The MzTEK Guide to Programming

photo credit flickr:Grietje & Barien

REVIEW LAST WEEK

To create a new list (to *declare* it):

```
int[] x = new int[4];
```

To create a new list (to *declare* it):

what kind of things are in the list? i.e. how much space needs to be reserved for each item

```
int[] x = new int[4];
```

To create a new list (to *declare* it):

what kind of things are in the list? i.e. how much space needs to be reserved for each item

```
int[] x = new int[4];
```

[ ] means the data type is an array

To create a new list (to *declare* it):

what kind of things are in the list? i.e. how much space needs to be reserved for each item

```
int[] x = new int[4];
```

[ ] means the data type is an array

what is the name of the list?

To create a new list (to *declare* it):

what kind of things are in the list? i.e. how much space needs to be reserved for each item

how long will the list be?

```
int[] x = new int[4];
```

[ ] means the data type is an array

what is the name of the list?

```
if ( )
{

}
else {

}
```

put in a comparison statement (like < or >)

```
if ( )
{

}
else {

}
```

put in a comparison statement (like < or >)

```
if ( )
{

}
else {

}
```

what to do if our comparison statement is true

put in a comparison statement (like < or >)

```
if ( )
{

}
else {

}
```

what to do if our comparison statement is true

what to do if our comparison statement is false

put in a comparison statement (like < or >)

```
if ( )
{


}
else {


}
```

what to do if our comparison statement is true

we don't have to always have an else statement, sometimes you only care if the statement is true

what to do if our comparison statement is false

What if we want to use multiple `if` statements?

```
int counter;
// some other code...
if (counter < 10) {
  if (counter > 0 ) {
    counter++;
  }
}
```

What if we want to use multiple `if` statements?

If `counter` is less than 10 and greater than 0, then increase counter by 1.

```
int counter;
// some other code...
if (counter < 10) {
  if (counter > 0 ) {
    counter++;
  }
}
```

What if we want to use multiple `if` statements?

```
int counter;
// some other code...
if (counter < 10) {
  if (counter > 0 ) {
    counter++;
  }
}
```

If `counter` is less than 10 and greater than 0, then increase counter by 1.

`counter` is only increased if both `if` statements are true.

What if we want to use multiple `if` statements?

If `counter` is less than 10 and greater than 0, then increase counter by 1.

```
int counter;
// some other code...
if (counter < 10) {
  if (counter > 0 ) {
    counter++;
  }
}
```

`counter` is only increased if both `if` statements are true.

These are called nested `if` statements, because one is inside the { } of the other.

What if we want to use multiple `if` statements?

```
int counter;
// some other code...
if (counter > 10) {
  counter = 0;
}
if (counter < 0 ) {
  counter = 0;
  }
}
```

What if we want to use multiple `if` statements?

If `counter` is greater than 10 or less than 0, then reset counter to 0.

```
int counter;
// some other code...
if (counter > 10) {
  counter = 0;
}
if (counter < 0 ) {
  counter = 0;
  }
}
```

What if we want to use multiple `if` statements?

If `counter` is greater than 10 or less than 0, then reset counter to 0.

```
int co
// some other code...
if (counter > 10) {
  counter = 0;
}
if (counter < 0 ) {
  counter = 0;
  }
}
```

`counter` is reset if either `if` statements are true.

BOOLEAN OPERATOR OR

True	OR	False	is
False	OR	True	is
True	OR	True	is
False	OR	False	is

BOOLEAN OPERATOR OR

True	OR	False	is	True
False	OR	True	is	
True	OR	True	is	
False	OR	False	is	

BOOLEAN OPERATOR OR

True	OR	False	is	True
False	OR	True	is	True
True	OR	True	is	
False	OR	False	is	

BOOLEAN OPERATOR OR

True	OR	False	is	True
False	OR	True	is	True
True	OR	True	is	True
False	OR	False	is	

BOOLEAN OPERATOR OR

True	OR	False	is	True
False	OR	True	is	True
True	OR	True	is	True
False	OR	False	is	False

BOOLEAN OPERATOR OR

True	OR	False	is	True
False	OR	True	is	Tr
True	OR	True	is	Tr
False	OR	False	is	False

When using OR in code, type ||

```
int counter;
// some other code...

if (counter > 10) {
 counter = 0;
}
if (counter < 0 ) {
 counter = 0;
  }
}
```

```
int counter;
// some other code...

if (counter > 10) {
  counter = 0;
}
if (counter < 0 ) {
  counter = 0;
  }
}
```

```
int counter;
// some other code...

if ((counter < 0) ||
counter > 10)) {
  counter = 0;
}
```

```
int counter;
// some other code...

if (counter > 10) {
  counter = 0;
}
if (counter < 0 ) {
  counter = 0;
  }
}
```

OR

```
int counter;
// some other code...

if ((counter < 0) ||
counter > 10)) {
  counter = 0;
}
```

BOOLEAN OPERATOR AND

True	AND	False	is
False	AND	True	is
True	AND	True	is
False	AND	False	is

BOOLEAN OPERATOR AND

True	AND	False	is	False
False	AND	True	is	
True	AND	True	is	
False	AND	False	is	

BOOLEAN OPERATOR AND

True	AND	False	is	False
False	AND	True	is	False
True	AND	True	is	
False	AND	False	is	

BOOLEAN OPERATOR AND

True	AND	False	is	False
False	AND	True	is	False
True	AND	True	is	True
False	AND	False	is	

BOOLEAN OPERATOR AND

True	AND	False	is	False
False	AND	True	is	False
True	AND	True	is	True
False	AND	False	is	False

BOOLEAN OPERATOR AND

True	AND	False	is	False
False	AND	True	is	Fa
True	AND	True	is	Tr
False	AND	False	is	False

When using AND in code, type &&

```
int counter;
// some other code...

if (counter < 10) {
  if (counter > 0 ) {
    counter++;
  }
}
```

```
int counter;
// some other code...

if (counter < 10) {
  if (counter > 0 ) {
    counter++;
  }
}
```

```
int counter;
// some other code...

if ((counter > 0) &&
(counter < 10)) {
  counter++;
}
```

```
int counter;
// some other code...

if (counter < 10) {
  if (counter > 0 ) {
    counter++;
  }
}
```

AND

```
int counter;
// some other code...

if ((counter > 0) $\boxed{\&\&}$
(counter < 10)) {
  counter++;
}
```

BOOLEAN OPERATOR NOT

NOT True is

NOT False is

BOOLEAN OPERATOR NOT

NOT	True	is	False
NOT	False	is	

BOOLEAN OPERATOR NOT

NOT True is False

NOT False is True

BOOLEAN OPERATOR NOT

NOT	True	is	False
NOT	False	is	True

When using NOT in code, type !

```
int stopLoop = 0;
// some other code...

if(!stopLoop) {
 // some more code...
}
```

NOT

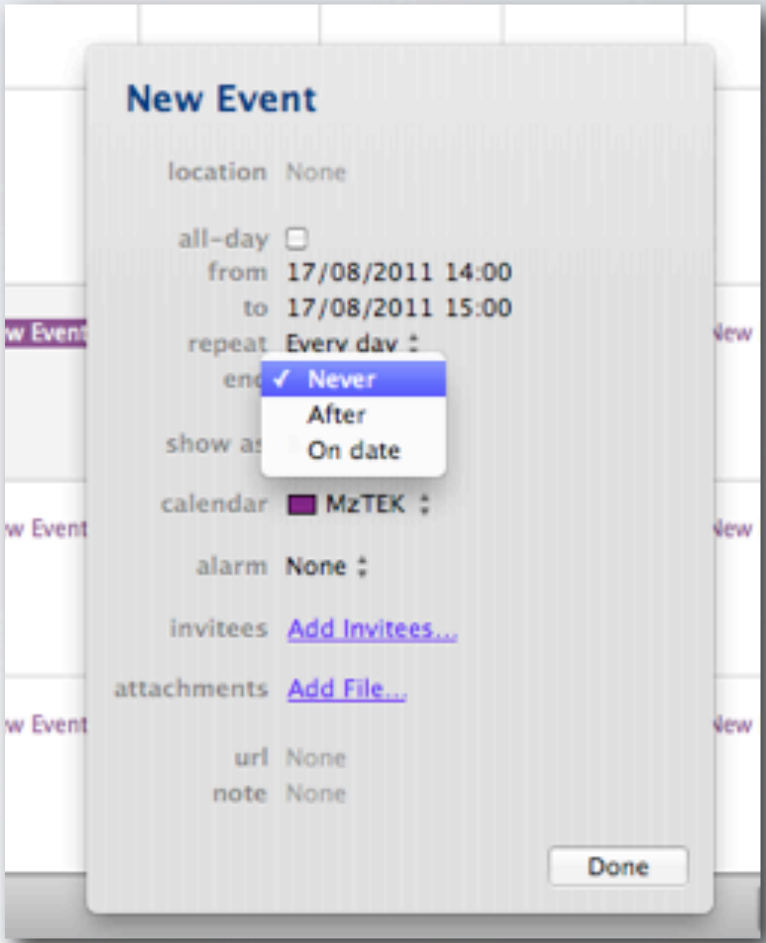
```
int stopLoop = 0;
// some other code...

if(!stopLoop) {
  // some more code...
}
```

LOOPS

There are two ways to repeat something:

1. Do this N number of times.

2. Keep doing this until something else happens.



New Event

location None

all-day ☐

from 17/08/2011 14:00

to 17/08/2011 15:00

repeat Every day

end [✓ Never] After On date

show as

calendar 🟪 MzTEK

alarm None

invitees Add Invitees...

attachments Add File...

url None

note None

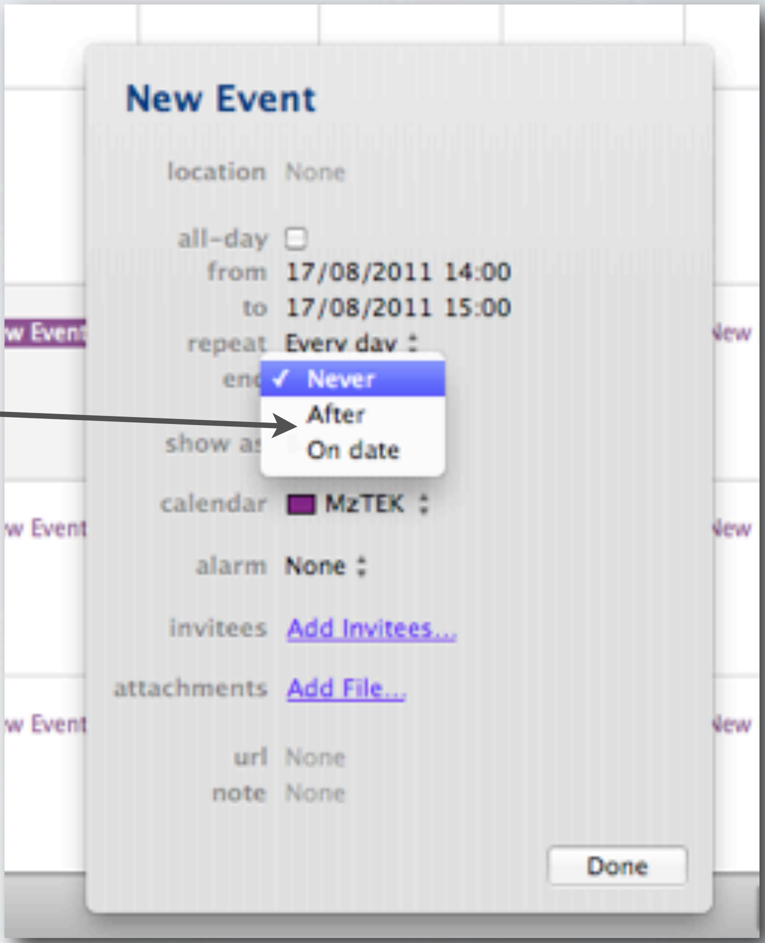
LOOPS

There are two ways to repeat something:

1. Do this times.

Repeat this event in the calendar this many times.

2. Keep doing this until something else happens.



New Event

location: None

all-day: ☐

from: 17/08/2011 14:00

to: 17/08/2011 15:00

repeat: Every day

end: [✓ Never] After On date

calendar: MzTEK

alarm: None

invitees: Add Invitees...

attachments: Add File...

url: None

note: None

LOOPS

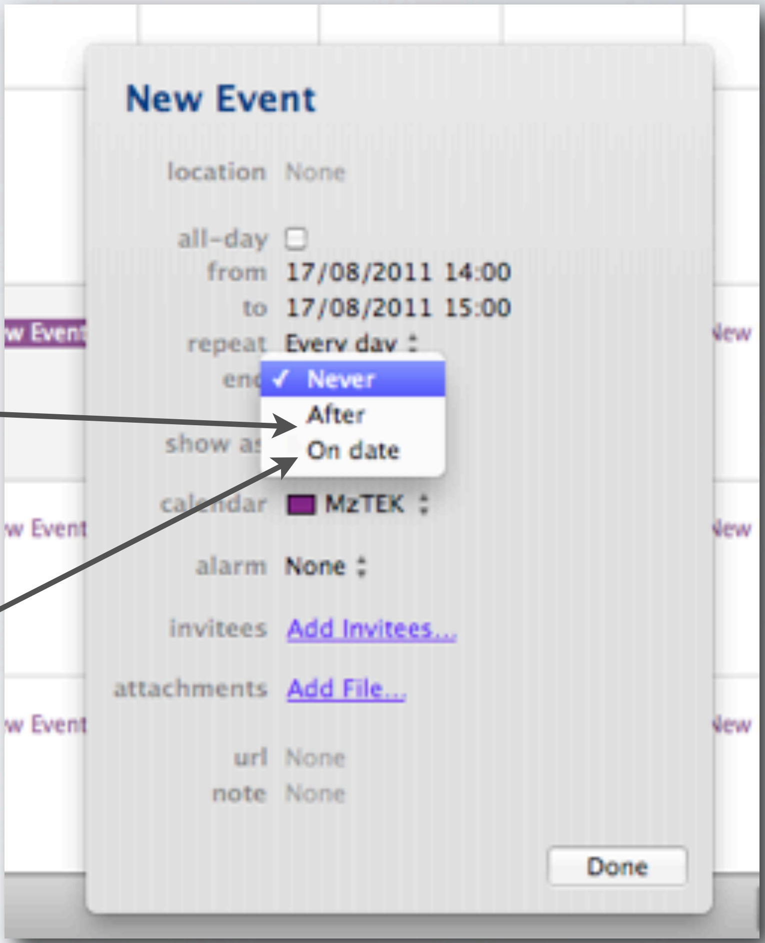
There are two ways to repeat something:

1. Do this N
   times.

2. Keep d
   someth .

Repeat this event in the calendar this many times.

Repeat this event in the calendar until a certain date occurs.



New Event

location None

all-day ☐

from 17/08/2011 14:00

to 17/08/2011 15:00

repeat Every day

end ✓ Never / After / On date

show as

calendar MzTEK

alarm None

invitees Add Invitees...

attachments Add File...

url None

note None

DO THIS N TIMES

```
int i;
for(i=0; i<4; i++) {

}
```

DO THIS N TIMES

start with a number, in this case 0

```
int i;
for(i=0; i<4; i++) {

}
```

DO THIS N TIMES

start with a number, in this case 0

if this statement is true

```
int i;
for(i=0; i<4; i++) {

}
```

DO THIS N TIMES

start with a number, in this case 0

if this statement is true

```
int i;
for(i=0; i<4; i++) {

}
```

then do whatever is written here

DO THIS N TIMES

start with a number, in this case 0

if this statement is true

```
int i;
for(i=0; i<4; i++) {

}
```

then do whatever is written here

when you've done what's in the { } once, do this, in this case add make i equal to its current value plus 1

DO THIS N TIMES

start with a number, in this case 0

if this statement is true

go back to see if the middle statement is still true

```
int i;
for(i=0; i<4; i++) {

}
```

then do whatever is written here

when you’ve done what’s in the { } once, do this, in this case add make i equal to its current value plus 1

KEEP DOING THIS UNTIL SOMETHING ELSE HAPPENS

```
while ( ) {

}
```

KEEP DOING THIS UNTIL SOMETHING ELSE HAPPENS

if the statement here is true

```
while ( ) {

}
```

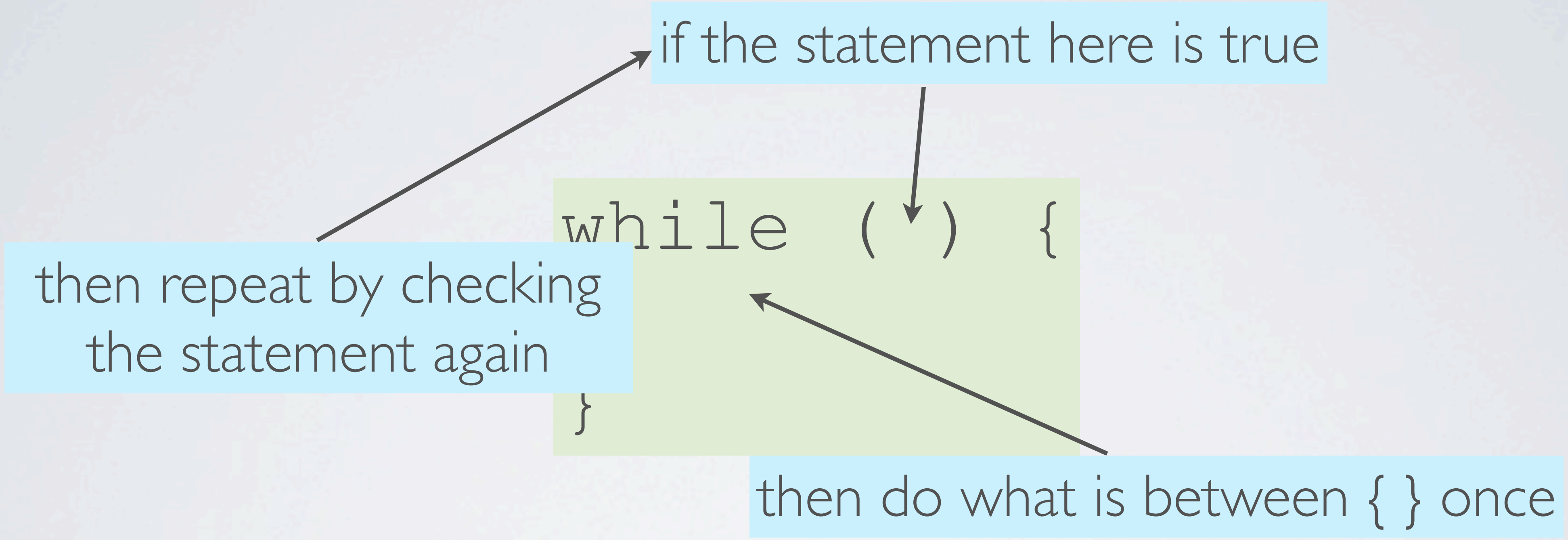
KEEP DOING THIS UNTIL SOMETHING ELSE HAPPENS

if the statement here is true

```
while ( ) {

}
```

then do what is between { } once

KEEP DOING THIS UNTIL SOMETHING ELSE HAPPENS



```
while ( ) {

}
```

if the statement here is true

then do what is between { } once

then repeat by checking the statement again

EXERCISE

Write out each iteration of these loops and what the variables equal at the end of each loop.

```
int i;
int j = 15;

for (i=0;  i<5; i++) {
  j = j * 2 - i;
}
```

```
int k = 100;

while ( k > 0 ) {
  k = k -10;
}
```

EXERCISE

Go through code at http://processing.org/learning/basics/embeddediteration.html

- identify all of the variables, why were those data types chosen?
- identify all of the comparisons made
- identify all control structures
- draw a diagram explaining what is happening in the code

```
float box_size = 11;
float box_space = 12;
int margin = 7;

size(200, 200);
background(0);
noStroke();

// Draw gray boxes

for (int i = margin; i < height-margin; i += box_space){
  if(box_size > 0){
    for(int j = margin; j < width-margin; j+= box_space){
      fill(255-box_size*10);
      rect(j, i, box_size, box_size);
    }
    box_size = box_size - 0.6;
  }
}
```

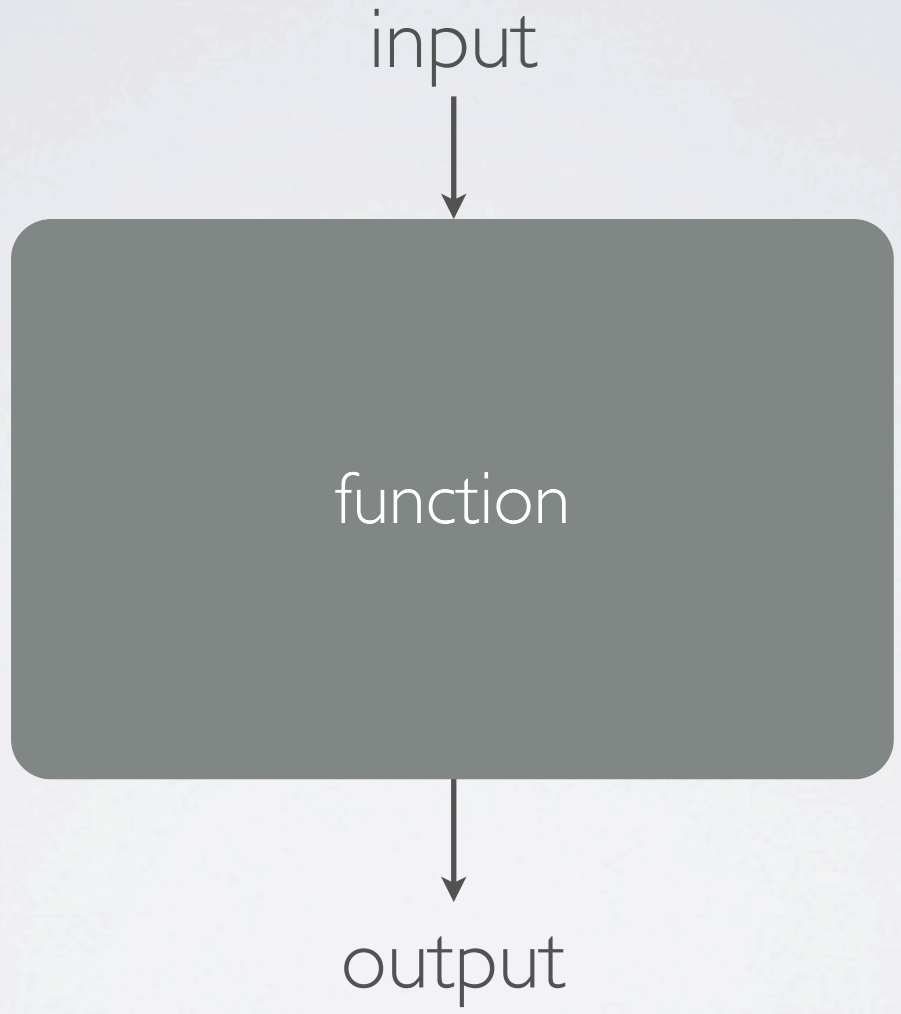
```
float box_size = 11;
float box_space = 12;
int margin = 7;

size(200, 200);
background(0);
noStroke();

// Draw gray boxes

for (int i = margin; i < height-margin; i += box_space){
  if(box_size > 0){
    for(int j = margin; j < width-margin; j+= box_space){
      fill(255-box_size*10);
      rect(j, i, box_size, box_size);
    }
    box_size = box_size - 0.6;
  }
}
```

```
float box_size = 11;
float box_space = 12;
int margin = 7;

size(200, 200);
background(0);
noStroke();

// Draw gray boxes

for (int i = margin; i < height-margin; i += box_space){
  if(box_size > 0){
    for(int j = margin; j < width-margin; j+= box_space){
      fill(255-box_size*10);
      rect(j, i, box_size, box_size);
    }
    box_size = box_size - 0.6;
  }
}
```

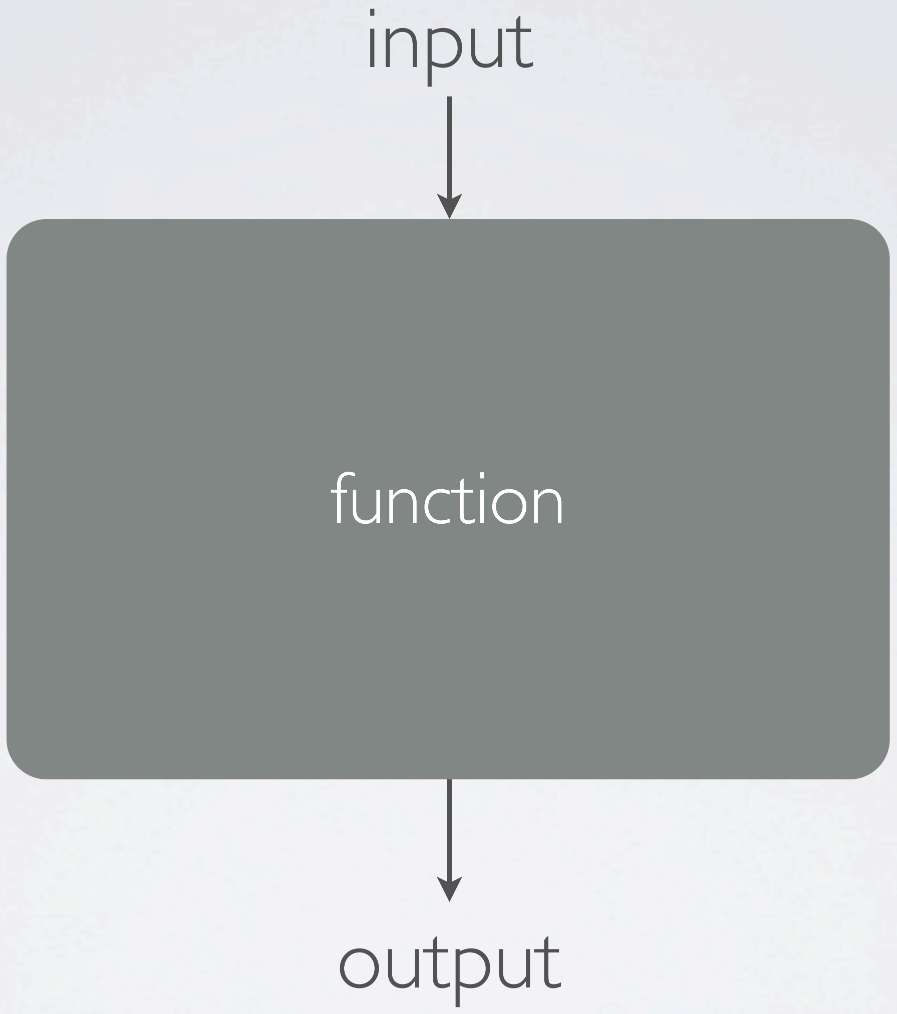
```
float box_size = 11;
float box_space = 12;
int margin = 7;

size(200, 200);
background(0);
noStroke();

// Draw gray boxes

for (int i = margin; i < height-margin; i += box_space){
  if(box_size > 0){
    for(int j = margin; j < width-margin; j+= box_space){
      fill(255-box_size*10);
      rect(j, i, box_size, box_size);
    }
    box_size = box_size - 0.6;
  }
}
```

```
float box_size = 11;
float box_space = 12;
int margin = 7;

size(200, 200);
background(0);
noStroke();

// Draw gray boxes

for (int i = margin; i < height-margin; i += box_space){
  if(box_size > 0){
    for(int j = margin; j < width-margin; j+= box_space){
      fill(255-box_size*10);
      rect(j, i, box_size, box_size);
    }
    box_size = box_size - 0.6;
  }
}
```

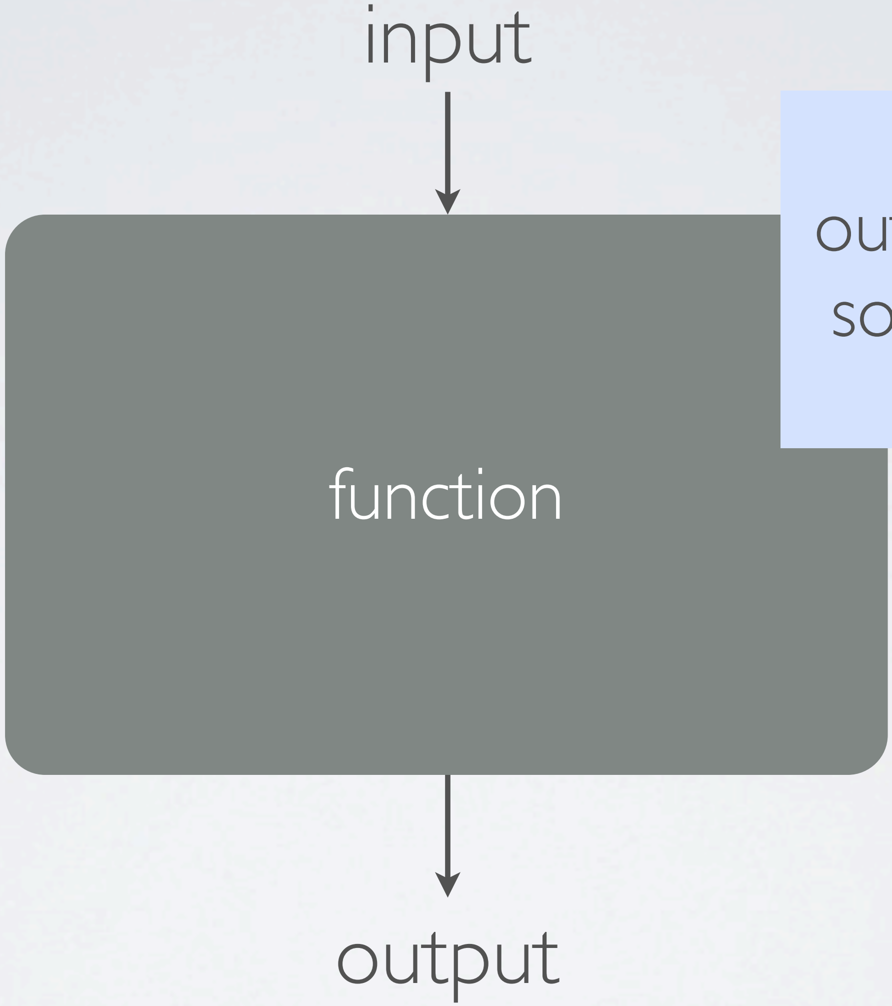
FUNCTIONS


input
function
output

A function is something that can take input, do something, and then output something.

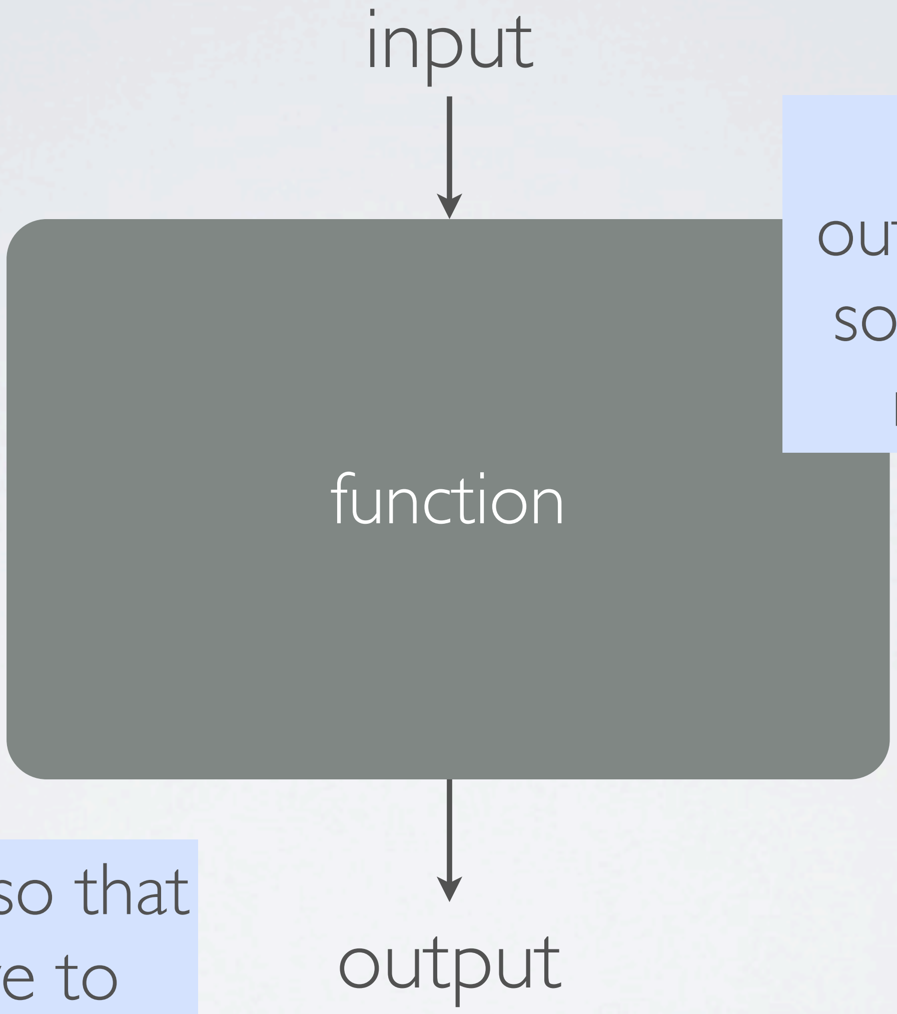


A function is something that can take input, do something, and then output something.



The input and output are optional, some functions do not have both.

A function is something that can take input, do something, and then output something.



The input and output are optional, some functions do not have both.

Functions exist so that you don't have to write a lot of code.

When you call the function size( )

```
size(300, 400);
```

When you call the function size( )

```
size(300, 400);
```

→ it creates a window with the parameters you entered

When you call the function size( )

```
size(300, 400);
```

$\Rightarrow$ it creates a window with the parameters you entered

and then the program continues with the next line of code.

When you call the function size( )

`size(300, 400);` $\Rightarrow$ it creates a window with the parameters you entered

and then the program continues with the next line of code.

It has a return type of `void,` so there's nothing given back directly to your program, but it does some work for you. It created the window.

When you call the function size( )

`size(300, 400);` $\Rightarrow$ it creates a window with the parameters you entered

and then the program continues with the next line of code.

It has a return type of `void`, so there's nothing given back directly to your program, but it does some work for you. It created the window.

In Processing, (as far as I know) all functions have a `void` return type.

`void` means that nothing is returned.

`void` means that nothing is returned.

```
void setup( ) {

}
```

`void` means that nothing is returned.

```
void setup( ) {

}
```

is typed when you want to have something happen between { }.

`void` means that nothing is returned.

```
void setup( ) {

}
```

is typed when you want to have something happen between { }.

The `void` in front of `setup` means nothing is returned after `setup( )` is finished.

In Arduino:

```
int val = 0;
int inPin = 7;
val = digitalRead(inPin);
```

In Arduino:

```
int val = 0;
int inPin = 7;
val = digitalRead(inPin);
```

function name

In Arduino:

```
int val = 0;
int inPin = 7;
val = digitalRead(inPin);
```

input parameters and type

function name

In Arduino:

```
int val = 0;
int inPin = 7;
val = digitalRead(inPin);
```

input parameters and type

function name

an int is returned, so it needs to be stored somewhere

FINAL EXERCISE

Create a Processing program that generatively draws depending on the mouse position.

Within your program use:

- Variables
- For or while loop
- If or if/else

Start with this code.

```
void setup() {
  // create the window
  size(400, 400);
}

void draw() {
  // set the colour
  fill(10, 10, 255);

  // draw the circle
  ellipse(mouseX, mouseY,
  100, 100);
}
```