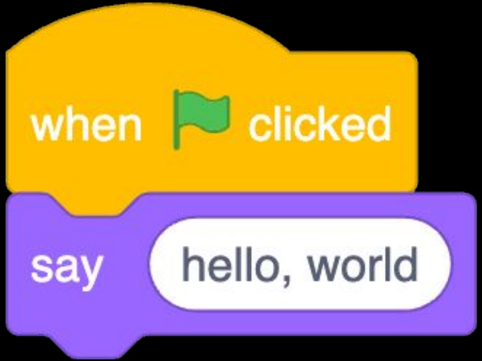
# This is CS50

- functions
    - arguments, side effects, return values
- conditionals
- Boolean expressions
- loops
- variables
- ...



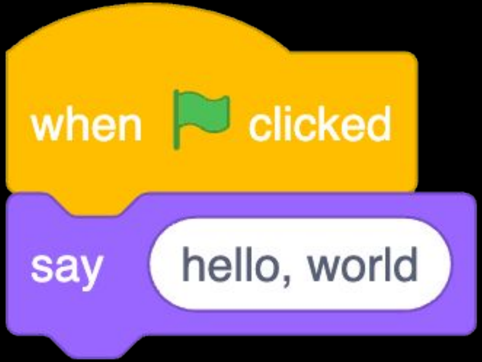
```
when 🏳 clicked
say (hello, world)
```

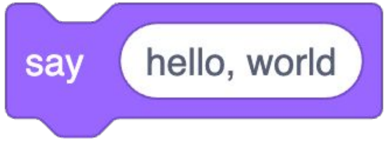
```c
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

## syntax highlighting

Import library

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

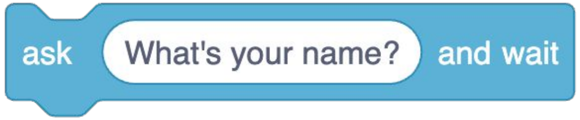

when clicked
say
hello, world

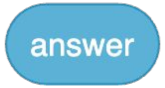


```
printf("hello, world\n");
```

newline

```
get_char

get_double

get_float

get_int

get_long

get_string

...
```



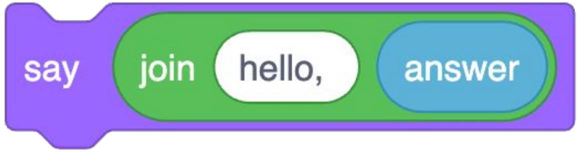


return value

```
string answer = get_string("What's your name? ");
```

need to declare

return value declared above



plugs into placeholder

```
printf("hello, %s\n", answer);
```

declared

placeholder!

%% = "%"

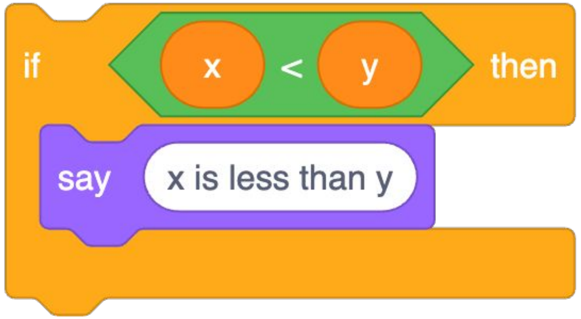
if you do %s %s , ___ , ___

*data*

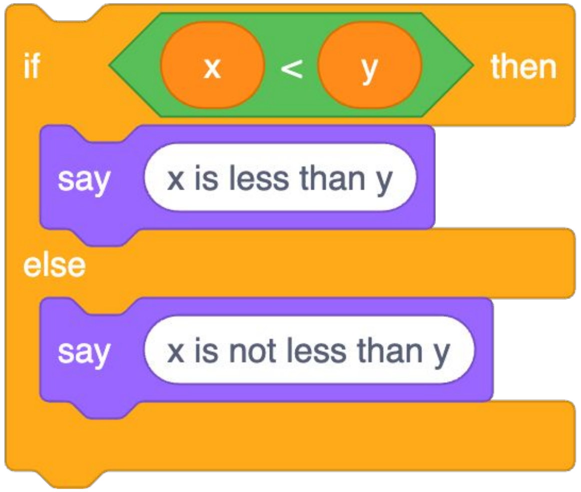
## types

```
bool

char

double

float

int

long

string

...
```

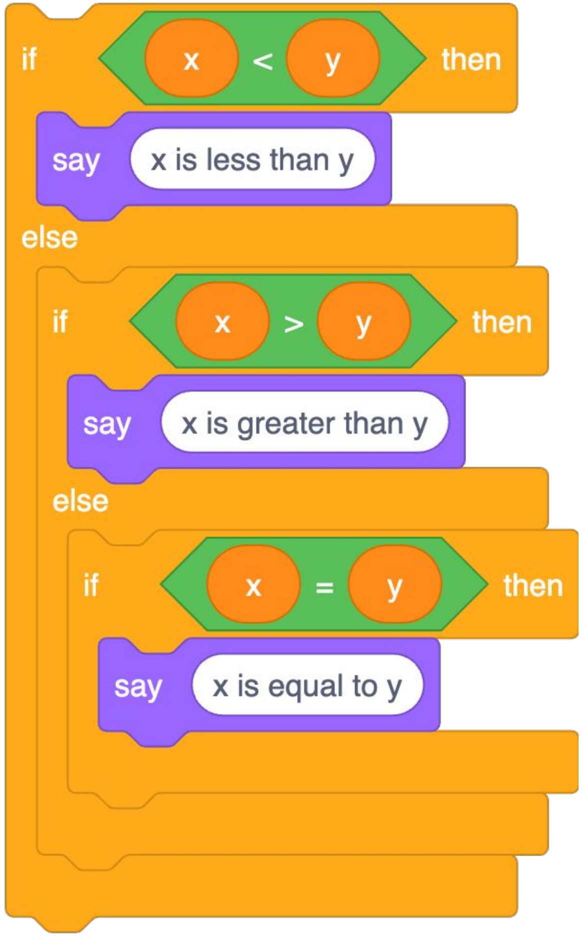
## conditionals



```
if (x < y)
{
    printf("x is less than y\n");
}
```



```
if (x < y)
{
    printf("x is less than y\n");
}
else
{
    printf("x is not less than y\n");
}
```

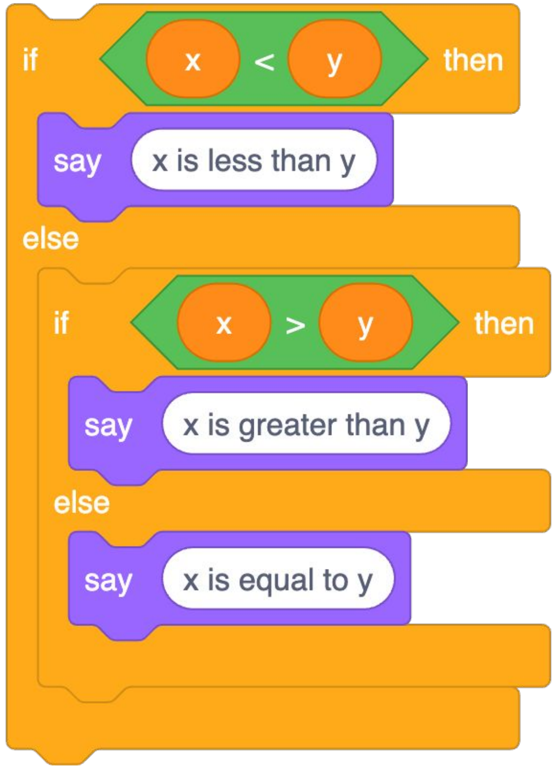


```
if x < y then
    say x is less than y
else
    if x > y then
        say x is greater than y
    else
        if x = y then
            say x is equal to y
```

```
if (x < y)
{
    printf("x is less than y\n");
}
else if (x > y)
{
    printf("x is greater than y\n");
}
else if (x == y)
{
    printf("x is equal to y\n");
}
```

redundant

like %% = "%"



```
if (x < y)
{
    printf("x is less than y\n");
}
else if (x > y)
{
    printf("x is greater than y\n");
}
else
{
    printf("x is equal to y\n");
}
```

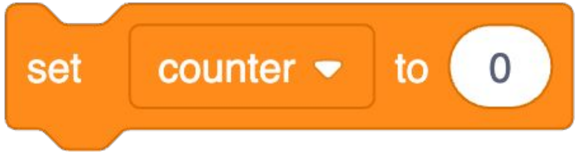
much more well designed

char ' '
string " "

|| or

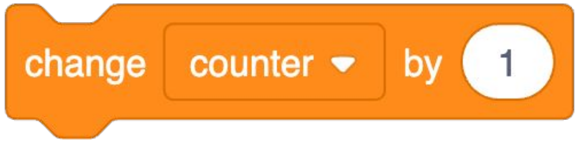
```
char c = get char ("Do you agree"
if ( c = 'y' || c = 'Y' )
{
   printf (" Agreed. "\n);
}

if ( c = 'n' || c = 'N' )
{
   printf (" Disagreed." \n );
}
```

## variables



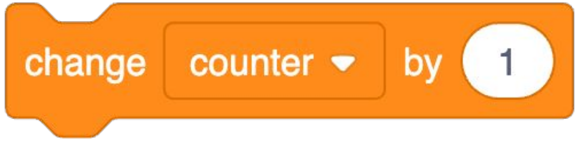
need to declare

```
int counter = 0;
```

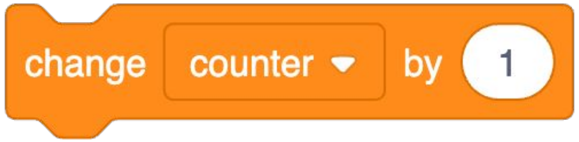


right to left

```
counter = counter + 1
```



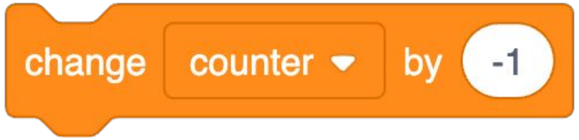
```
counter += 1;
```



change counter by 1

```
counter++;
```

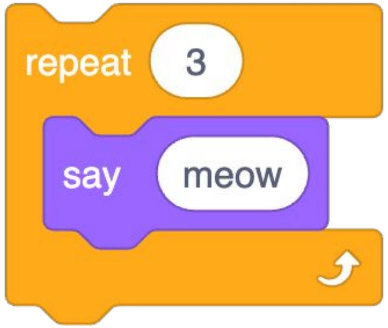
add 1



```
counter--;
```

subtract 1

## loops



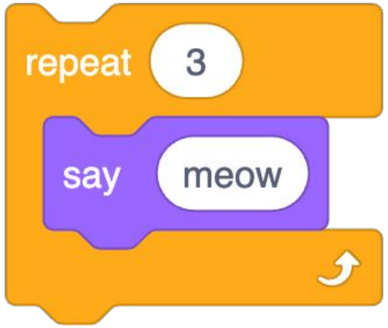
declare ↘ as an integer

counter is 3

```
int counter = 3;
while (counter > 0)
{
    printf("meow\n");
    counter = counter - 1;
}
```

and also

while loop



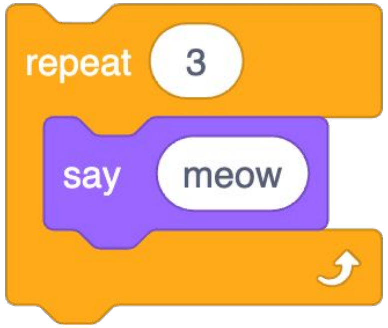
```
int i = 3;
while (i > 0)
{
    printf("meow\n");
    i--;
}
```

or i-=1;
or i = i-1;

counting down from 3

3
2
1
0

while loop



note: starting from 1

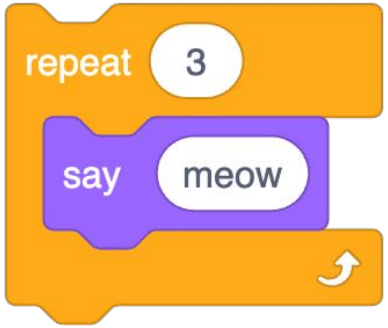
less than or equal 2

```
int i = 1;
while (i <= 3)
{
    printf("meow\n");
    i++;
}
```

counting up

1
2
3

while loop

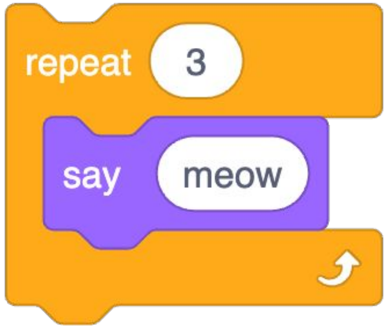


best practice to start w/ 0

```
int i = 0;
while (i < 3)
{
    printf("meow\n");
    i++;
}
```

count up ↑

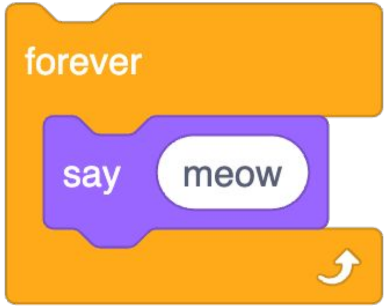
for loop



just happens once

these loop

```
for (int i = 0; i < 3; i++)
{
    printf("meow\n");
}
```



or (

```
while (true)
{

}
```

## Linux

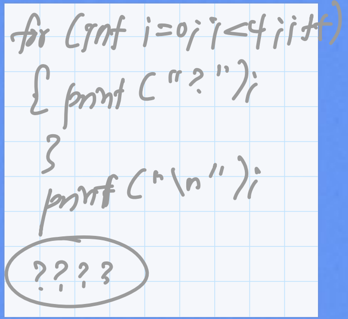
graphical user interface

## GUI

## command-line interface

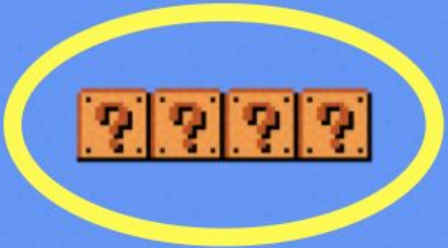
## CLI

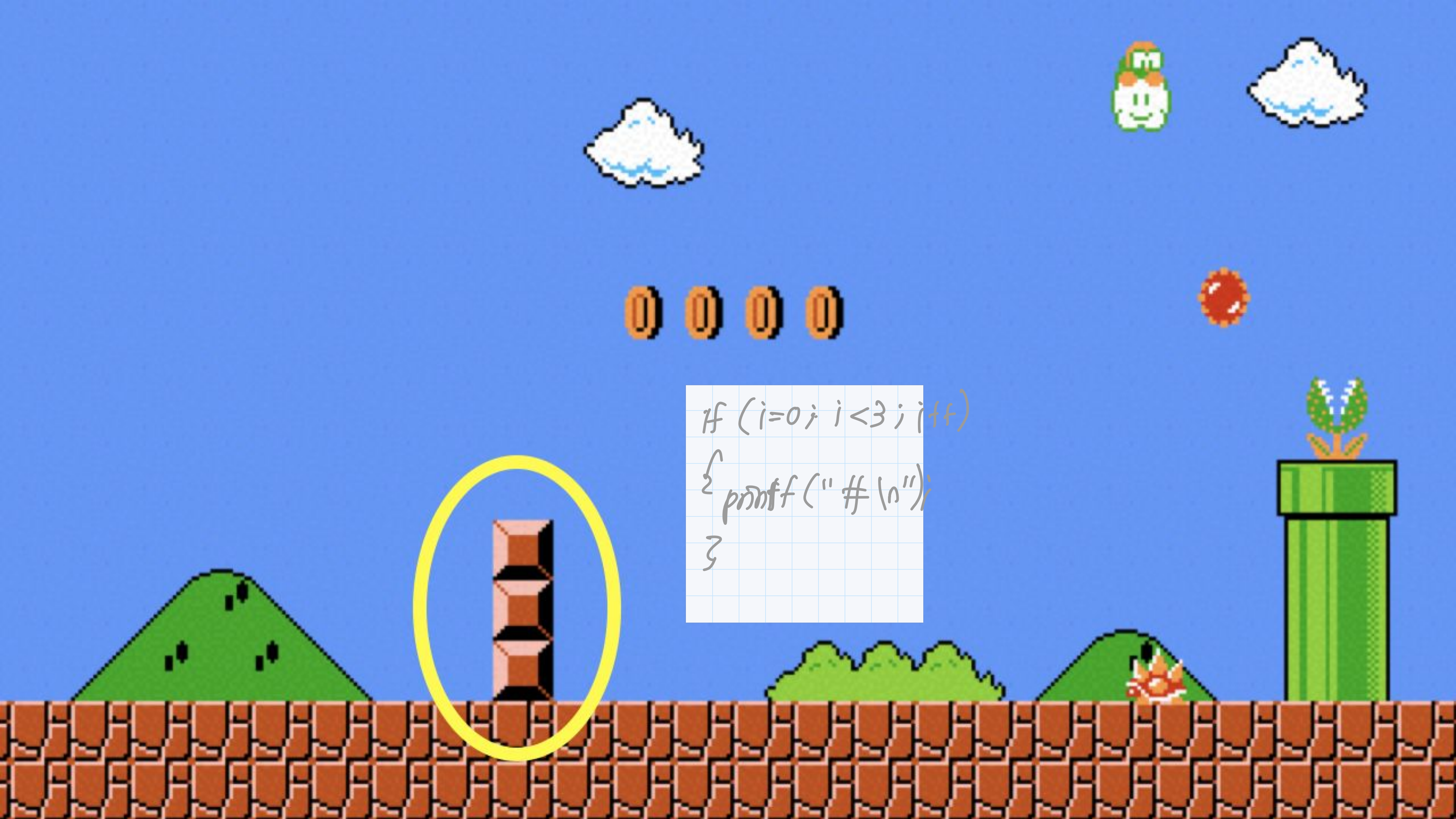
```
cd

cp

ls     list

mkdir

mv

rm

rmdir

...
```



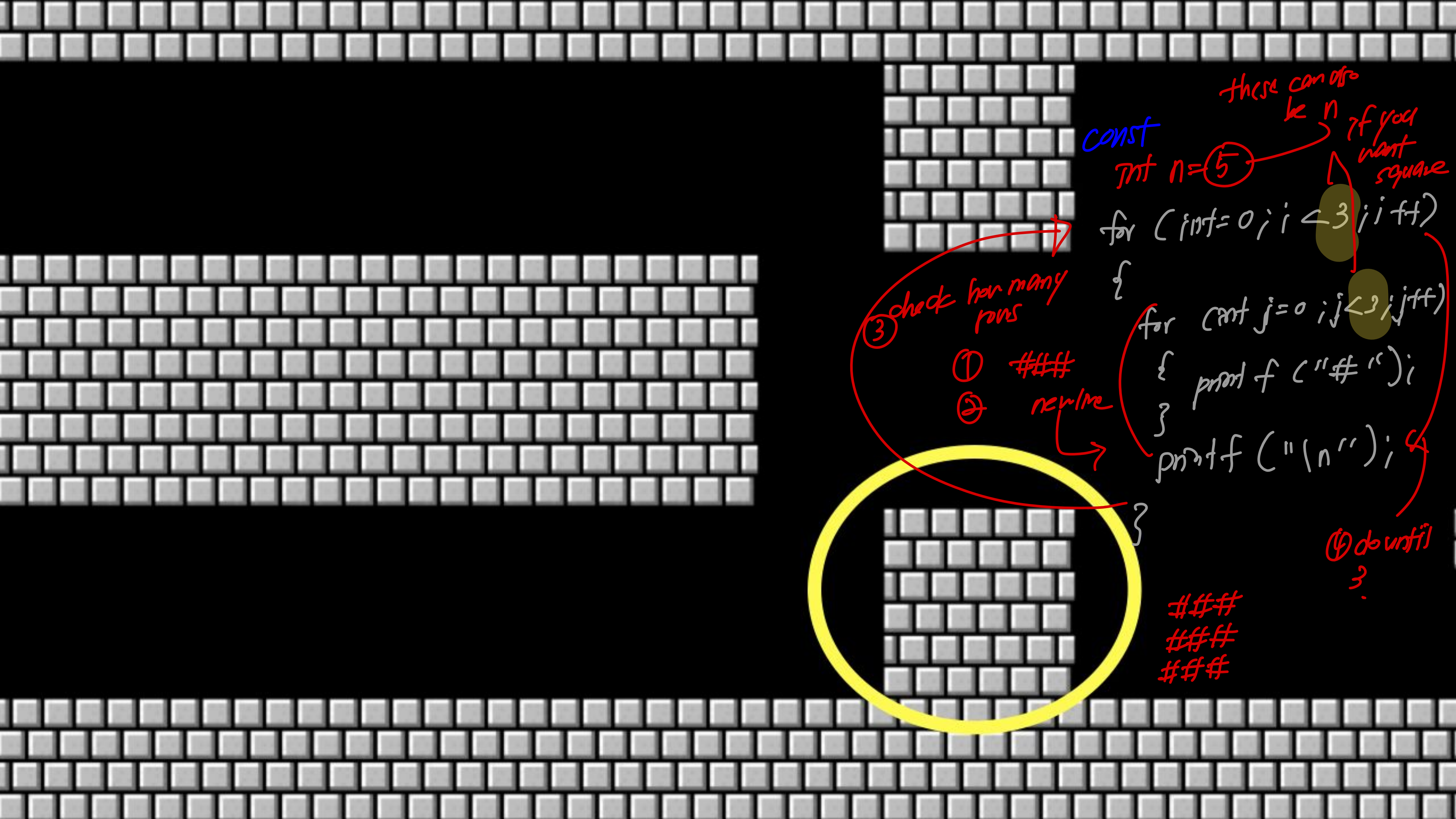
```
for (int i=0; i<4; i++)
 { print ("?");
  }
  printf("\n");
```

????





```
if (i=0; i<3; i++)
{
 printf("#\n");
}
```



these can also be n if you want square

const int n = $\boxed{5}$

```
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        printf("#");
    }
    printf("\n");
}
```

③ check how many rows

① ###

② newline

④ do until 3

```
###
###
###
```

do while loop

```
do {
    n = get_int("size: ");

}
while (n < 1);
```

③ loop

① ask n value

② check for n and while it is < 1, then

## OPERATORS

+

-

*

/

%

## format codes

☆ important

%c → char

%f → floating point value (decimal)

%i → integer

%li → long integer

%s → string

truncation

```
long x = get_long("x: ");
long y = get_long("y: ");

float z = (float) x / (float) y;
printf("%f\n", z);
```

declar

still plugging in.

# type casting

converting

## floating-point imprecision

double = 64 bits
vs
32 bits