values

```
int calls = 4;
printf("calls is %i\n", calls);
```

Types and format codes

Numbers		Text
`int (%i)`	`long (%li)`	`char (%c)`
`float (%f)`	`double (%f)`	`string (%s)`

Exercise

Create a C program that prompts a user for:

- A name
- An age
- A phone number

Print the values back to the user as confirmation.

Part 2

Breaking down loops and conditionals

```
if (calls < 1)
{
    printf("Call more often!\n");
}
```

Boolean expression

```
if (calls < 1)
{
    printf("Call more often!\n");
}
```

conditional

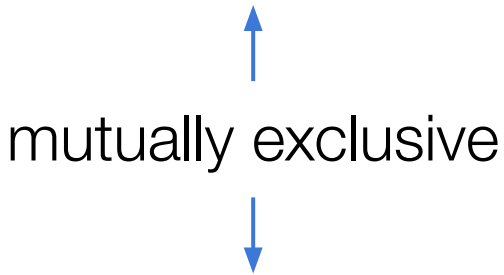
```
if (calls < 1)
{
    printf("Call more often!");
}
```

```
if (calls < 1)
{
    printf("Call more often!");
}
```

executed conditionally

```
if (calls < 1)
{
    printf("Call more often!\n");
}
else
{
    printf("Thanks for calling!\n");
}
```

```
if (calls < 1)
{
    printf("Call more often!\n");
}
else
{
    printf("Thanks for calling!\n");
}
```



```
int i = 0;
while (i < 10)
{
    printf("%i\n", i);
    i = i + 1;
}
```

initialization

```
int i = 0;
while (i < 10)
{
    printf("%i\n", i);
    i = i + 1;
}
```

Boolean expression

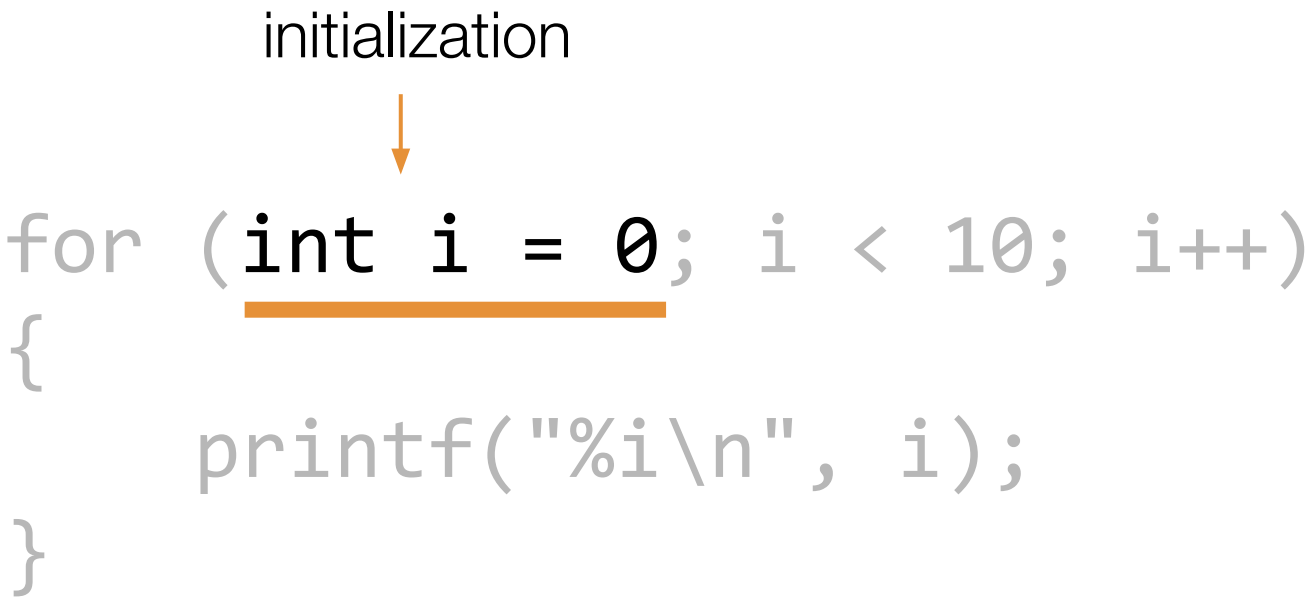
```
int i = 0;
while (i < 10)
{
    printf("%i\n", i);
    i = i + 1;
}
```

```
int i = 0;
while (i < 10)
{
    printf("%i\n", i);
    i = i + 1;
}
```

incrementation

```c
int i = 0;
while (i < 10)
{
    printf("%i\n", i);
    i = i + 1;
}
```

```
for (int i = 0; i < 10; i++)
{
    printf("%i\n", i);
}
```


initialization
for (int i = 0; i < 10; i++)
{
printf("%i\n", i);
}

Boolean expression

```
for (int i = 0; i < 10; i++)
{
    printf("%i\n", i);
}
```

incrementation

```
for (int i = 0; i < 10; i++)
{
    printf("%i\n", i);
}
```

```
for (int i = 0; i < 10; i++)
{
    printf("%i\n", i);
}
```

```
int n;
do
{
    n = get_int("n: ");
}
while (n <= 0);
```

```
int n;
do
{
    n = get_int("n: ");
}
while (n <= 0);
```

```c
int n;
do
{
    n = get_int("n: ");
}
while (n <= 0);
```

Part 3 Lab

https://cs50.harvard.edu/college/2022/fall/labs/1/

- Work an example yourself
- Write down exactly what you did
- Create an algorithm after working multiple examples
- Test your algorithm by hand
- Translate your algorithm to code
- Find errors in your code by running test cases
- Fix errors in your code

- Work an example yourself
- Write down exactly what you did
- Create an algorithm after working multiple examples
- Test your algorithm by hand
- Translate your algorithm to code
- Find errors in your code by running test cases
- Fix errors in your code

- Work an example yourself
- Write down exactly what you did
- Create an algorithm after working multiple examples
- Test your algorithm by hand
- Translate your algorithm to code
- Find errors in your code by running test cases
- Fix errors in your code

We have a population of **n** llamas.

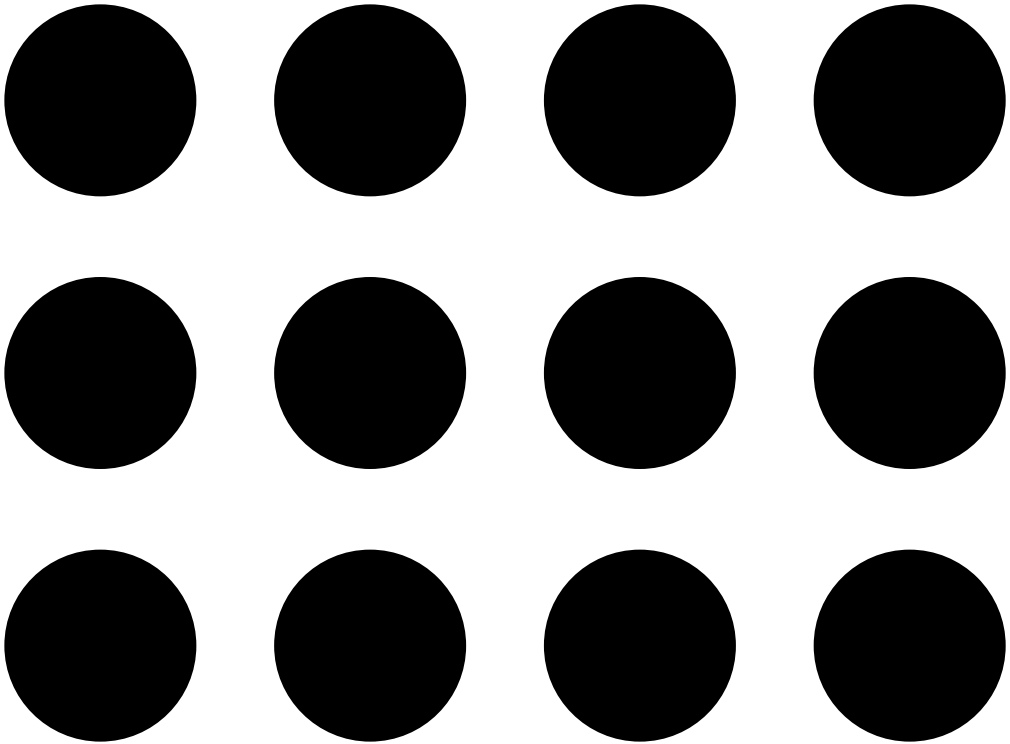
Each year, **n/3** new llamas are born, and **n/4** llamas pass away.

How many years will it take to have a certain population of llamas?

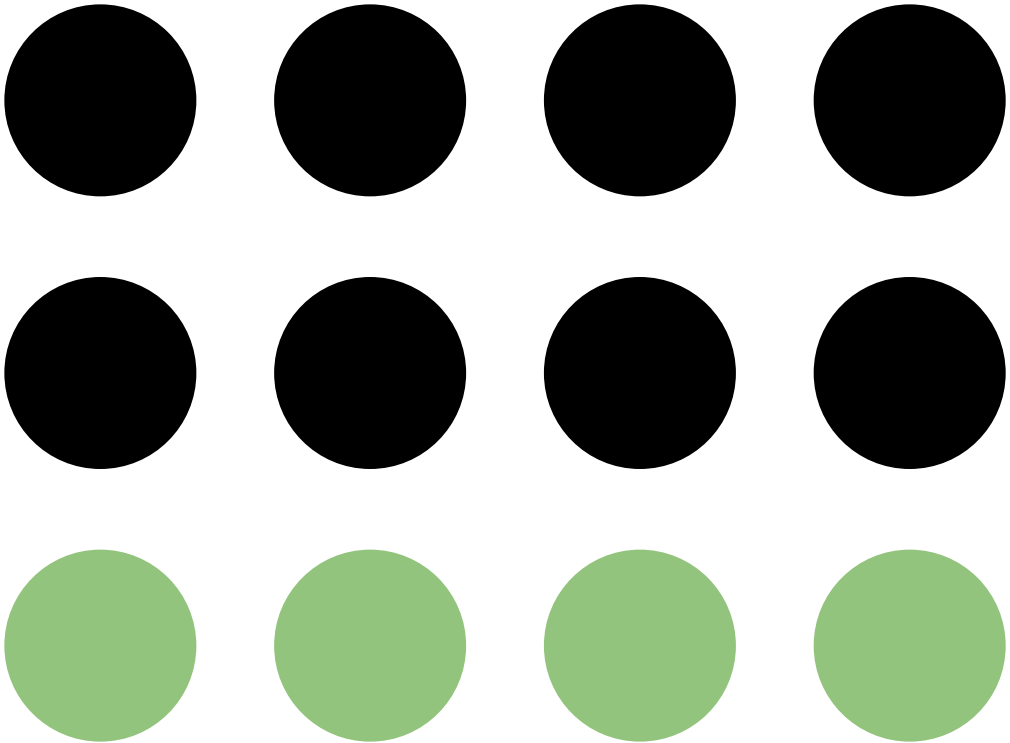
We have a population of **12** llamas.

Each year, **12/3** new llamas are born, and **12/4** llamas pass away.

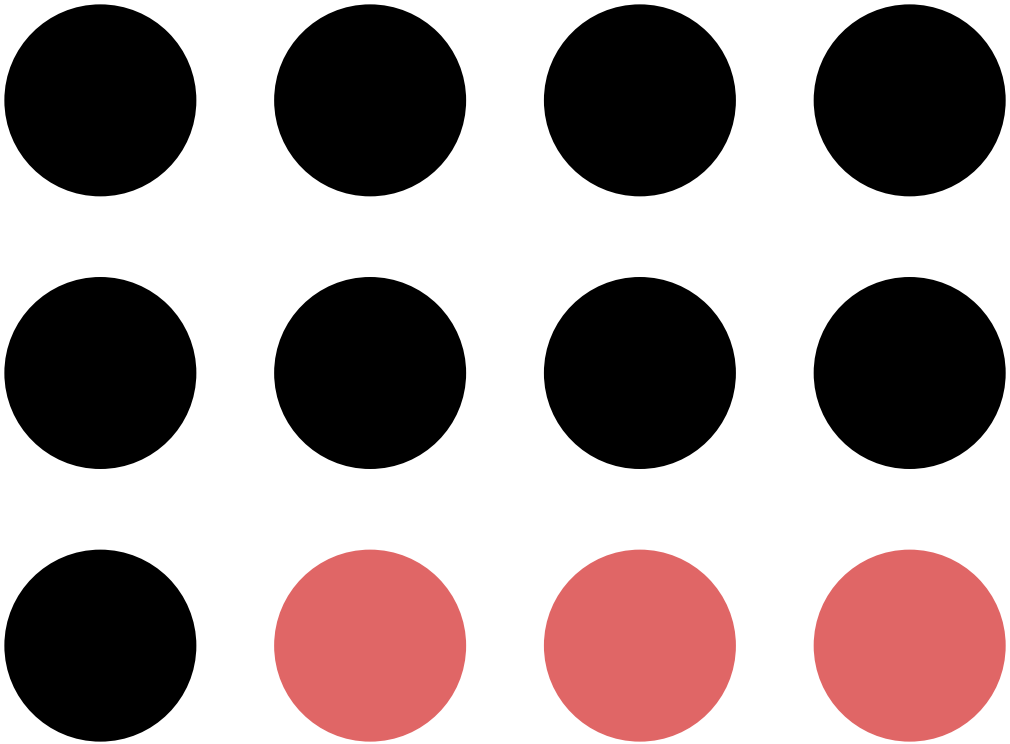
How many years will it take to have a population of **13** llamas?



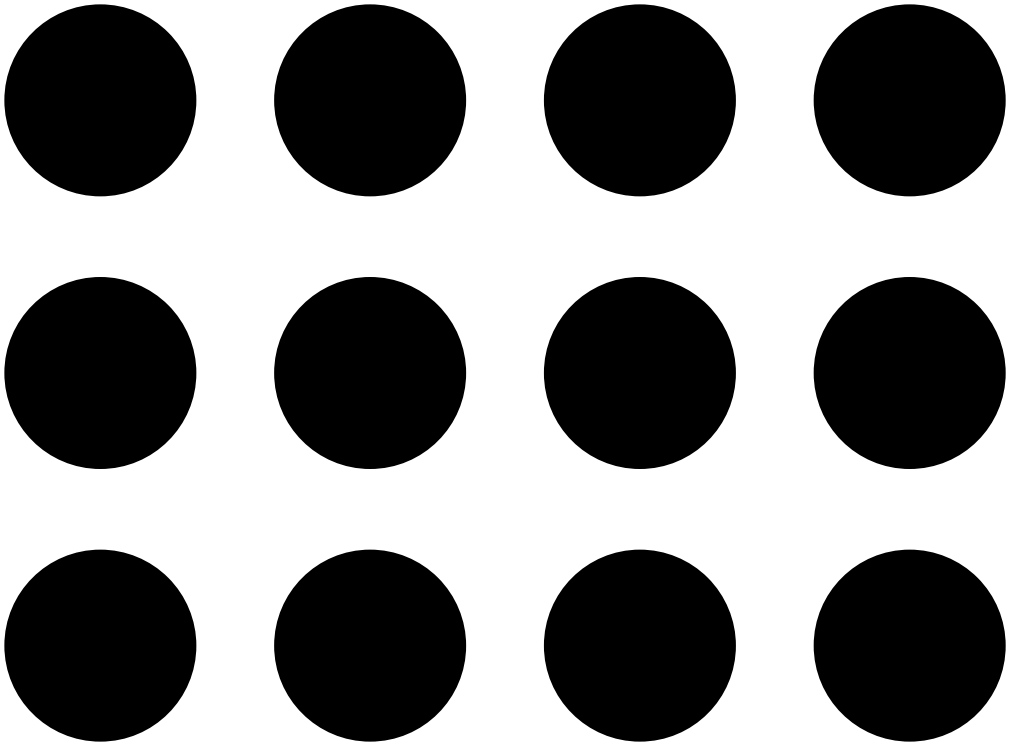
Year 0



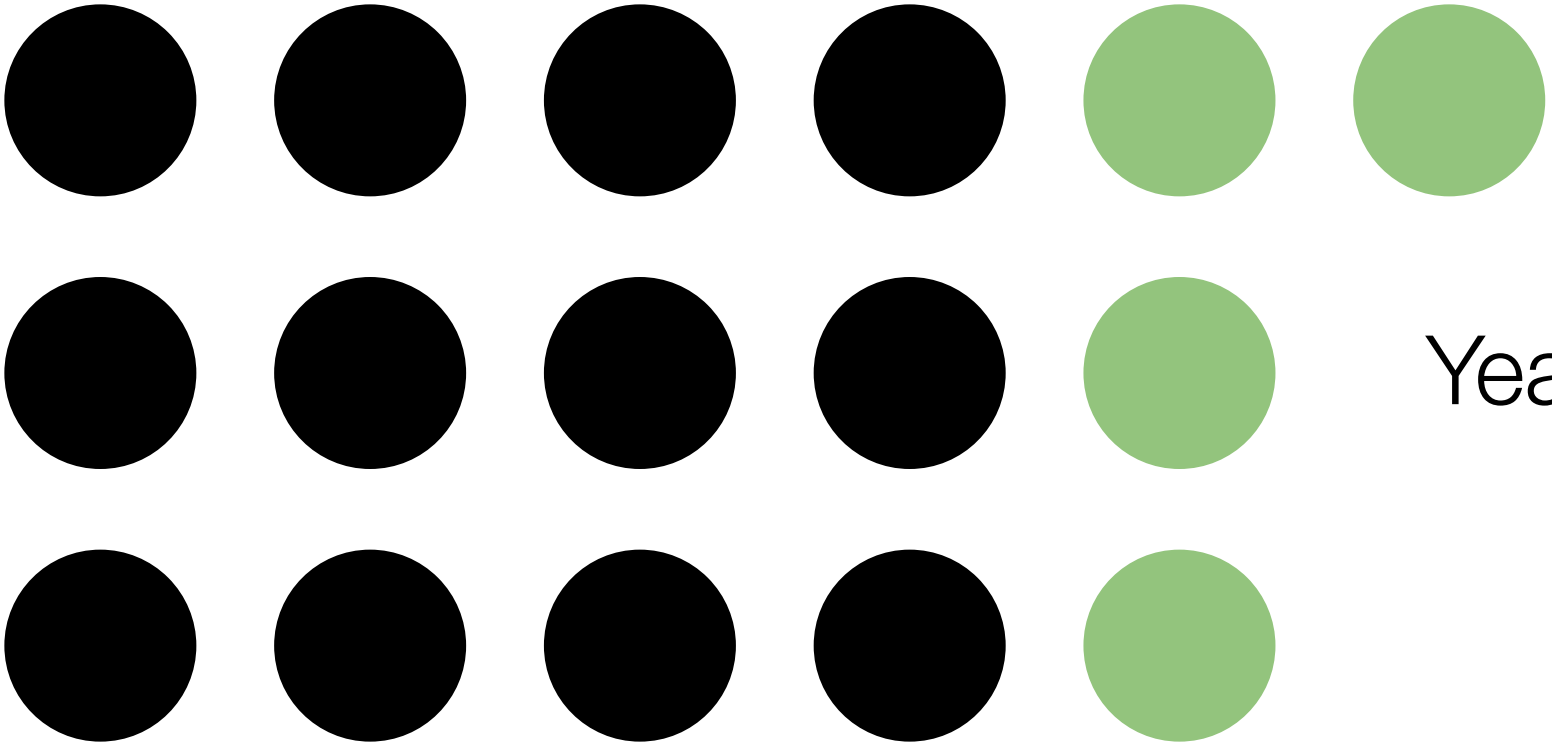
Year 0



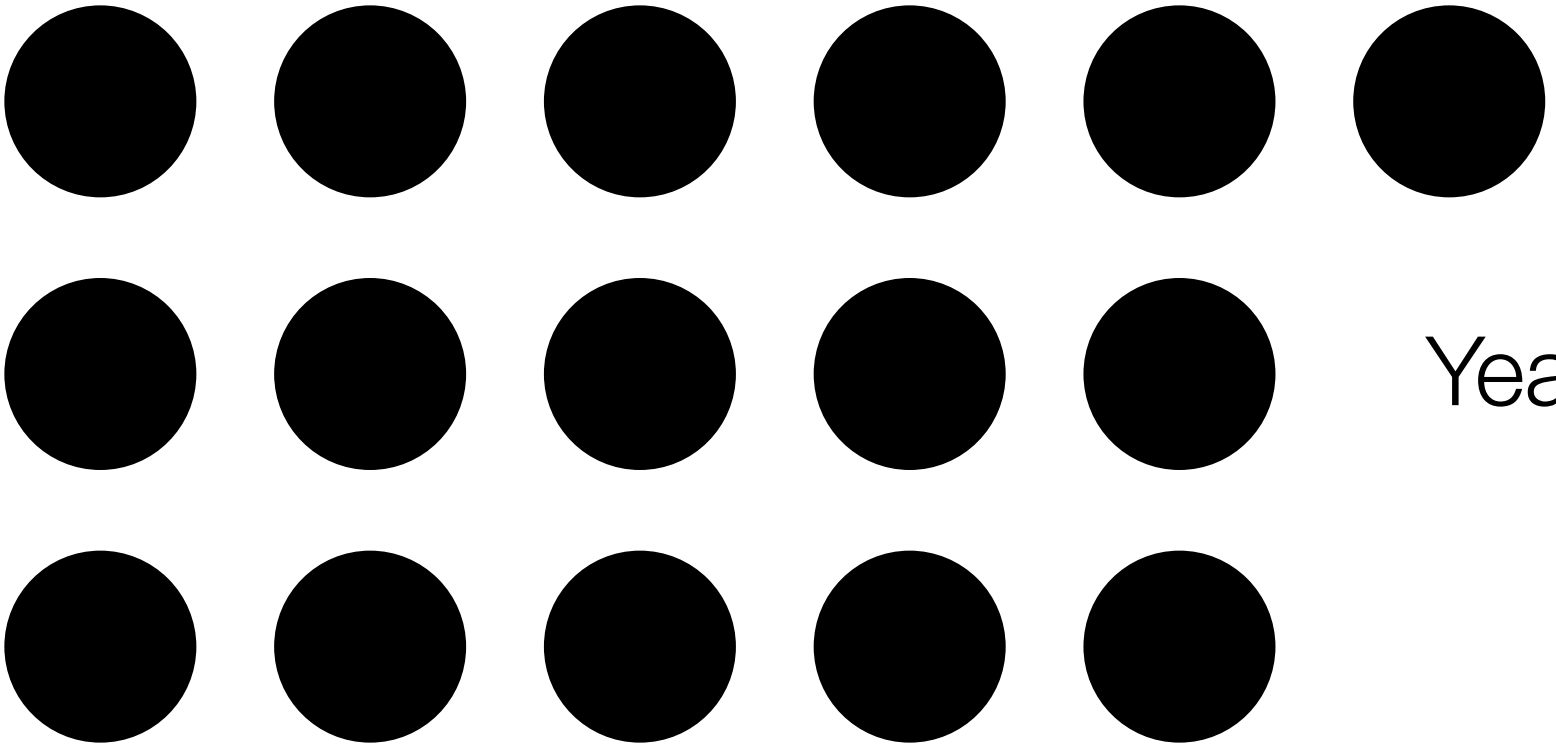
Year 0



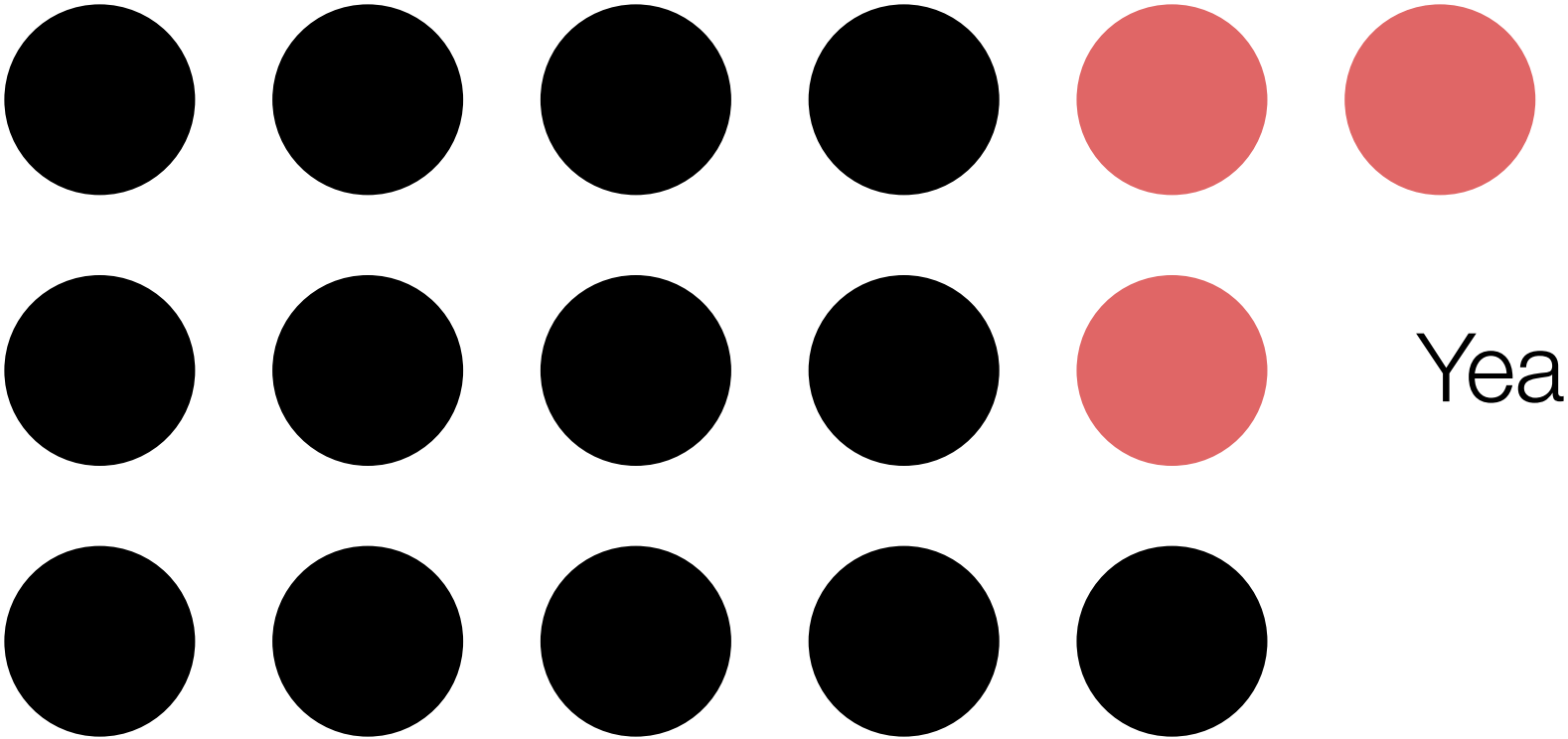
Year 0

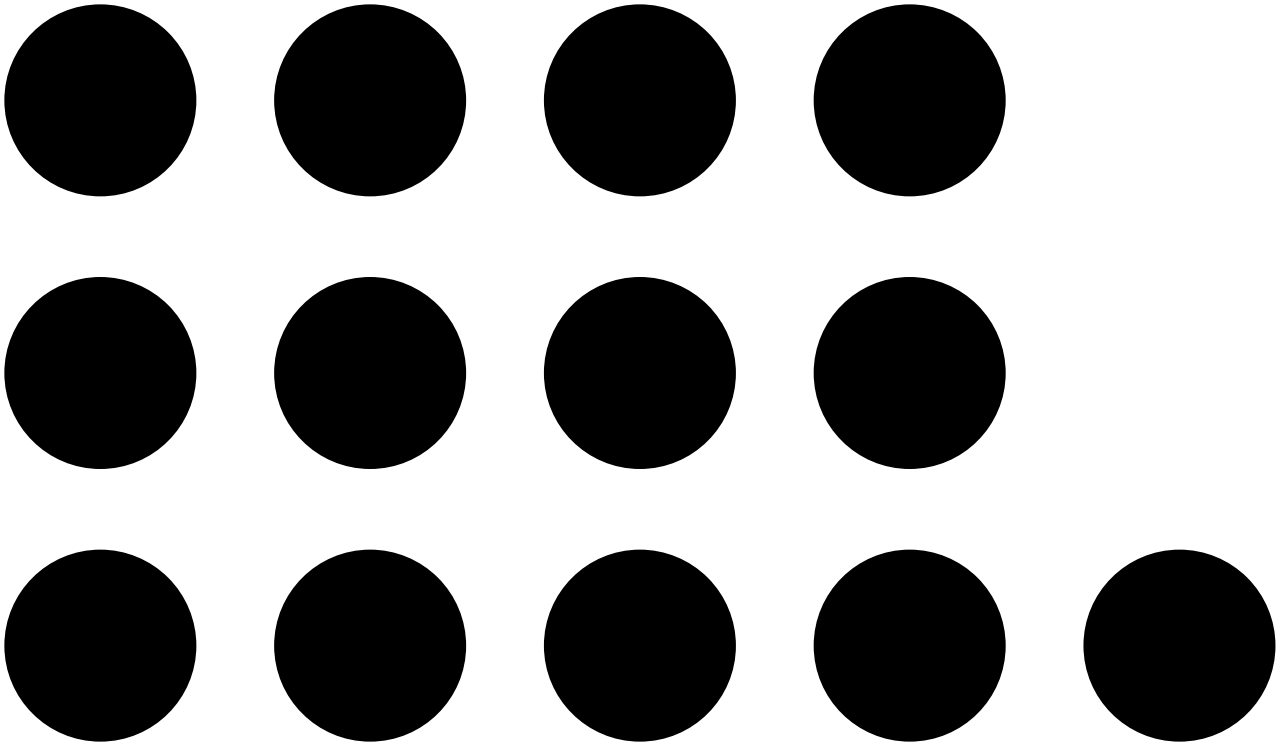


Year 1



Year 1


Year 1



Year 1

- Prompt the user for a **starting number** of llamas
- Prompt the user for a **goal number** of llamas
- **Add and subtract llamas every "year"** until we reach the goal number of llamas
- **Print the number of years** it took to reach the goal number of llamas

This was CS50