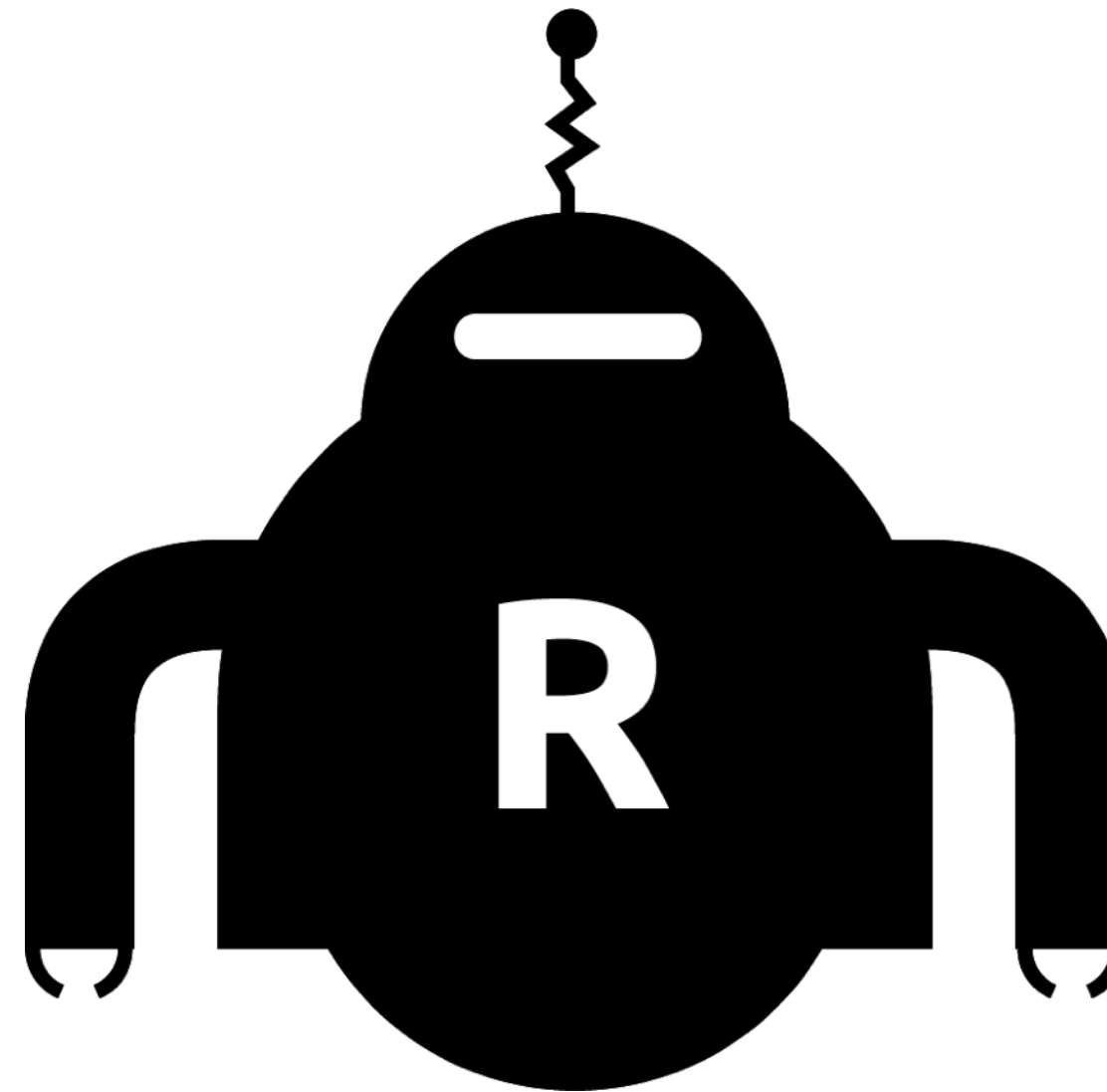
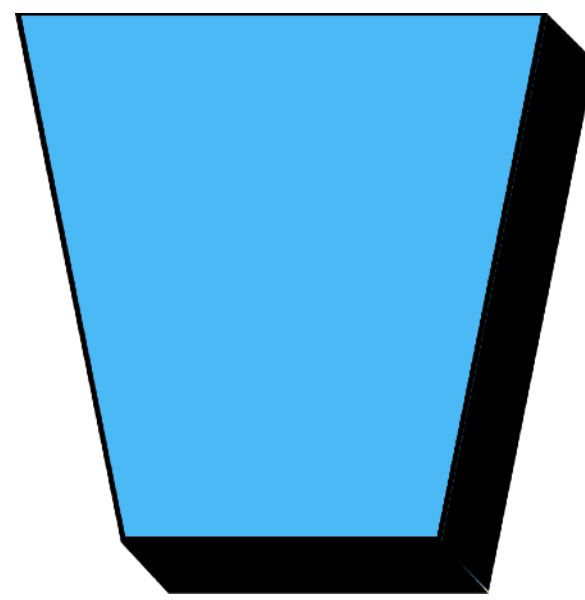


programming for the visually oriented

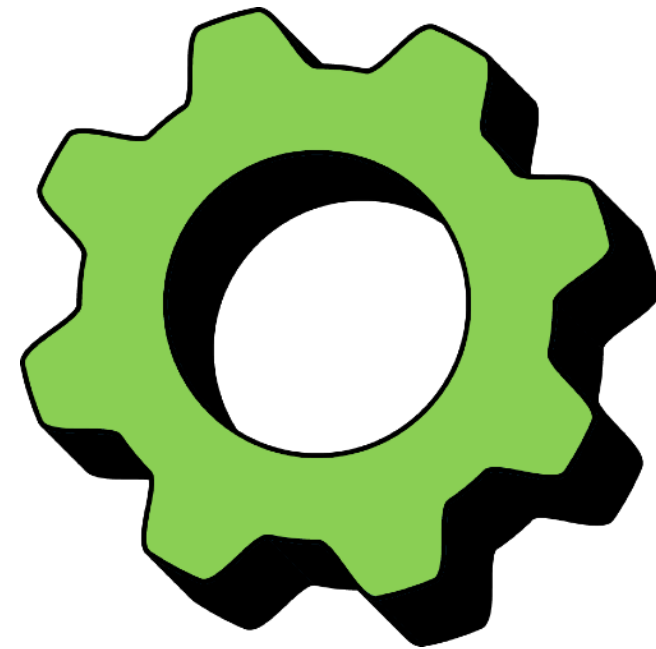
C# in Unity3D
version 1, 2013
carl emil carlsen
<http://sixthsensor.dk>



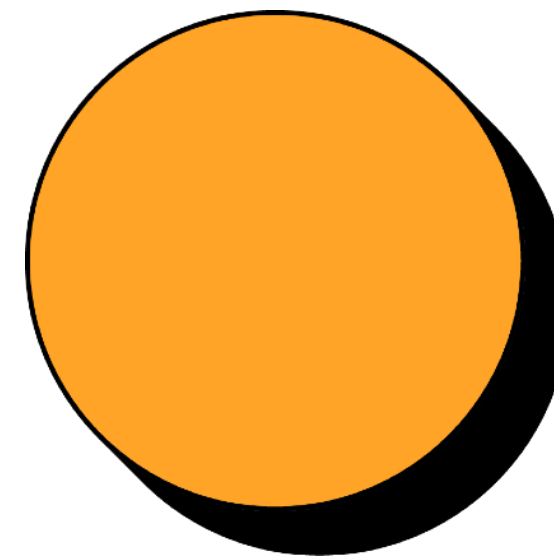
rob the robot will be reading your code



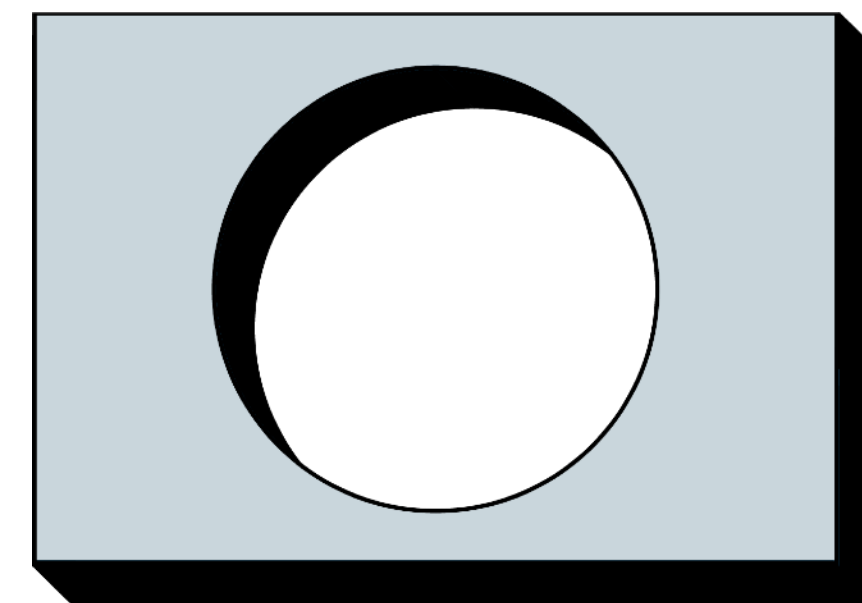
variable



function

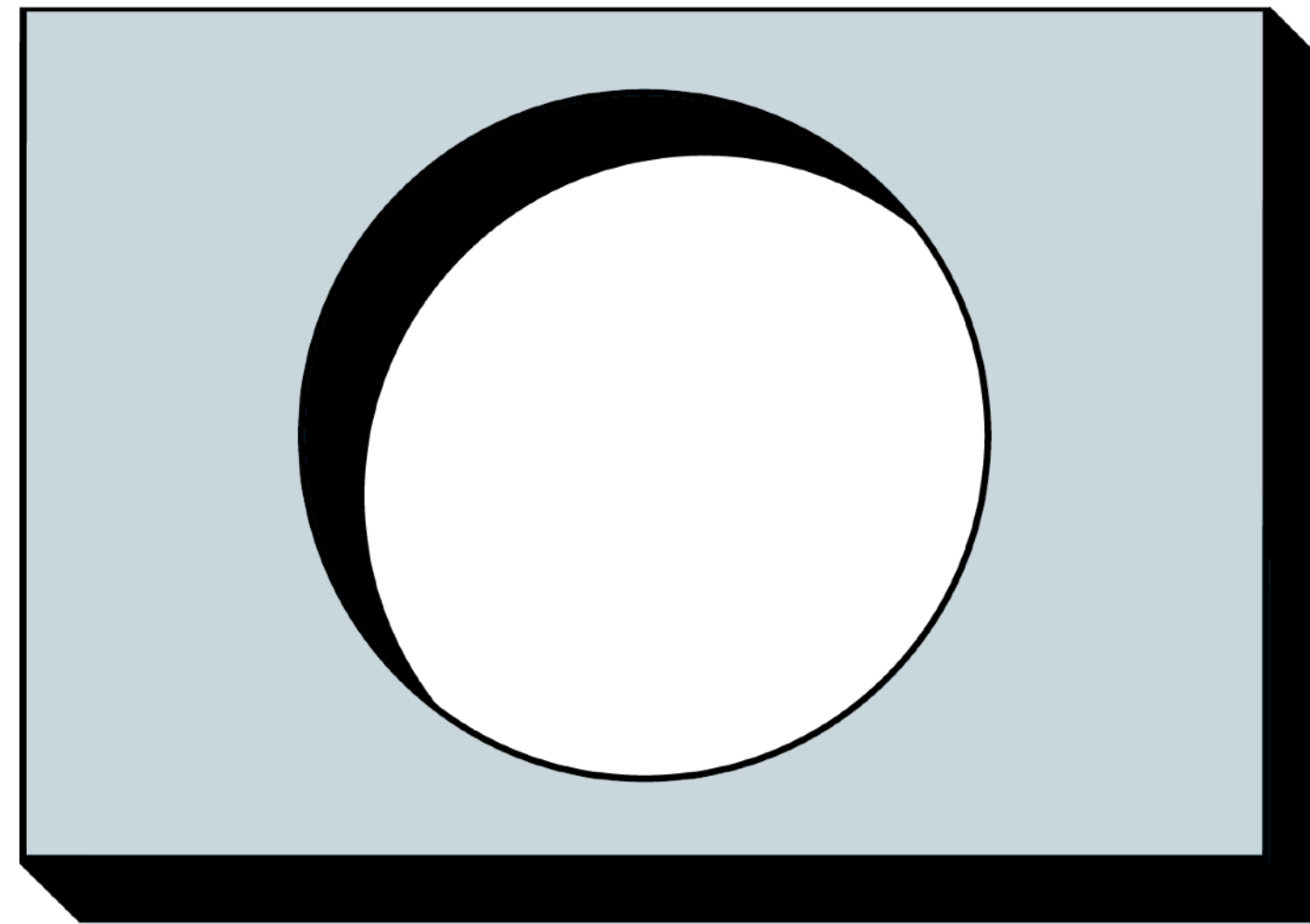


object

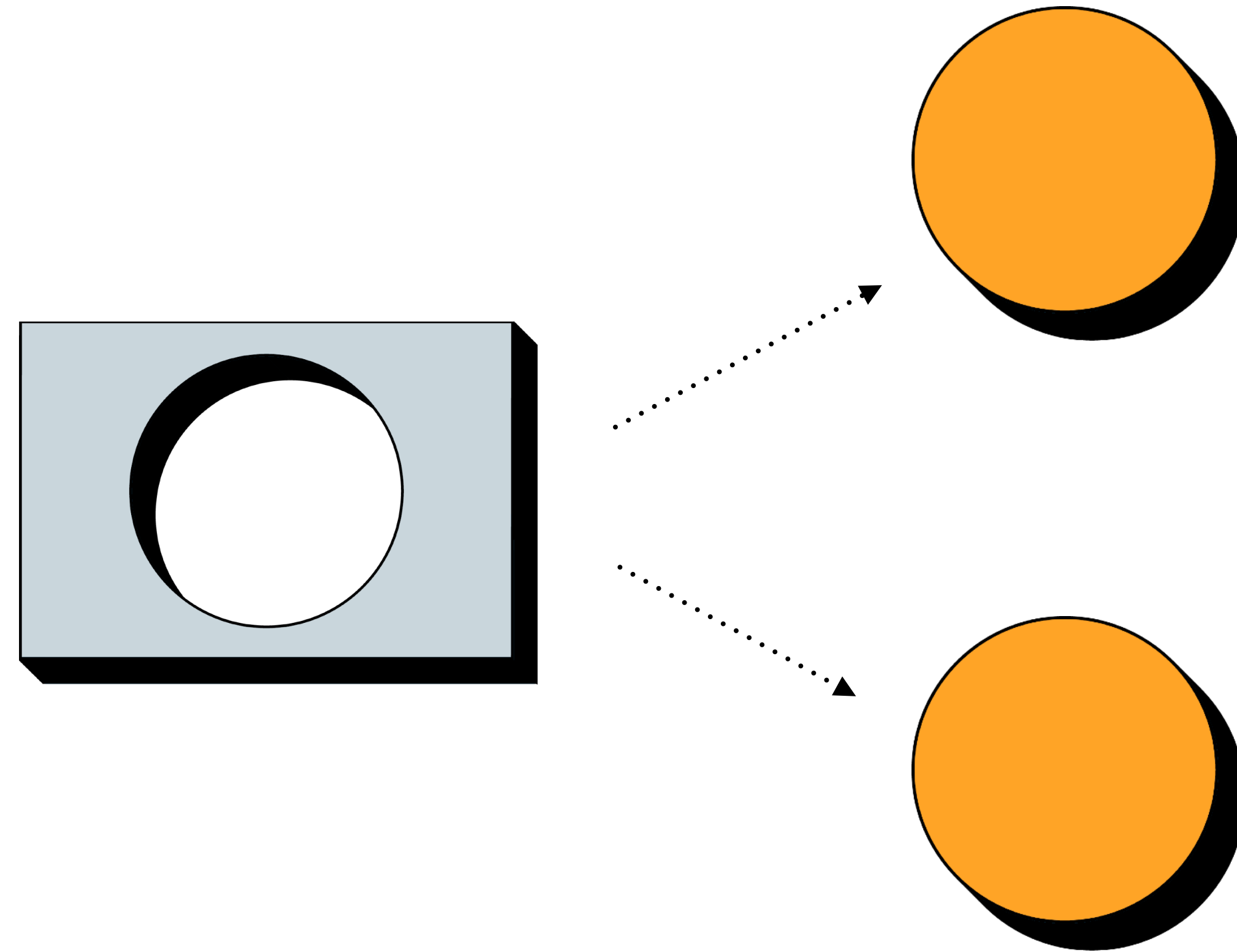


class

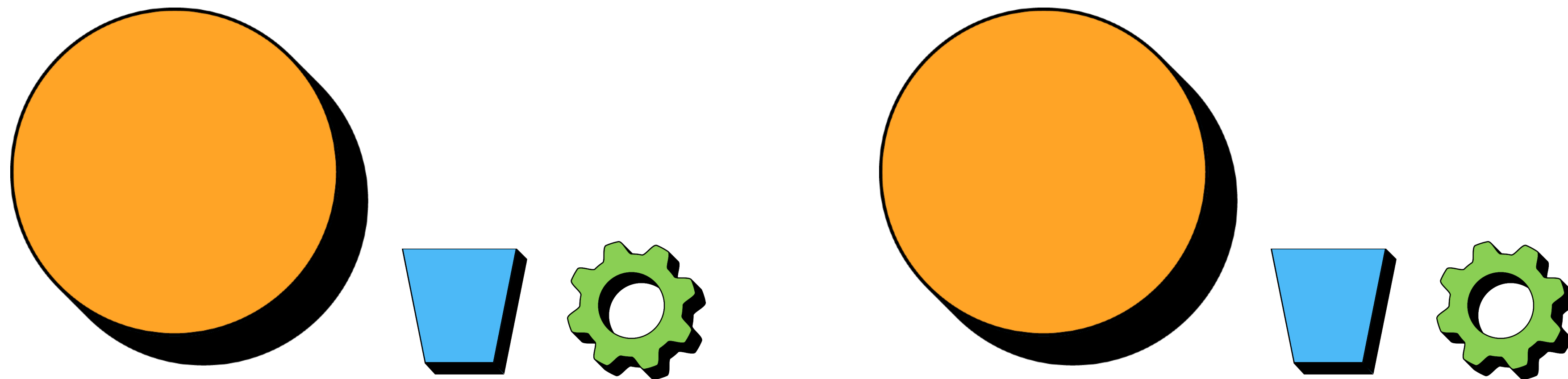
basic concepts



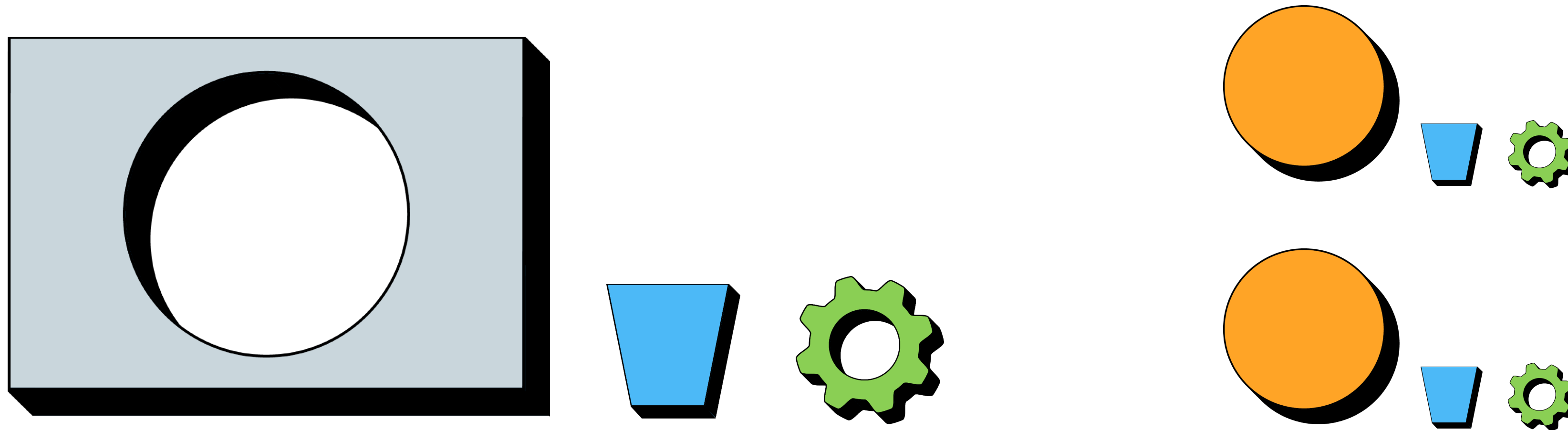
a class defines a type of object



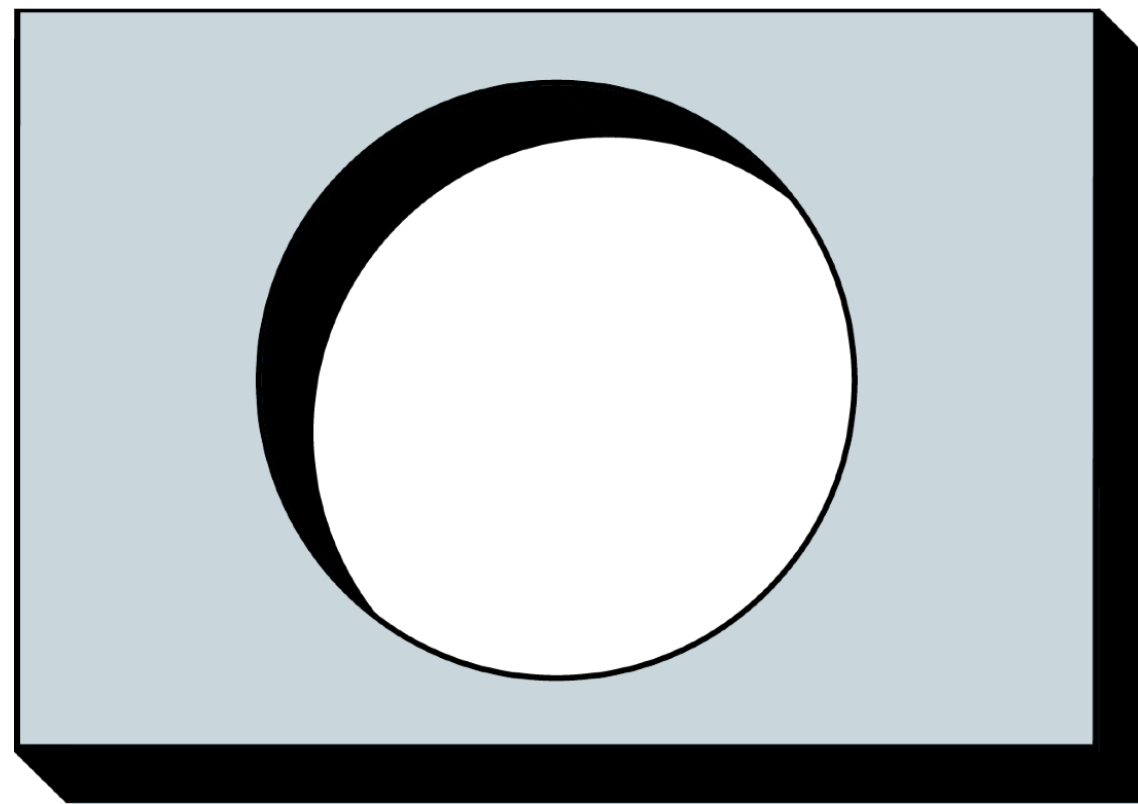
objects are "instantiated" from a class



each object has it's own variables and functions



a class can have it's own variables and functions too

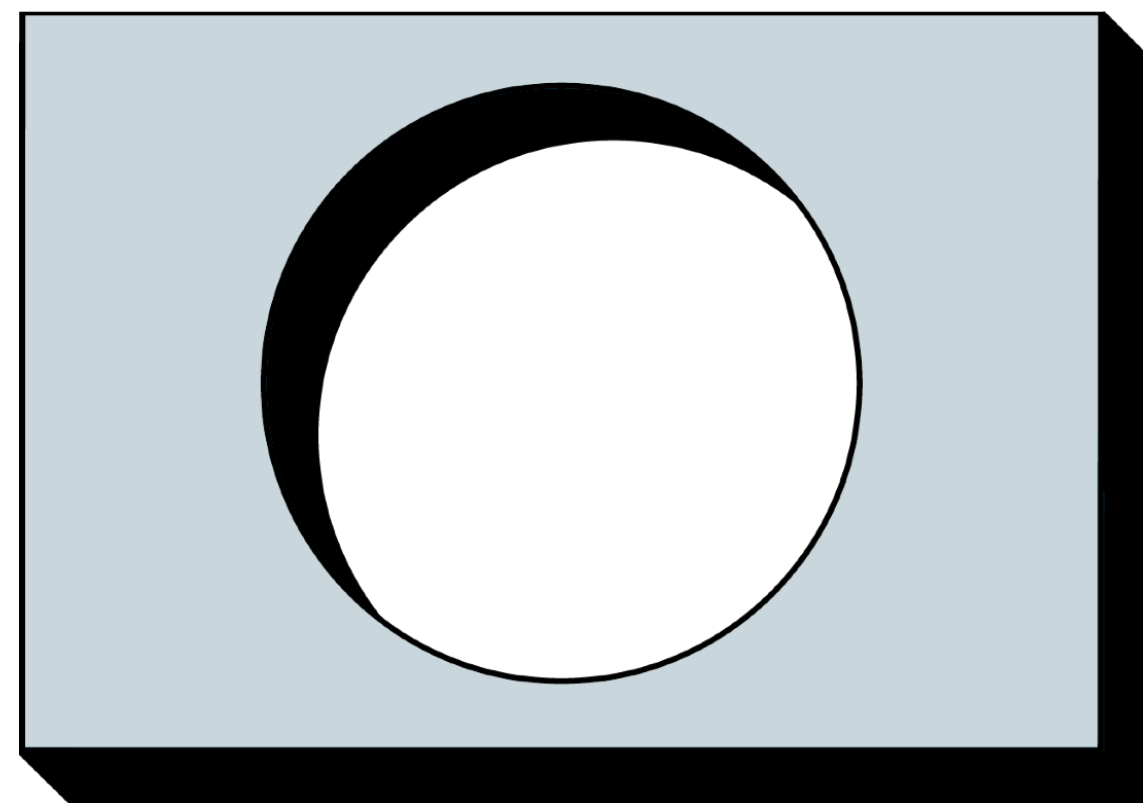


Debug

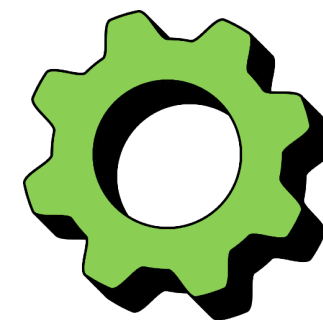


Log

class function example



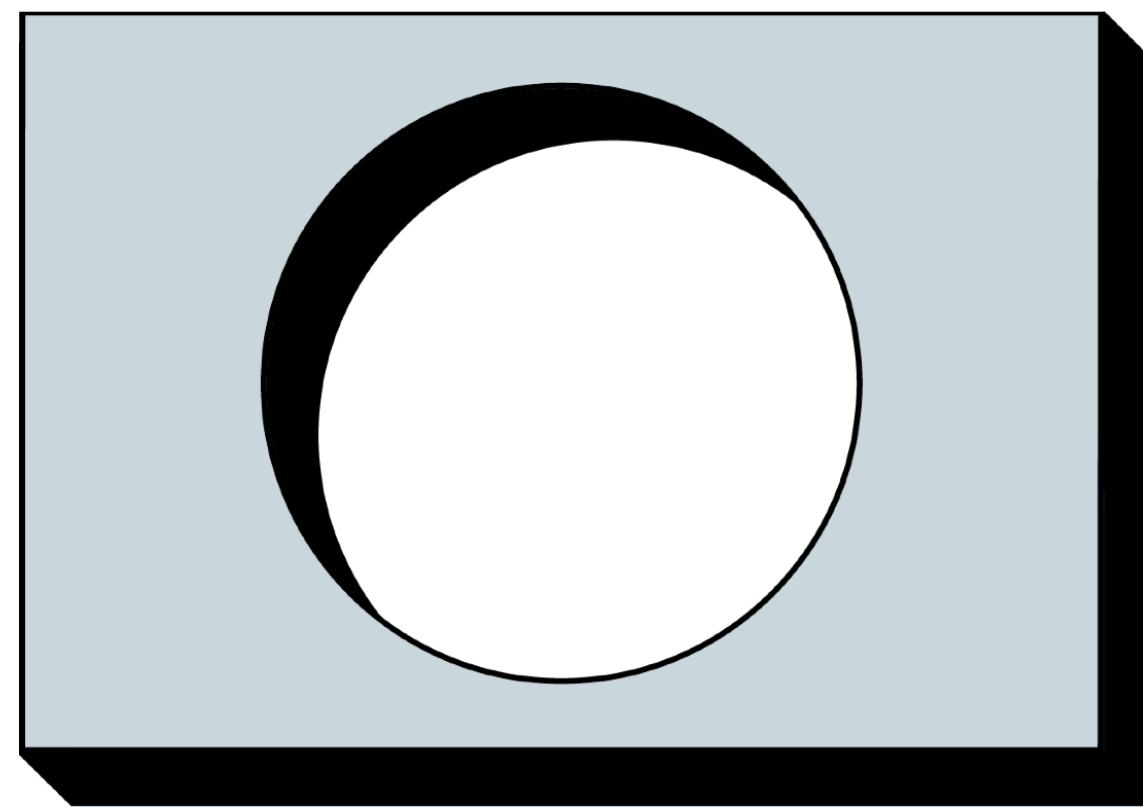
Debug



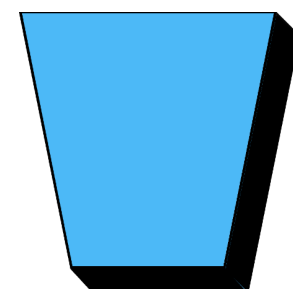
Log

```
Debug.Log("hallo world");
```

class function example

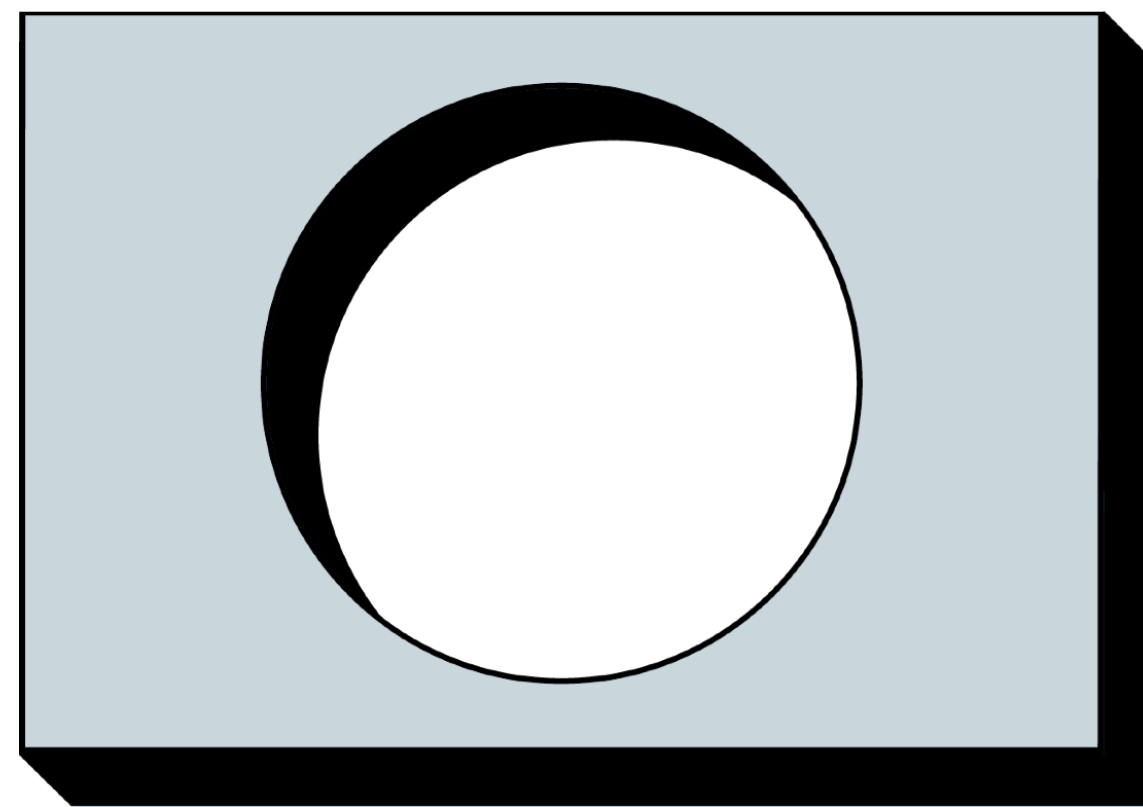


Random

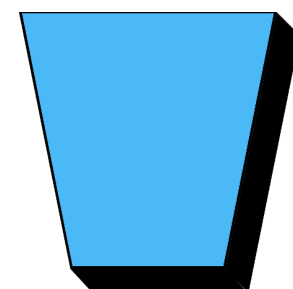


value

class variable example



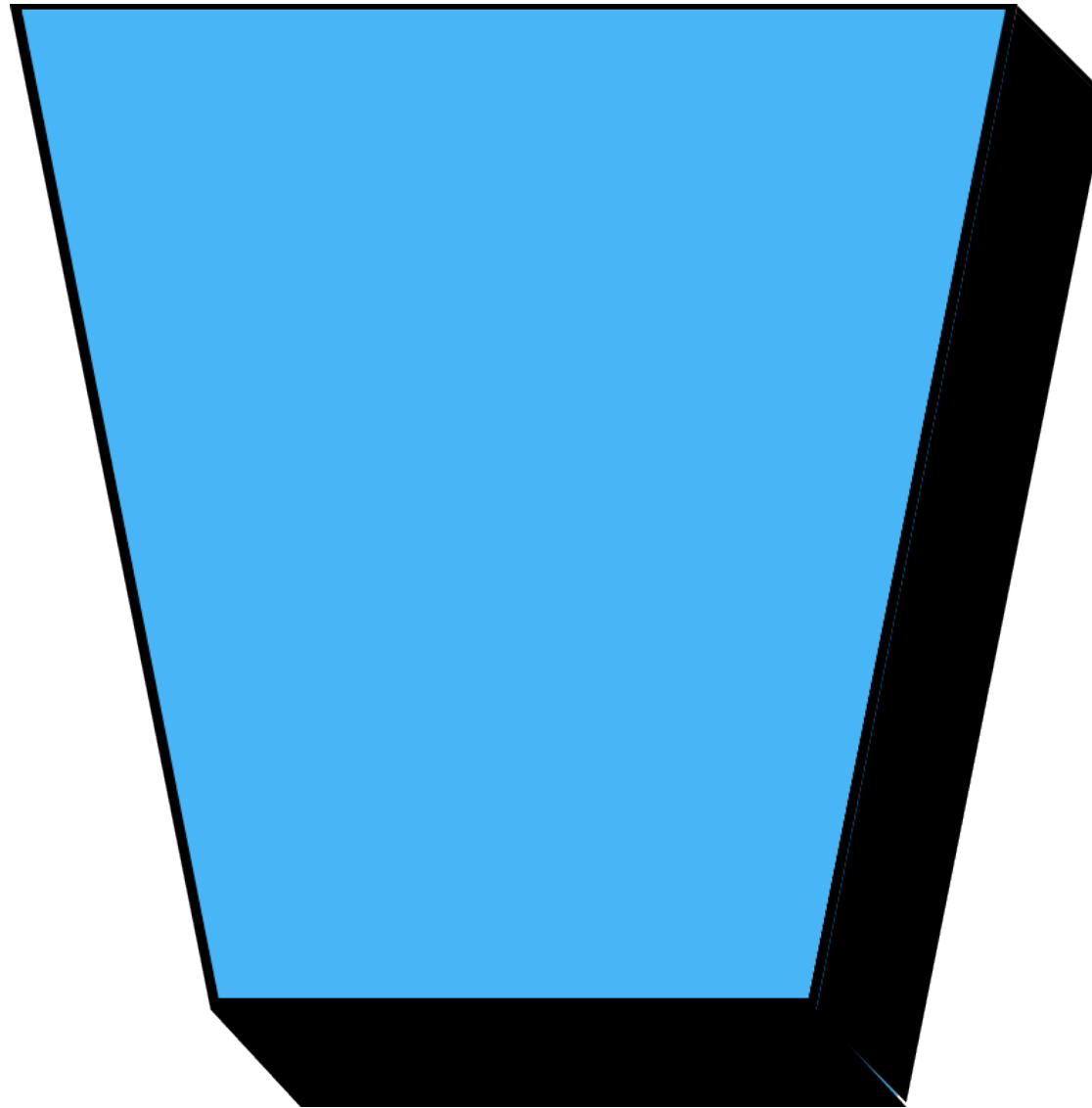
Random



value

`Random.value;`

class variable example



a variable is like a bin where
you can store information

```
int age;
```

defining a variable

```
int age;
```

data type

name

end of line

defining a variable

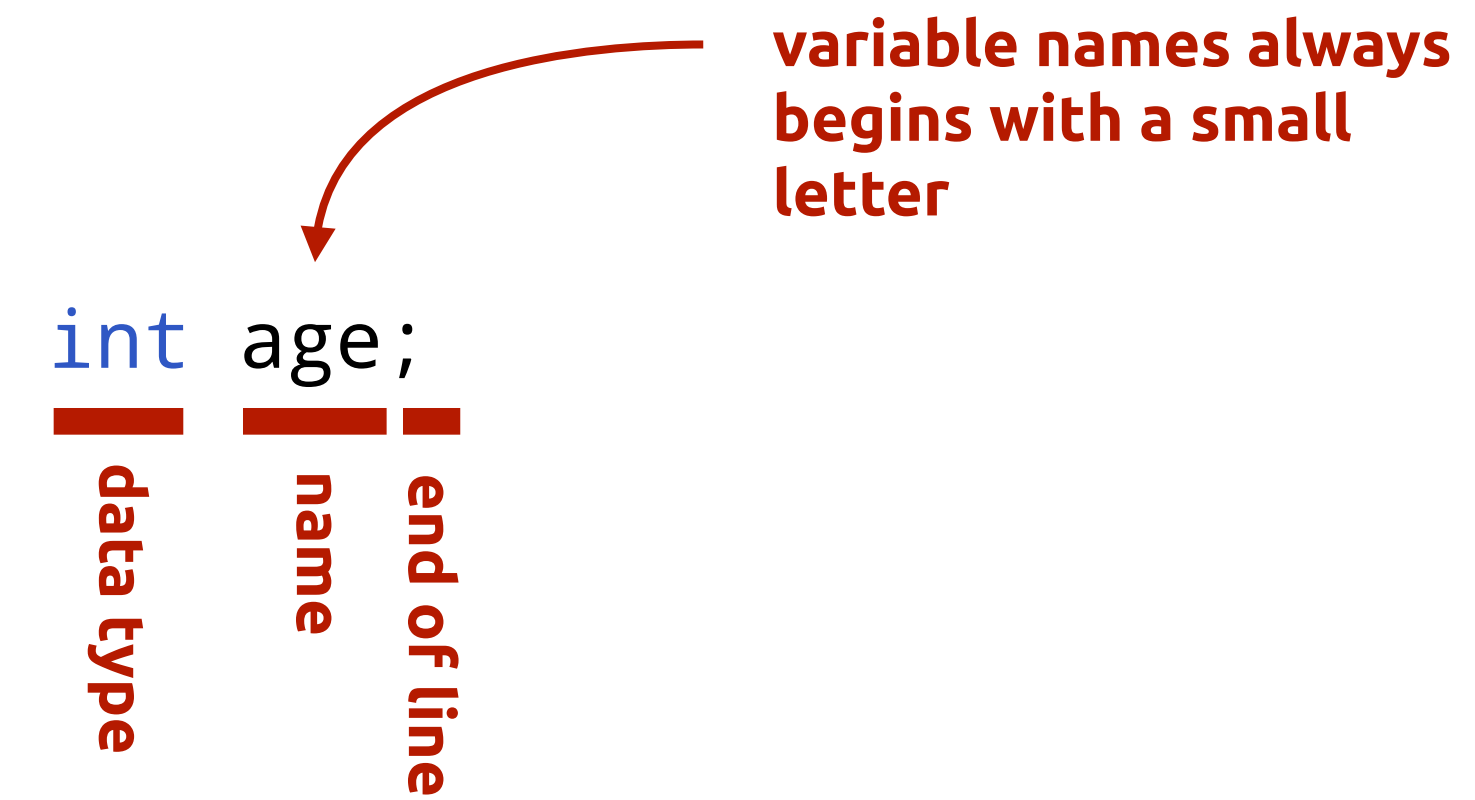
`int age;`

data type

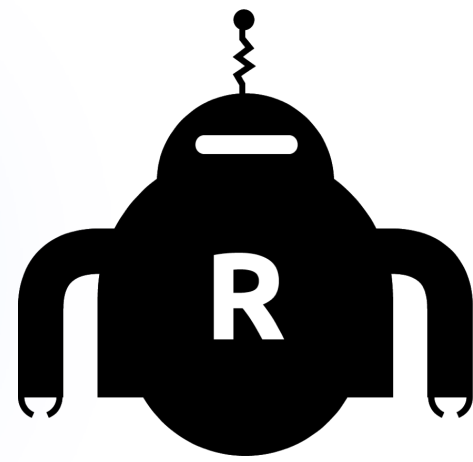
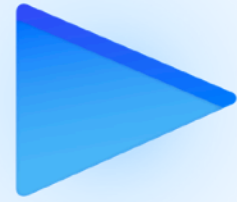
name

end of line

variable names always begins with a small letter

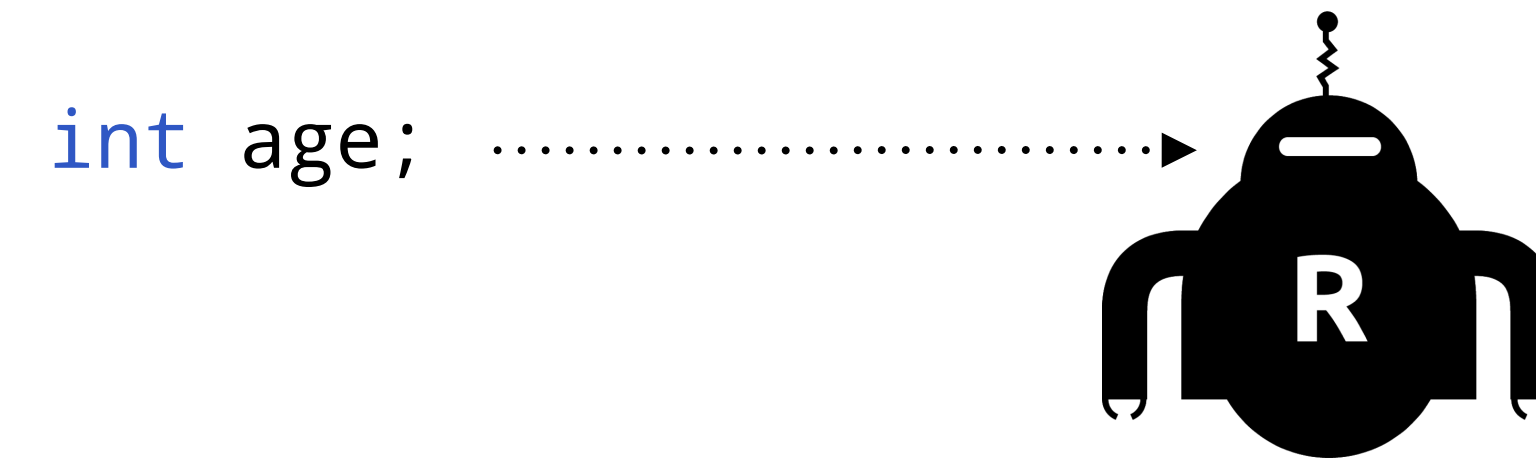
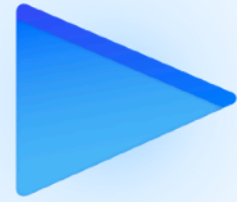


defining a variable



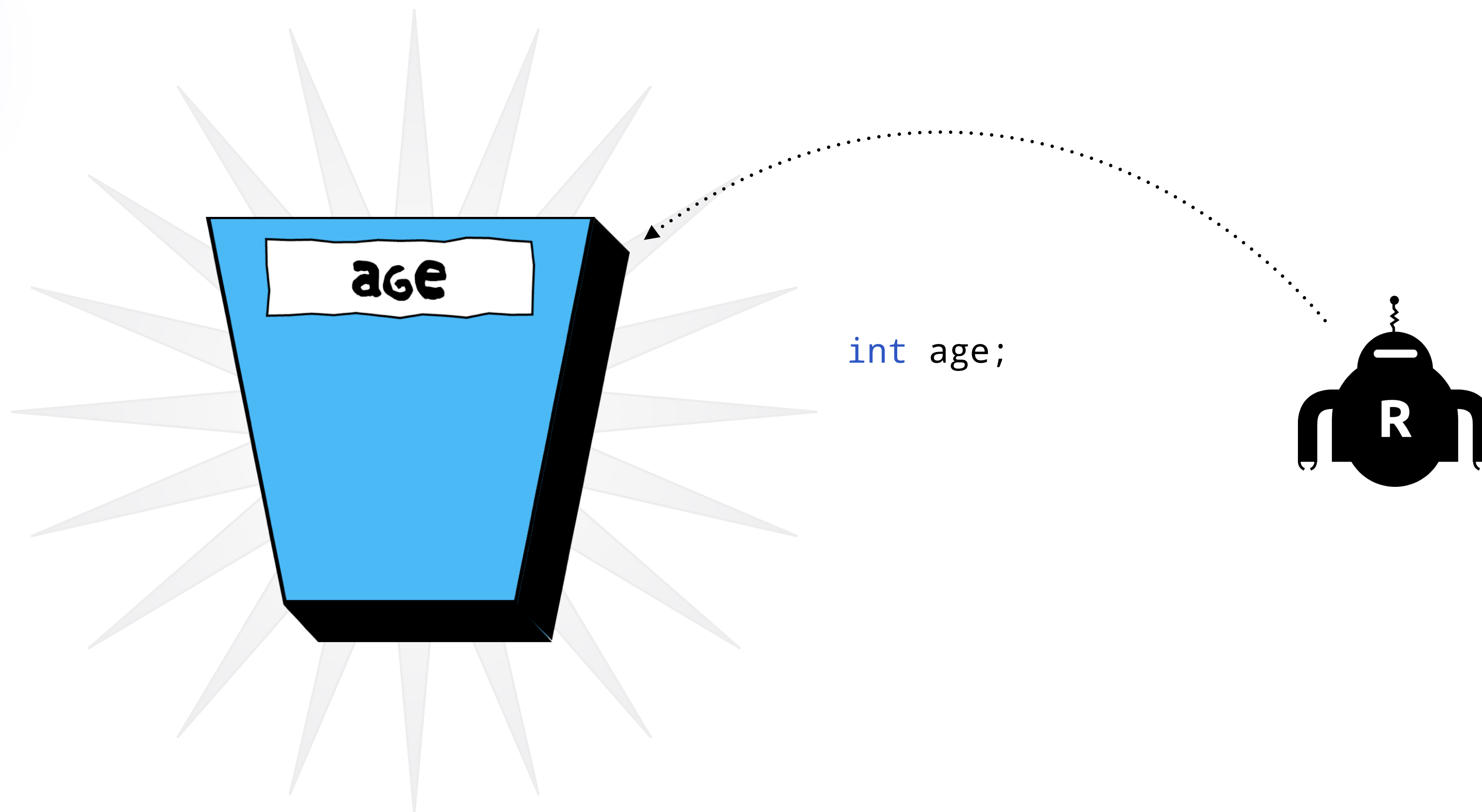
```
int age;
```

defining a variable



ok; i'll dedicate
space in the
computers memory
for a variable
named 'age'

defining a variable



defining a variable

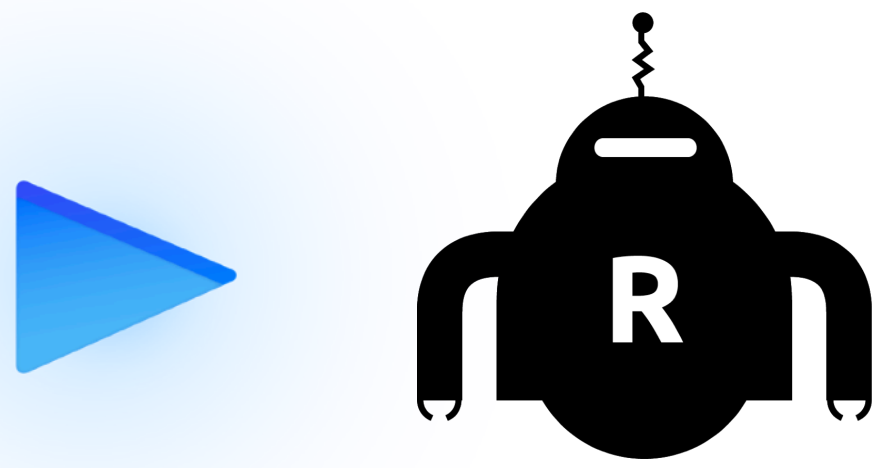
```
int age;  
age = 29;
```

setting a variable

```
int age;  
age = 29;
```

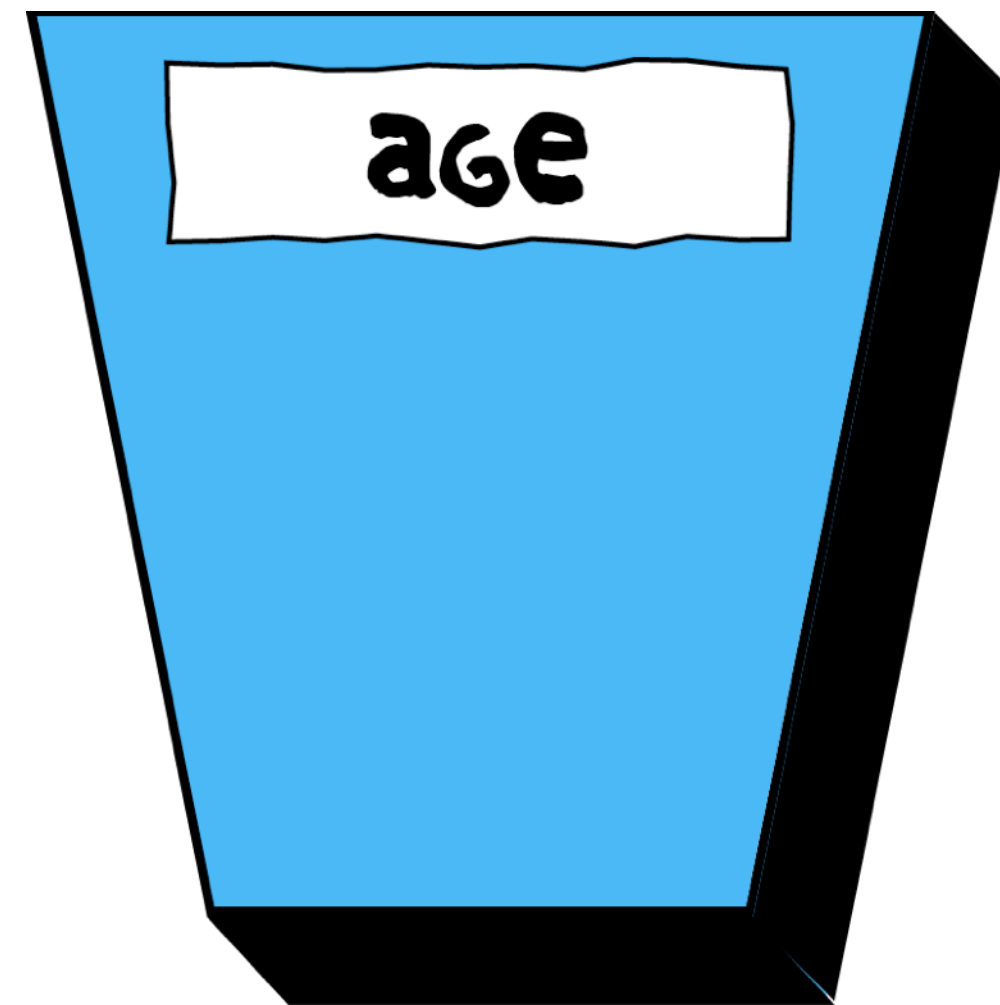
variable name
assignment operator
integer value
end of line

setting a variable

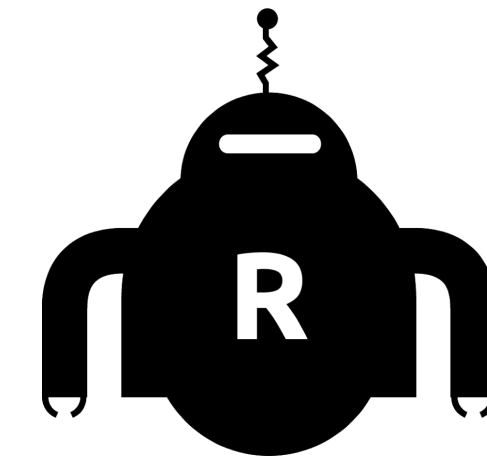


```
int age;  
age = 29;
```

setting a variable

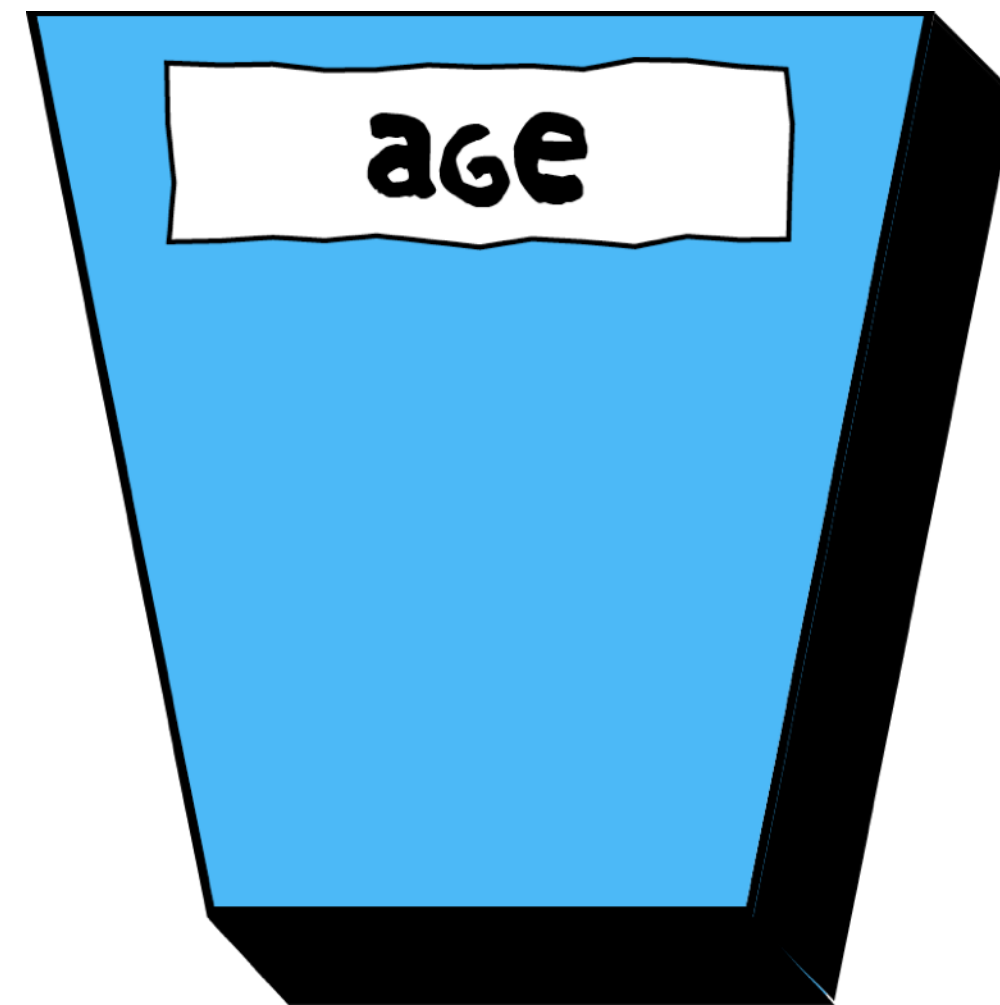
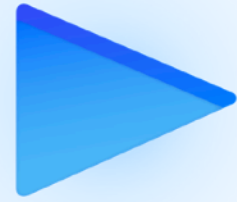


```
int age;  
age = 29;
```

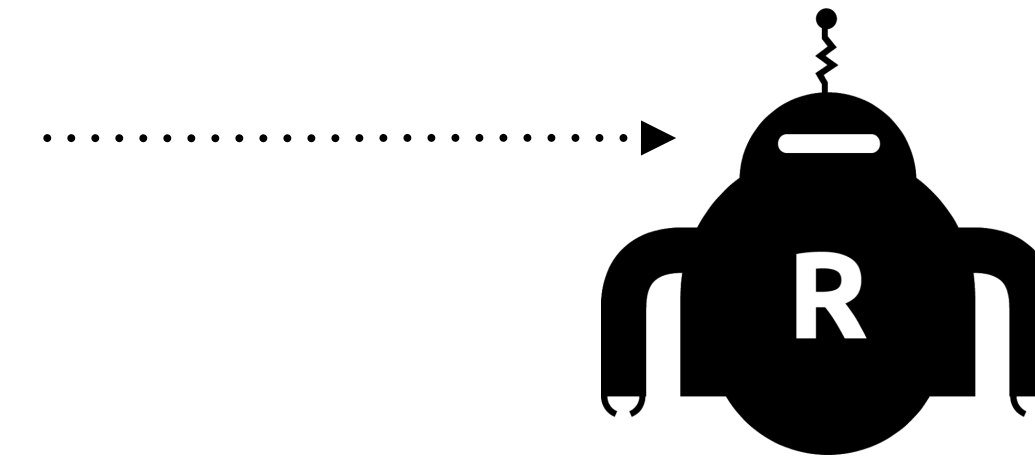


so far i've
created an
integer variable
named age.

setting a variable

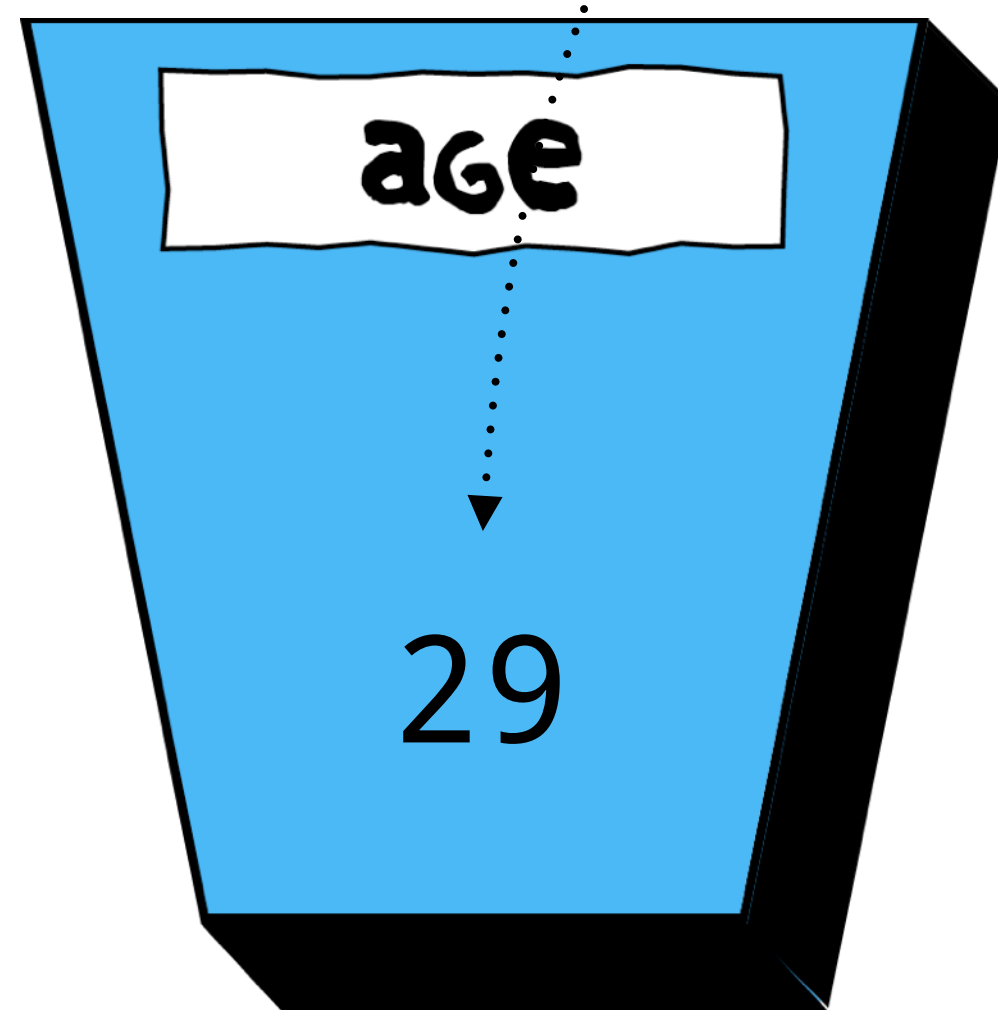


```
int age;  
age = 29;
```

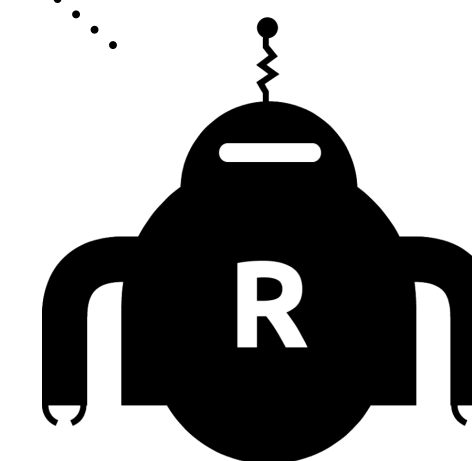


ok; i'll store
the value '29'
in the variable
named 'age'

setting a variable



```
int age;  
age = 29;
```



setting a variable

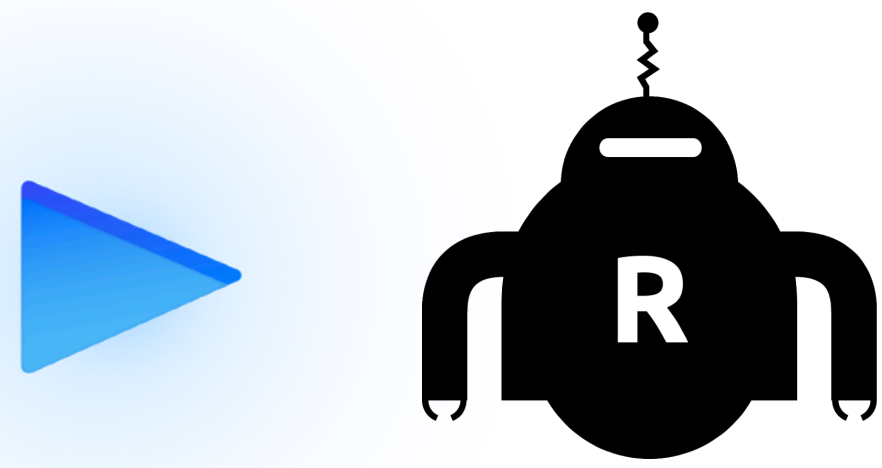

```
int age;  
age = 29;  
age = age + 1;
```

getting a variable

```
int age;  
age = 29;  
age = age + 1;
```

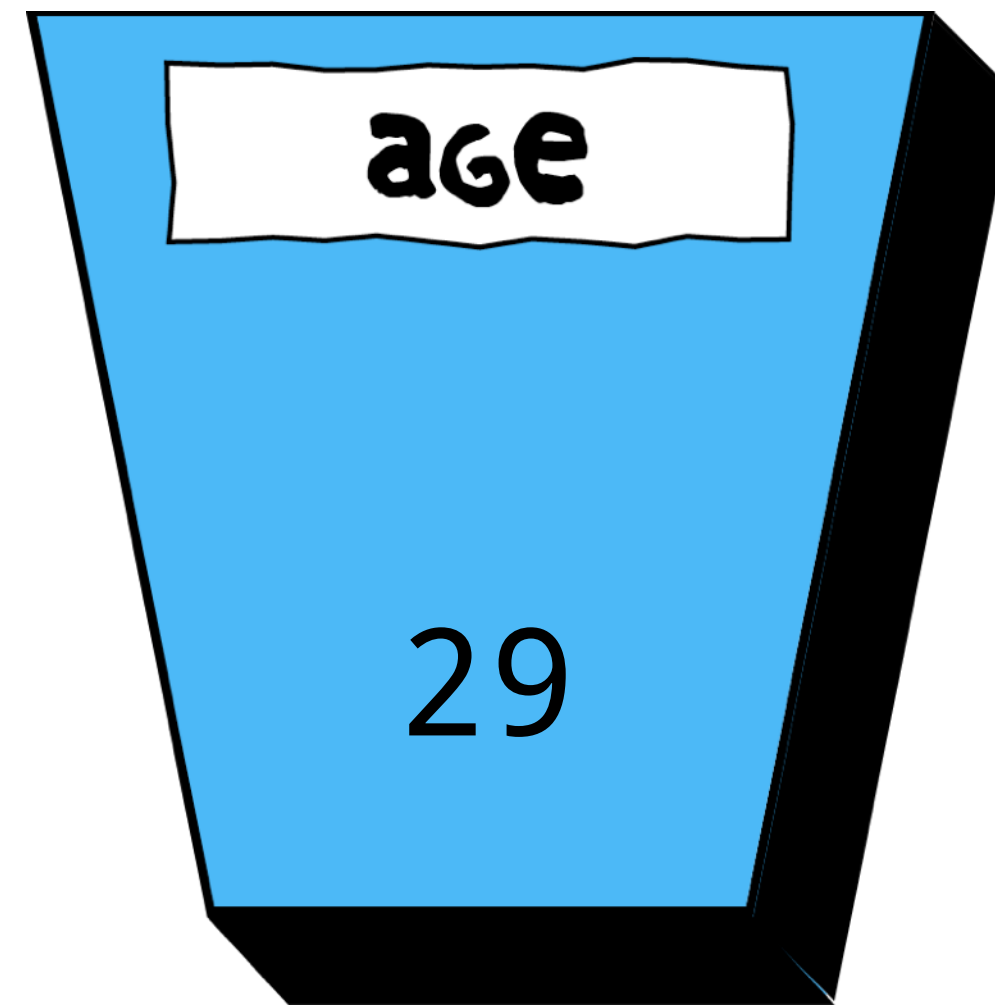
variable name
assignment operator
variable name
addition operator
integer value
end of line

getting a variable

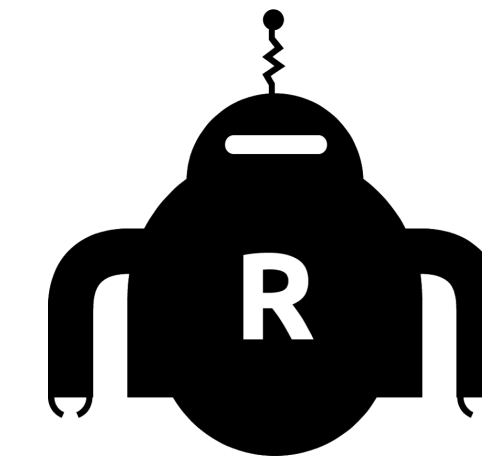


```
int age;  
age = 29;  
age = age + 1;
```

getting a variable

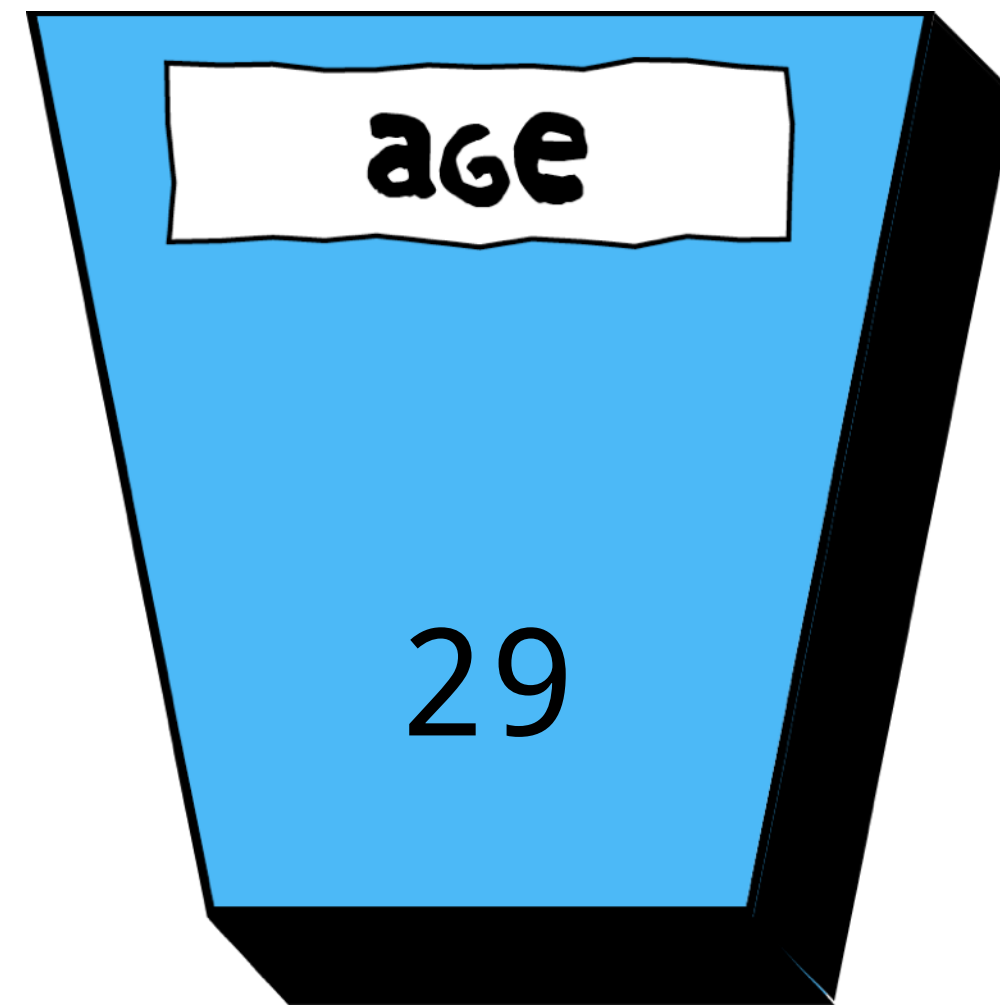


```
int age;  
age = 29;  
age = age + 1;
```

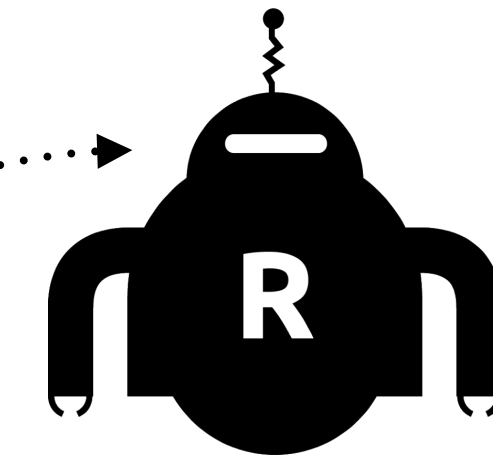


getting a variable

so far i've
created an
integer variable
named age and
set it's value
to '29'.

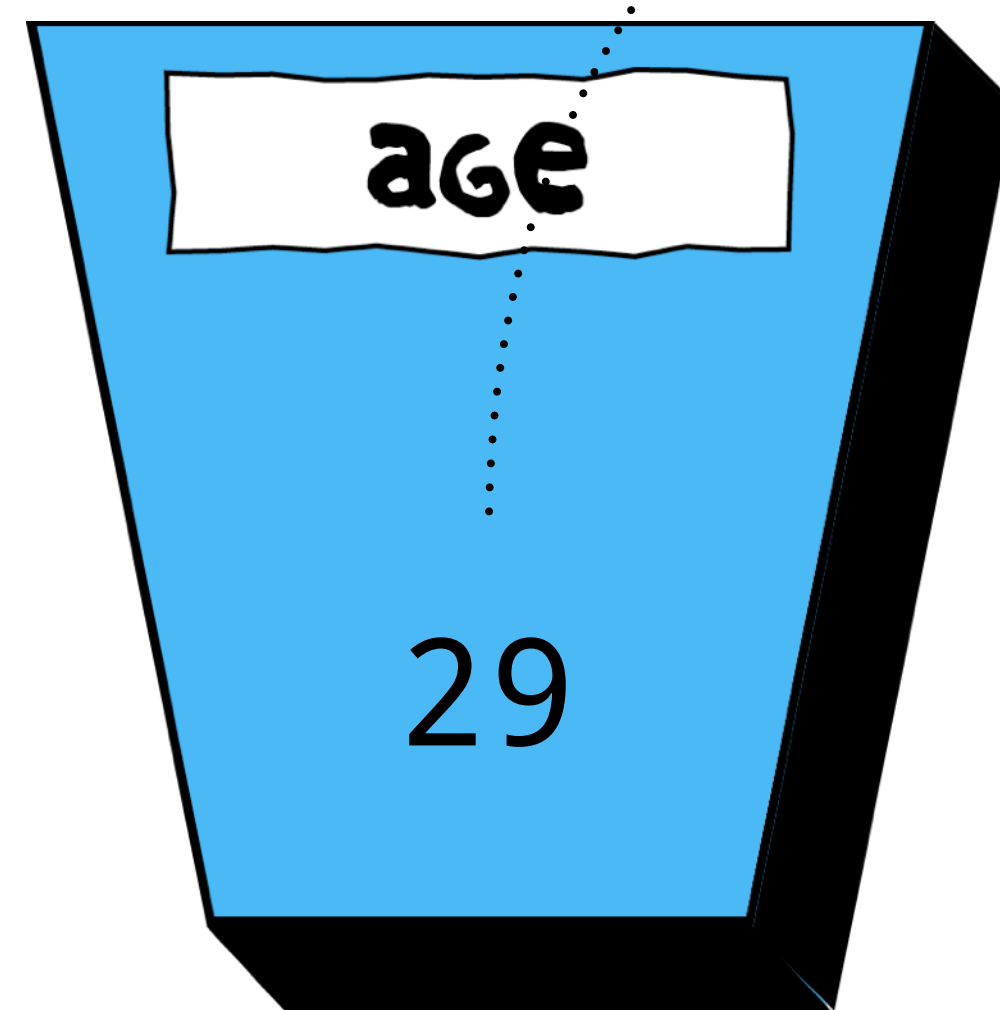


```
int age;  
age = 29;  
age = age + 1;
```

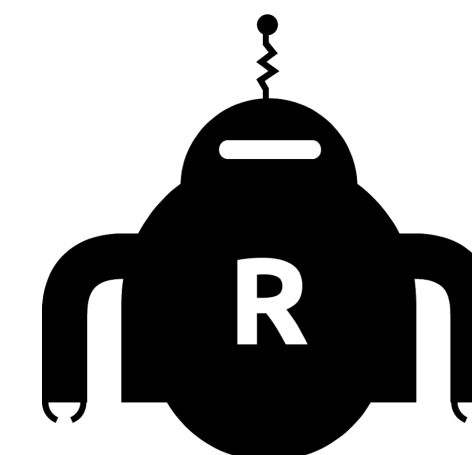


ok; i'll get
the value of
the variable
named 'age'

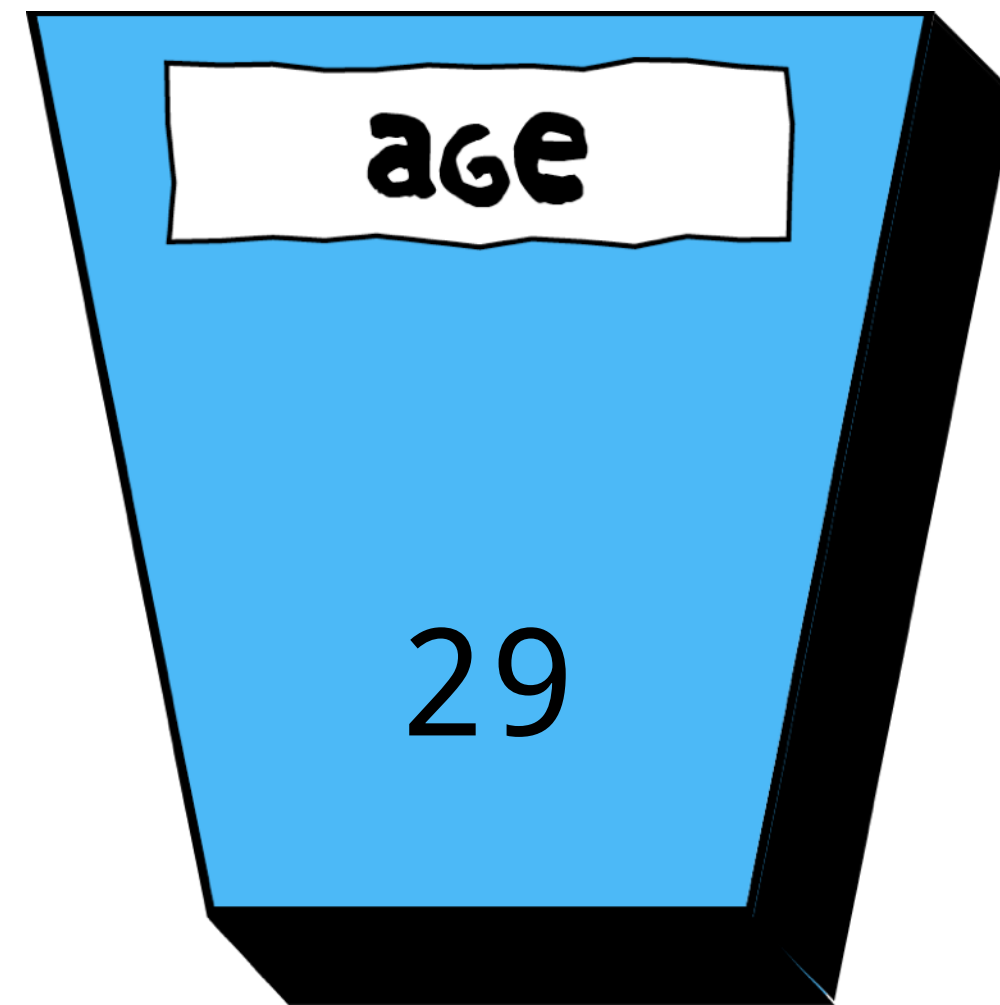
getting a variable



```
int age;  
age = 29;  
age = 29 + 1;
```

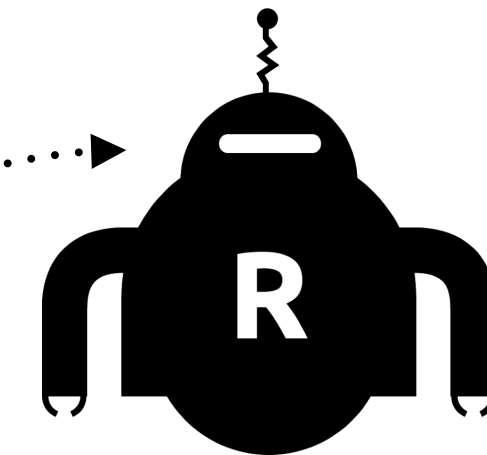


getting a variable

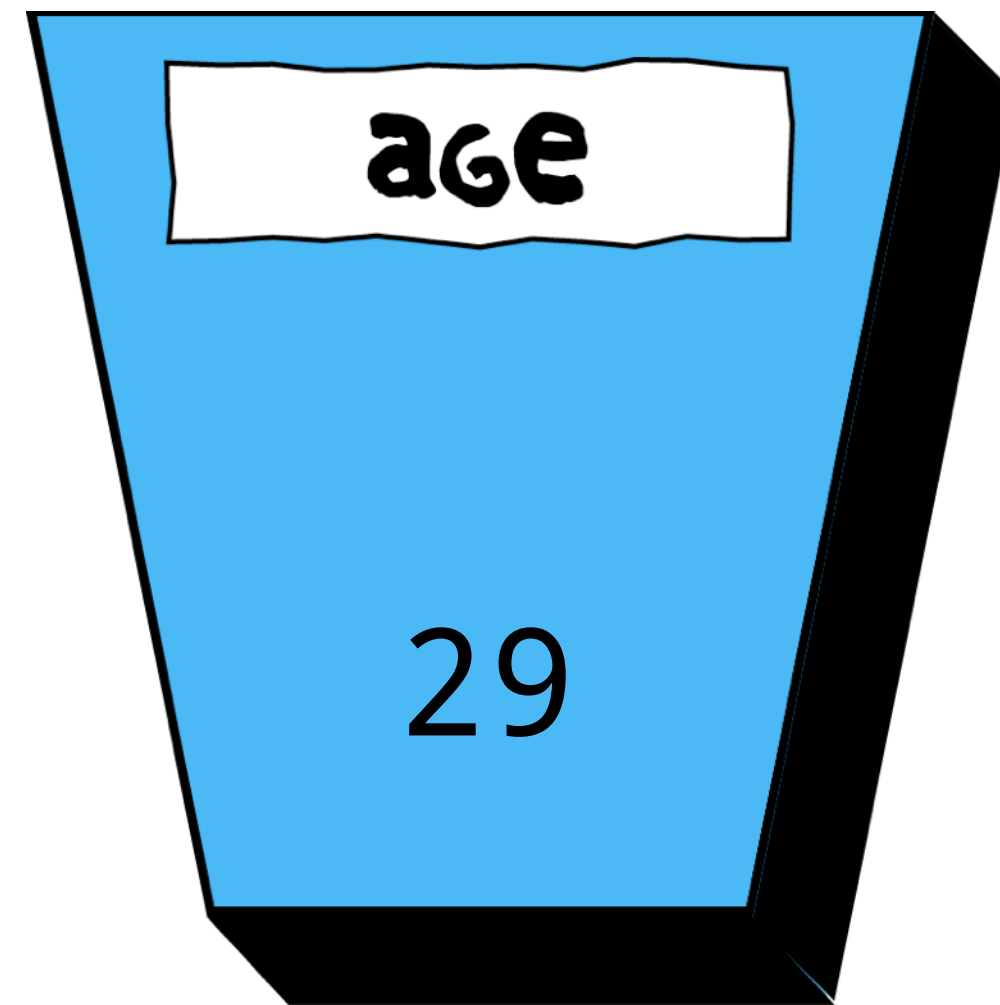


```
int age;  
age = 29;  
age = 29 + 1;
```

ok; i'll do
the math for
you

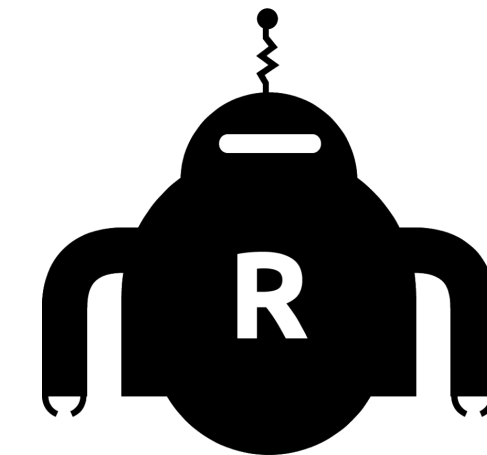


getting a variable

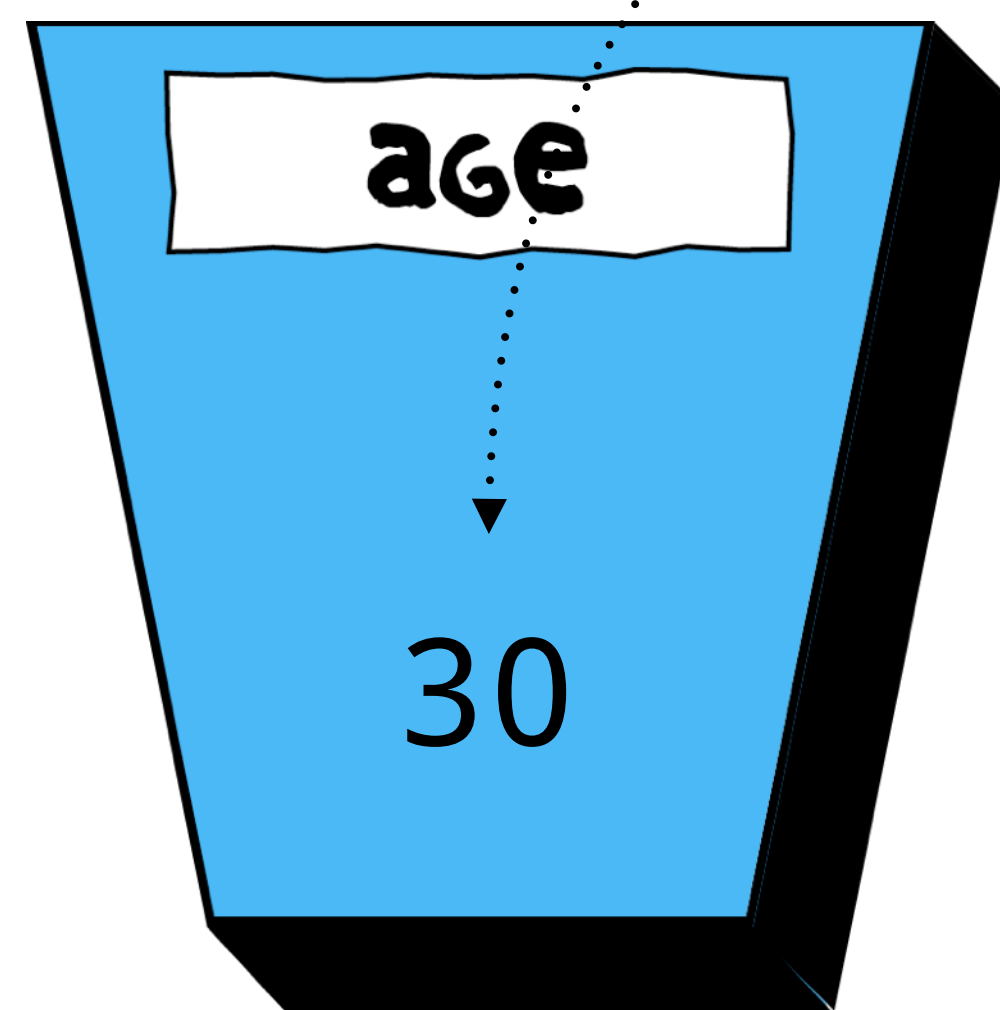


```
int age;  
age = 29;  
age = 30;
```

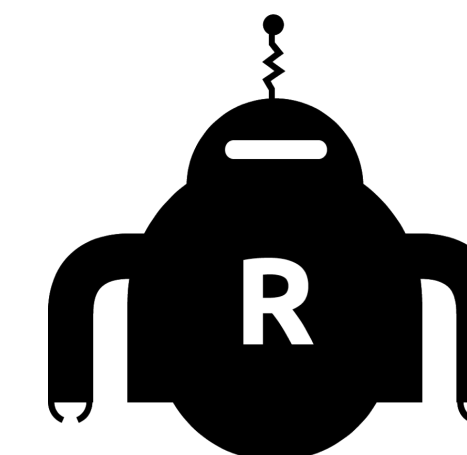
ok; now i'll
store the result
in the variable
named age



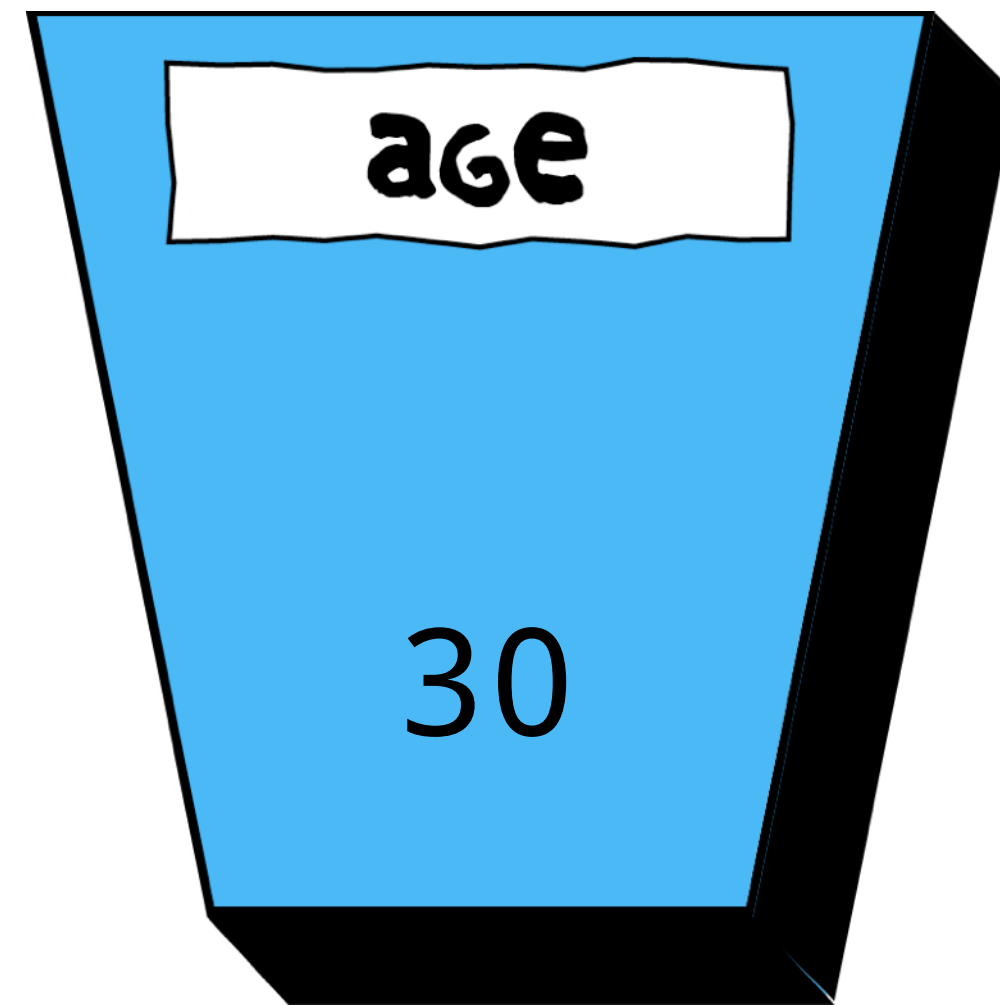
getting a variable



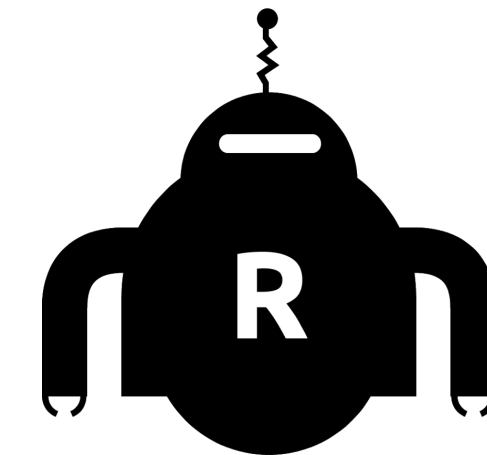
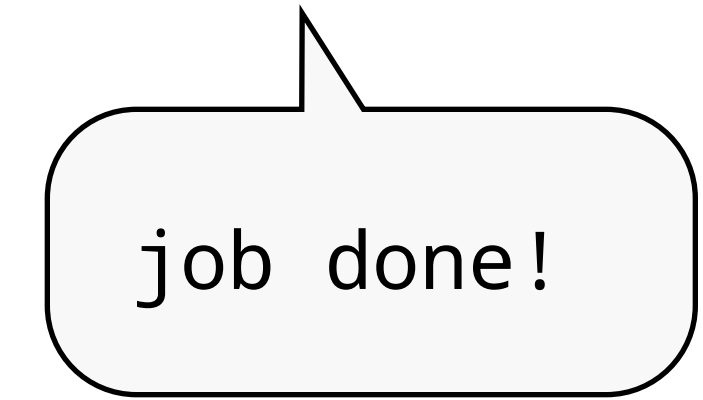
```
int age;  
age = 29;  
age = 30;
```



getting a variable



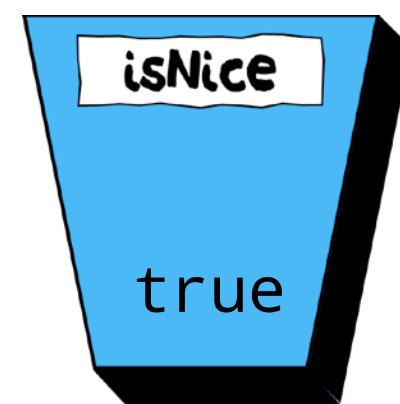
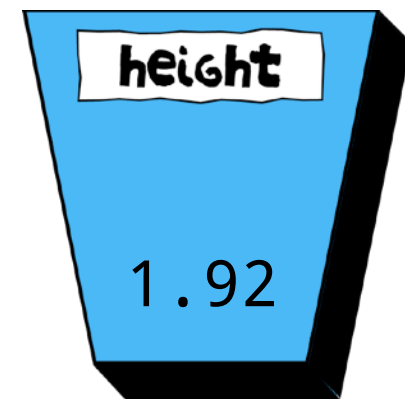
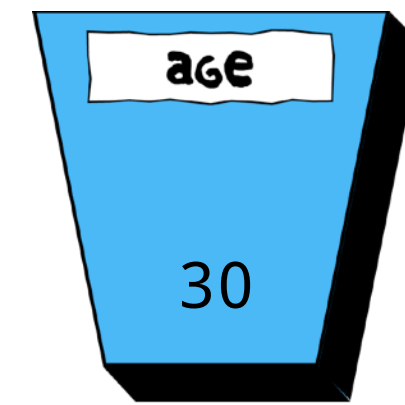
```
int age;  
age = 29;  
age = age + 1;
```



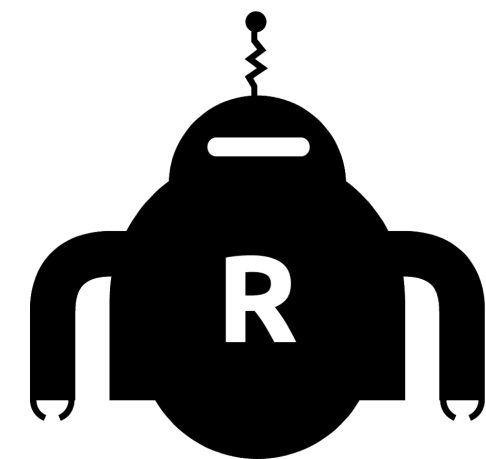
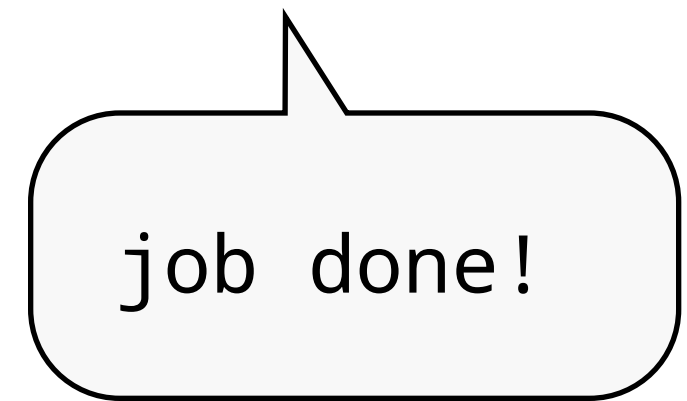
getting a variable

```
int age = 35;  
float height = 1.92f;  
string name = "Carl";  
bool likesCoffee = true;
```

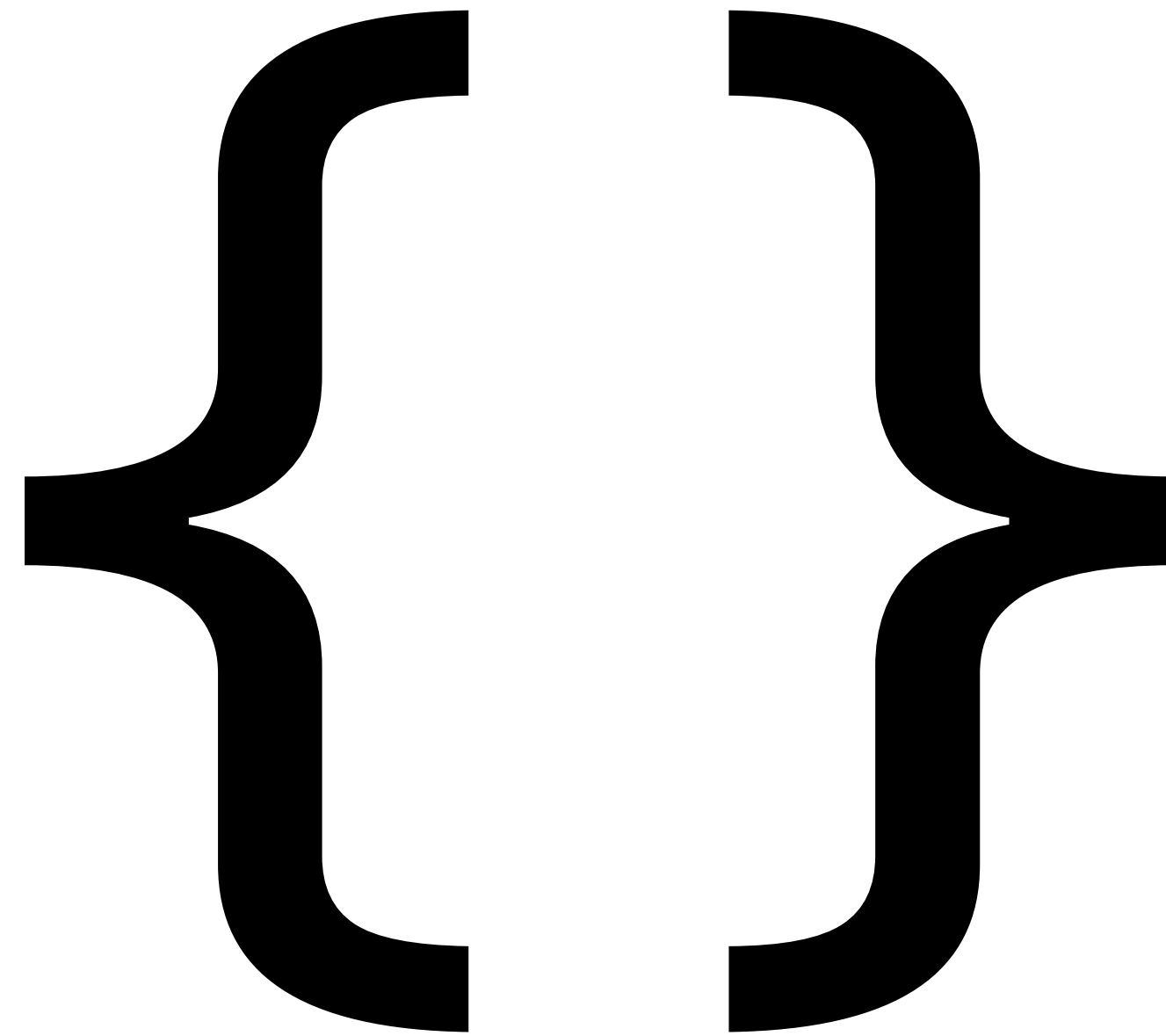
common data types



```
int age = 30;  
float height = 1.92f;  
string name = "Carl";  
bool isNice = true;
```



common data types



scopes

```
{  
    // code goes here  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
}
```

scope

```
begin {  
    // code goes here  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
    //  
end }
```

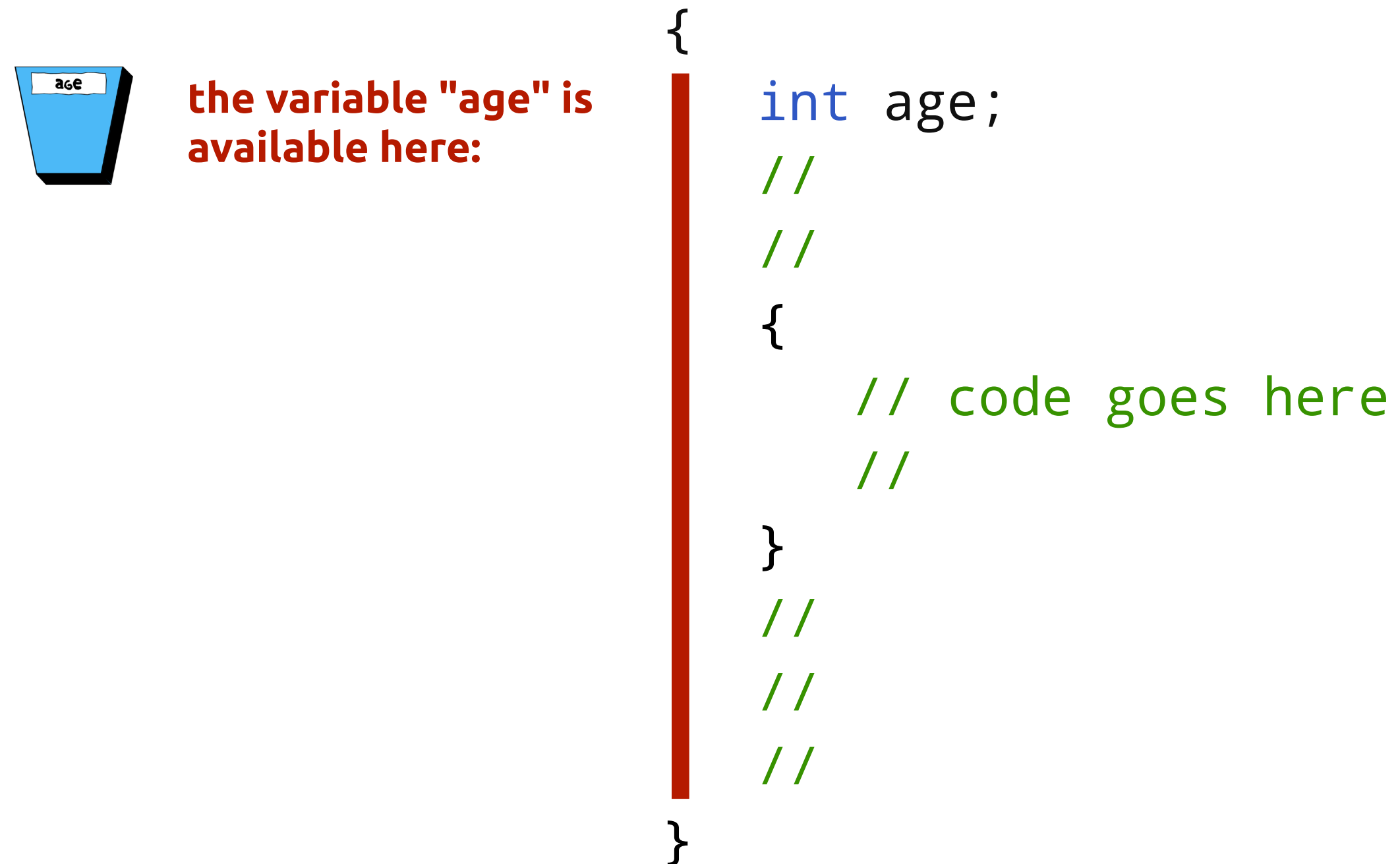
scope

```
{  
    // code goes here  
    //  
    //  
    {  
        // code goes here  
        //  
    }  
    //  
    //  
    //  
}
```

scopes can be nested


```
{  
    int age;  
    //  
    //  
    {  
        // code goes here  
        //  
    }  
    //  
    //  
    //  
}
```

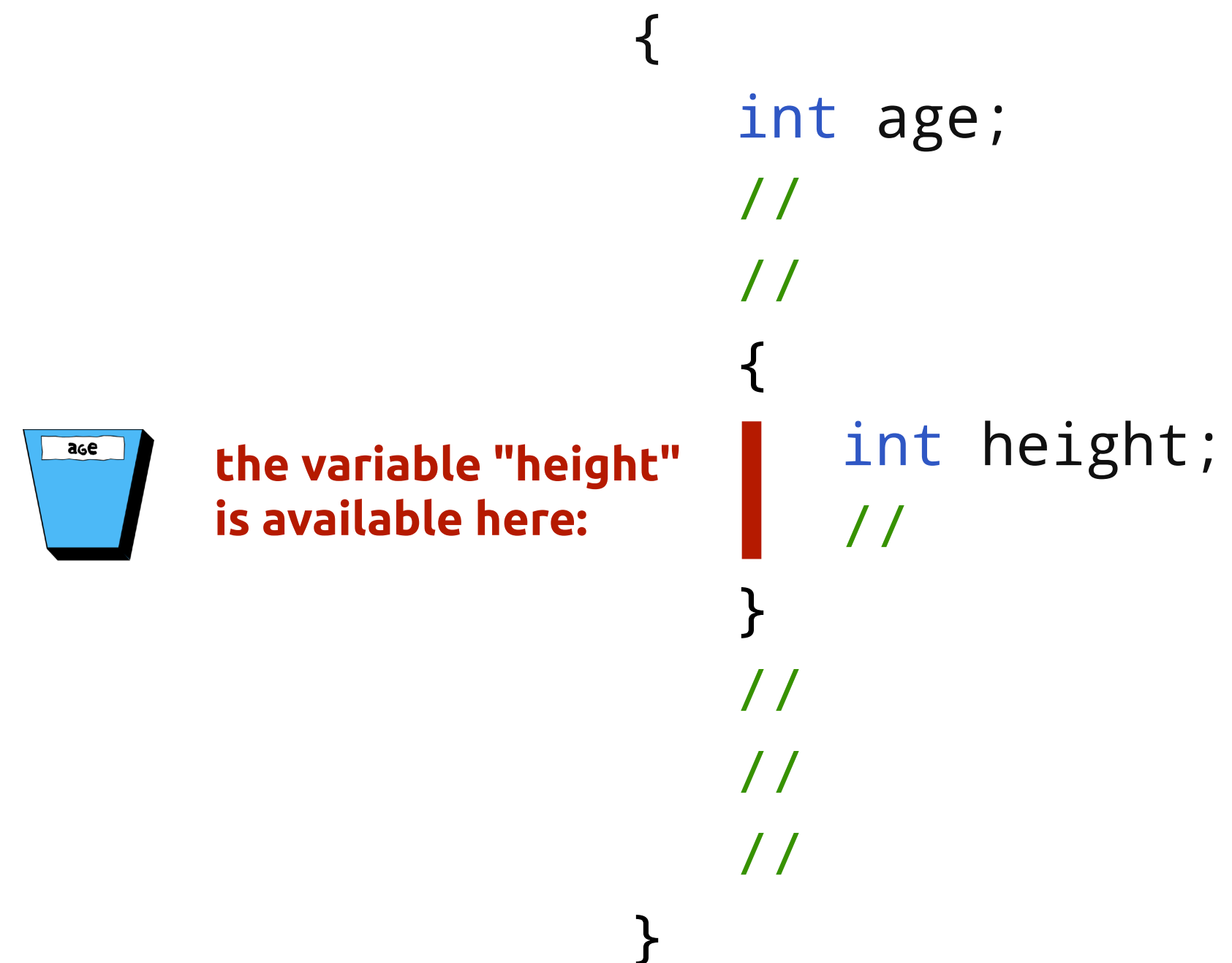
variable in outer scope



variable in outer scope

```
{  
    int age;  
    //  
    //  
    {  
        int height;  
        //  
    }  
    //  
    //  
    //  
}
```

variable in inner scope



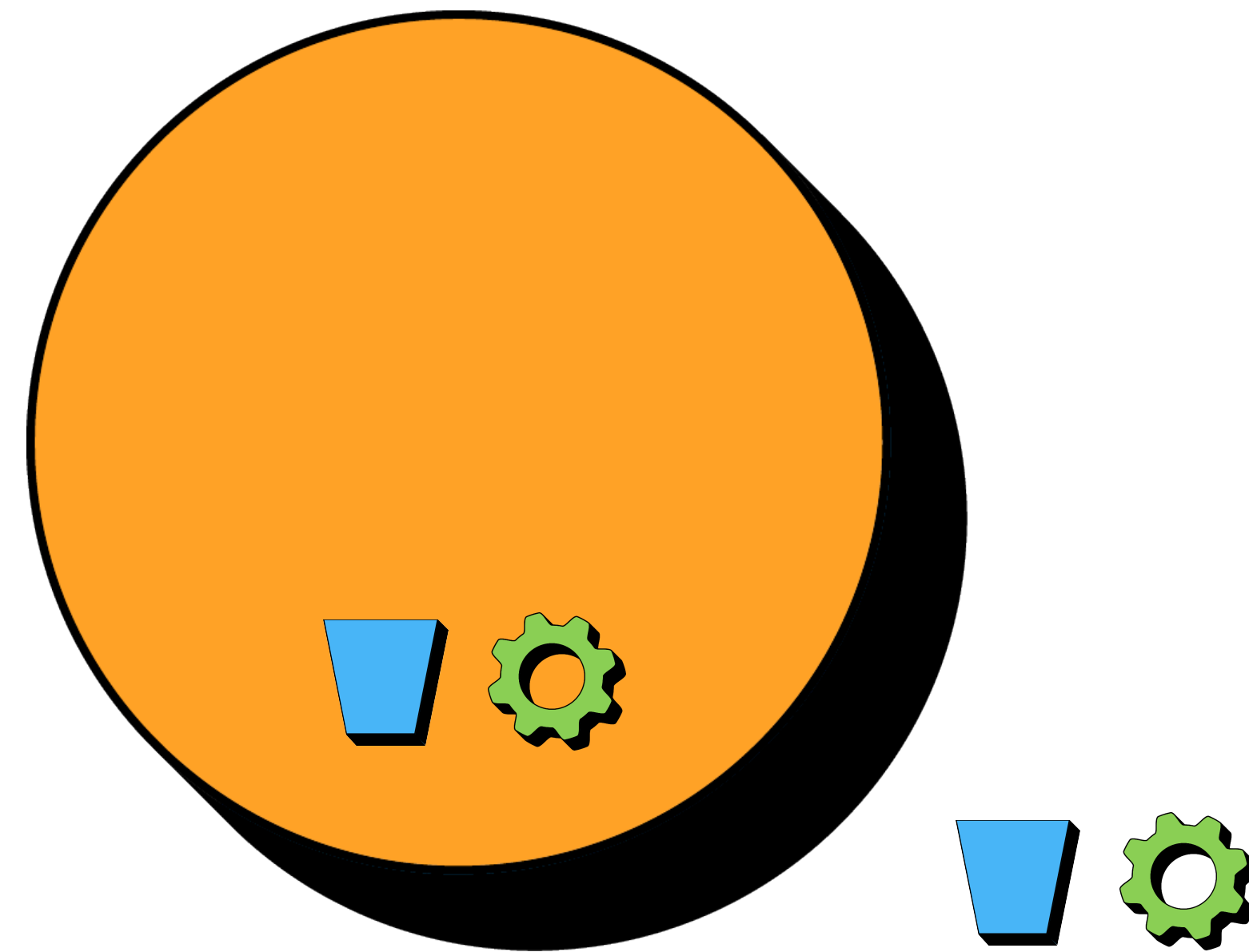
the variable "height"
is available here:

```
{  
    int age;  
    //  
    //  
    {  
        int height;  
        //  
    }  
    //  
    //  
    //  
}
```

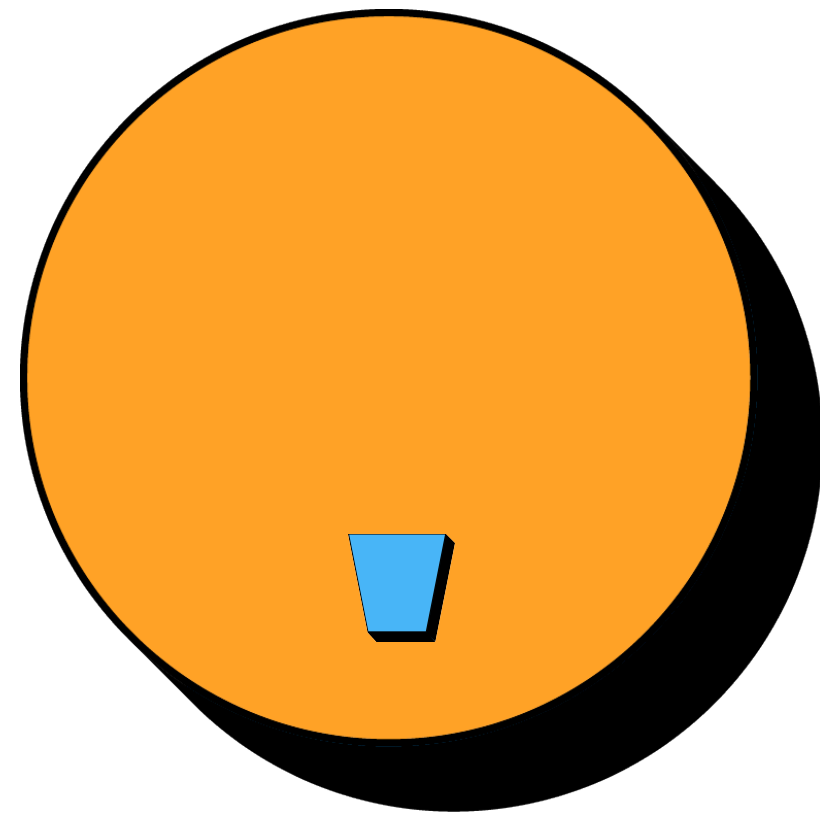
variable in inner scope

```
{  
    int age;  
    //  
    //  
    {  
        int height;  
        //  
    }  
    //  
    //  
    //  
}
```

conclusion: variables live inside scopes

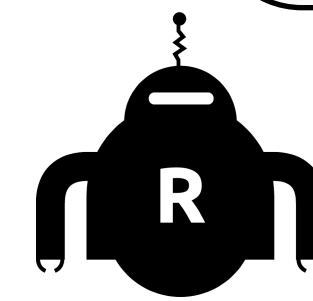


private and public

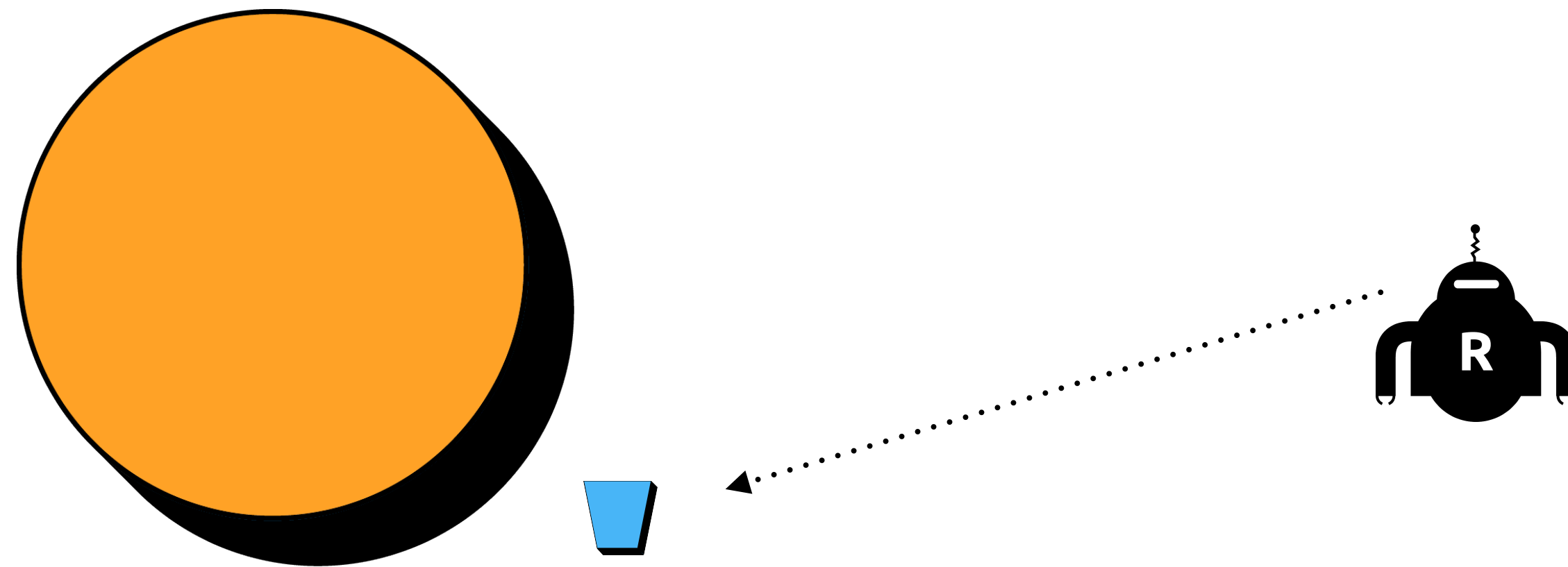


```
int age = 30;
```

private variable



cannot access
the variable
'age'



```
public int age = 30;
```

public variable

if

conditional statements

```
if( true ){  
    // then execute code here  
}
```

the if-statement

```
if( false ){  
    // do not execute code here  
}
```

the if-statement

```
if( false ){  
    // do not execute code here  
} else {  
    // instead, execute code here  
}
```

the else-statement

```
if( true ){  
    // code here will be executed  
} else {  
    // code here will not be executed  
}
```

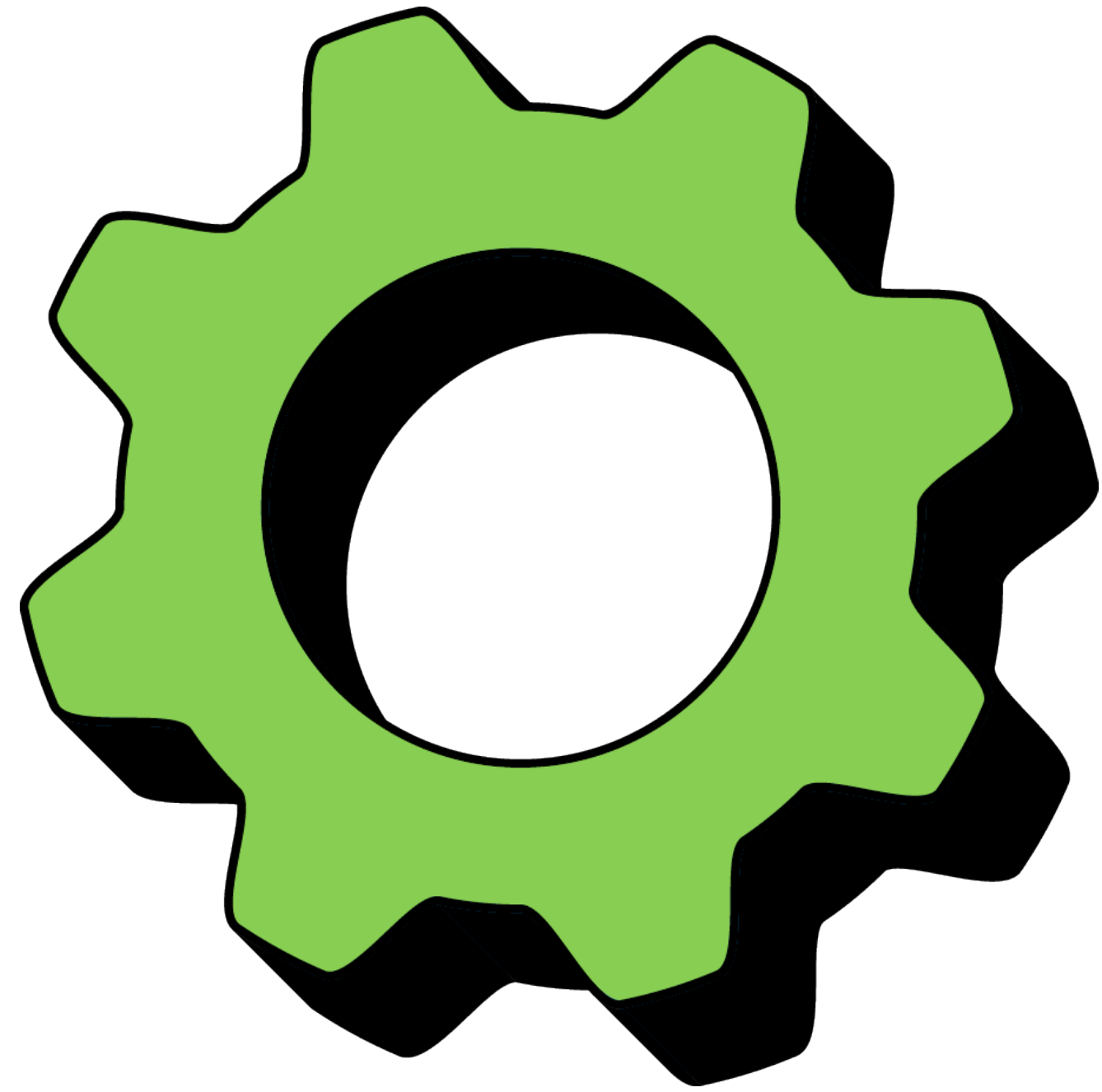
the else-statement

```
bool isSunny = true;  
bool isRainy = true;  
  
if( isSunny && isRainy ){  
    // show rainbow!  
}
```

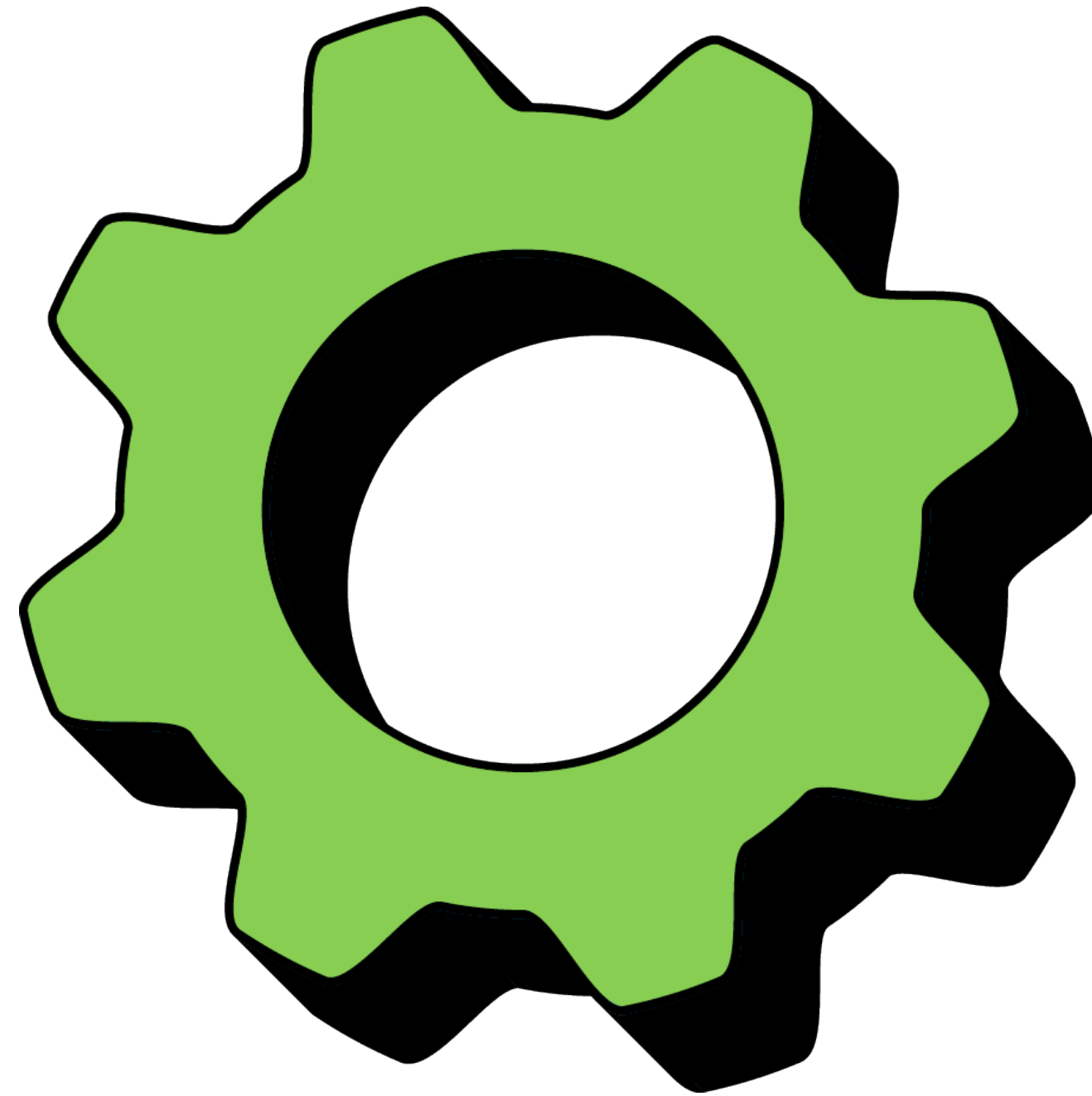
the AND-statement

```
bool isAlarmDead = false;  
bool isBusLate = true;  
  
if( isAlarmDead || isBusLate ){  
    // will be late for work  
}
```


the OR-statement



a function does a job




defining a function



```
void SayHi(){  
    Debug.Log("hi!");  
}
```

defining a function

return type
name
arguments
begin




```
void SayHi(){  
    Debug.Log("hi!");  
}
```

end

defining a function


function names
always begins with a
capital letter

return type name arguments begin




```
void SayHi(){  
    Debug.Log("hi!");  
}  
end
```

defining a function



```
void Start(){  
    SayHi();  
}
```

calling a function

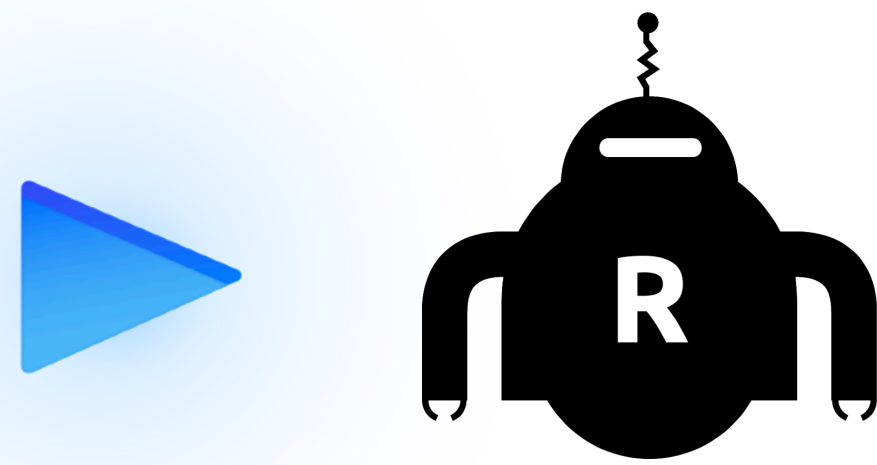


```
void Start(){  
    SayHi();  
}
```

**function
name**

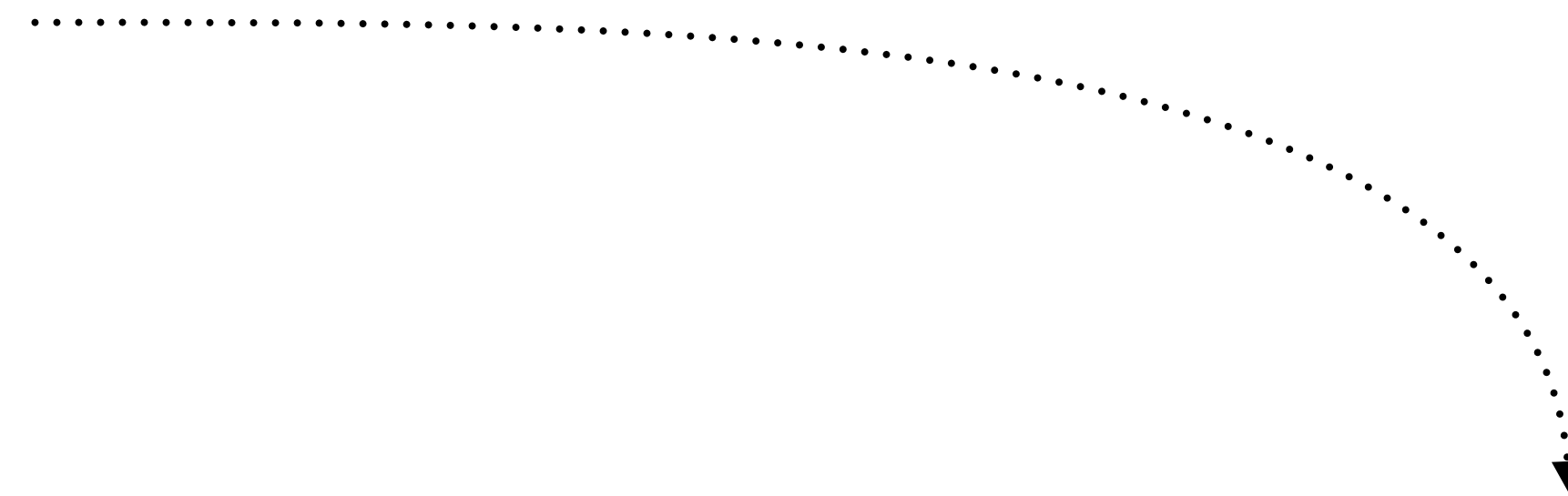
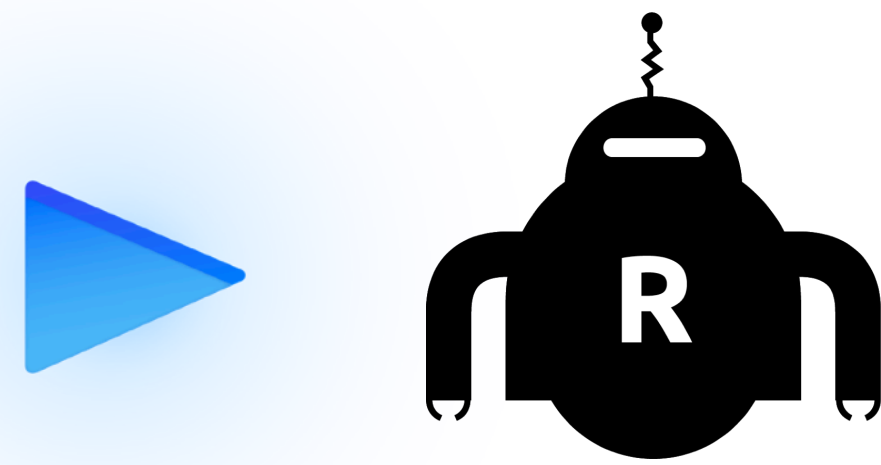
call it!

calling a function



```
void Start(){  
    SayHi();  
}  
  
void SayHi(){  
    Debug.Log("hi!");  
}
```

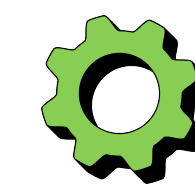
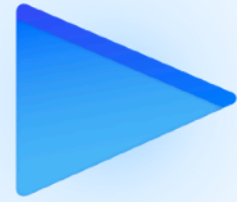
calling a function



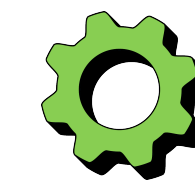
unity calls the
Start function
on the first
frame

```
void Start(){  
    SayHi();  
}  
  
void SayHi(){  
    Debug.Log("hi!");  
}
```

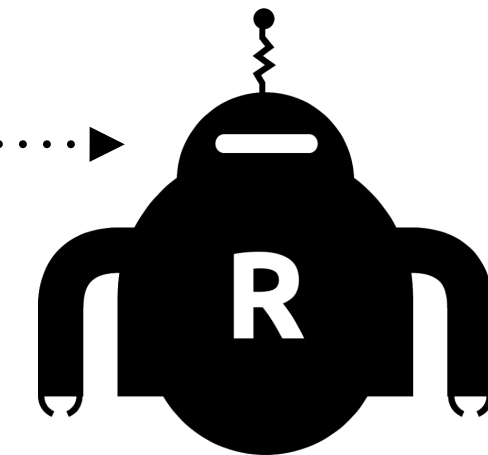
calling a function



```
void Start(){  
    SayHi();  
}
```

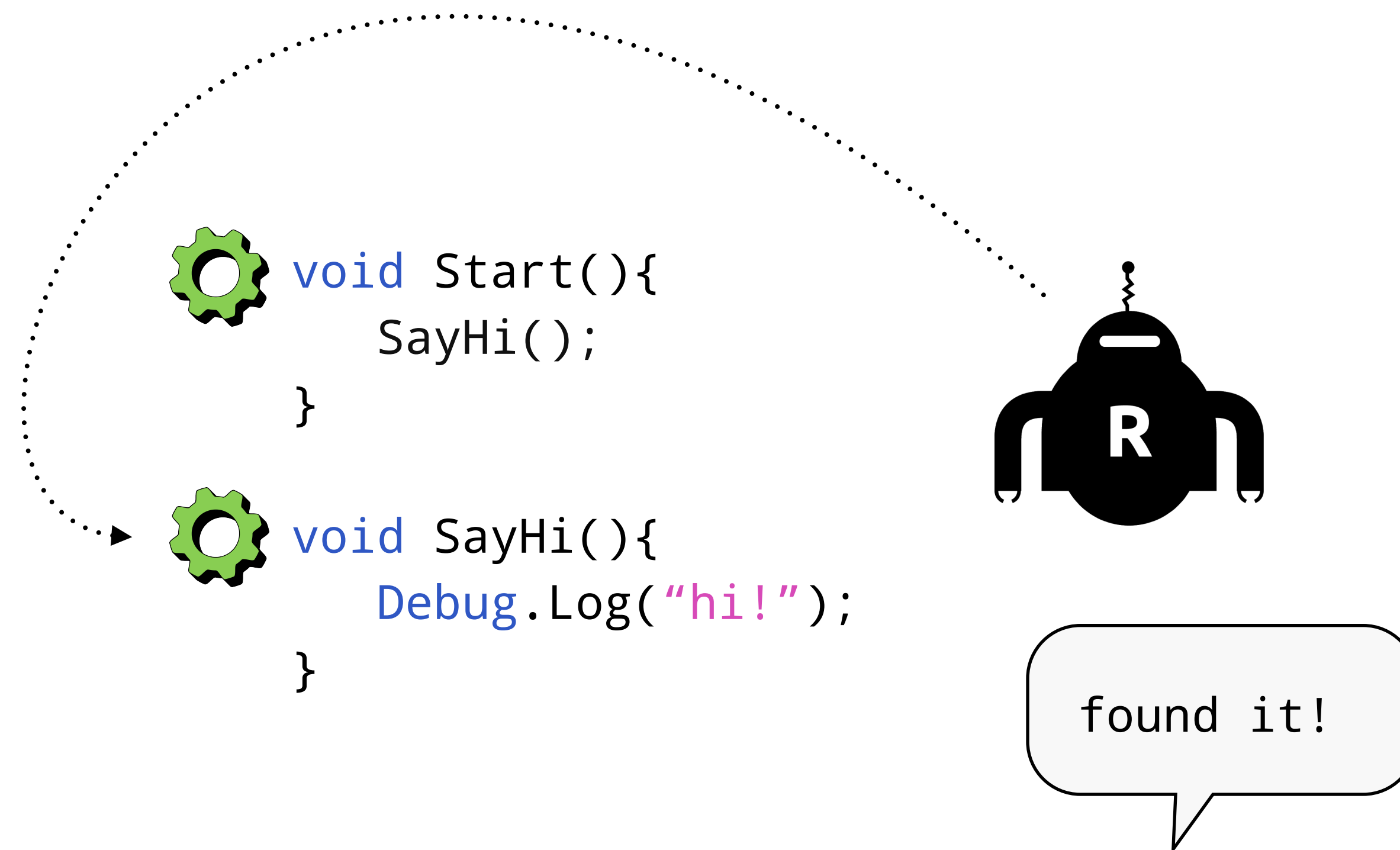


```
void SayHi(){  
    Debug.Log("hi!");  
}
```




ok; i'll find
the function
named 'SayHi'
and continue
there


calling a function



calling a function

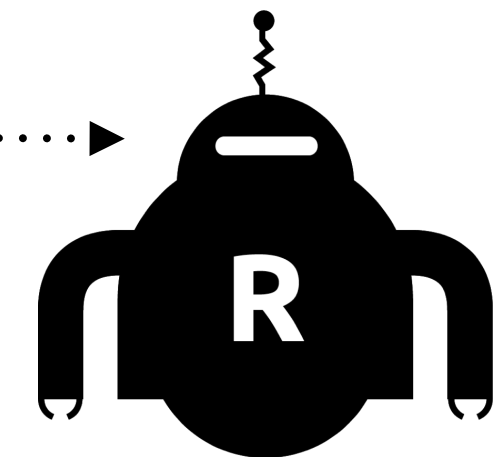


```
void Start(){  
    SayHi();  
}
```

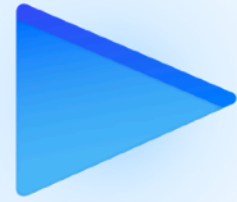


```
void SayHi(){  
    Debug.Log("hi!");  
}
```

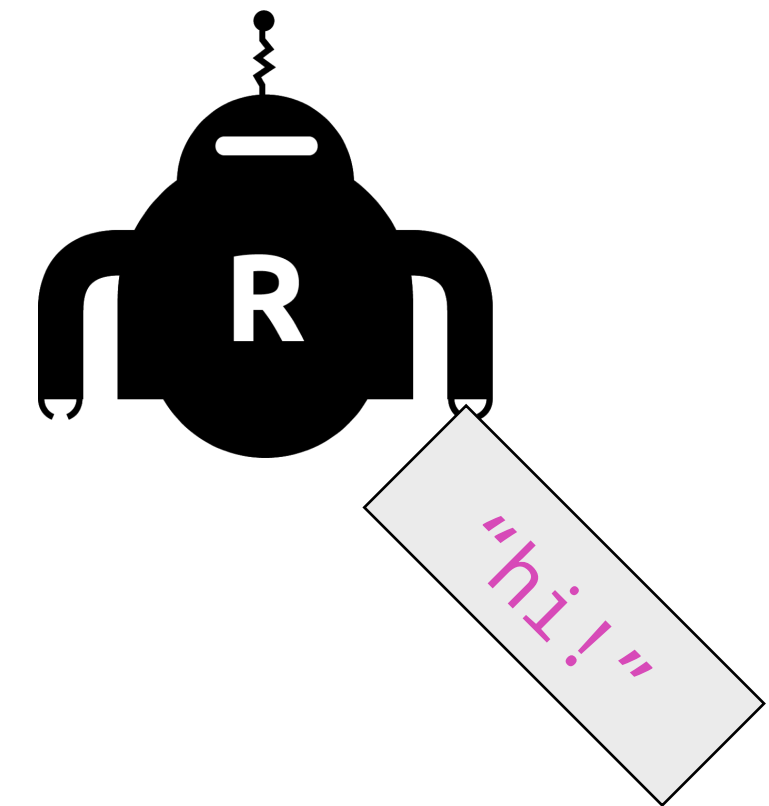
ok; i'll find the
function named
'Log' in the
class Debug and
carry the value
'hi!' to it



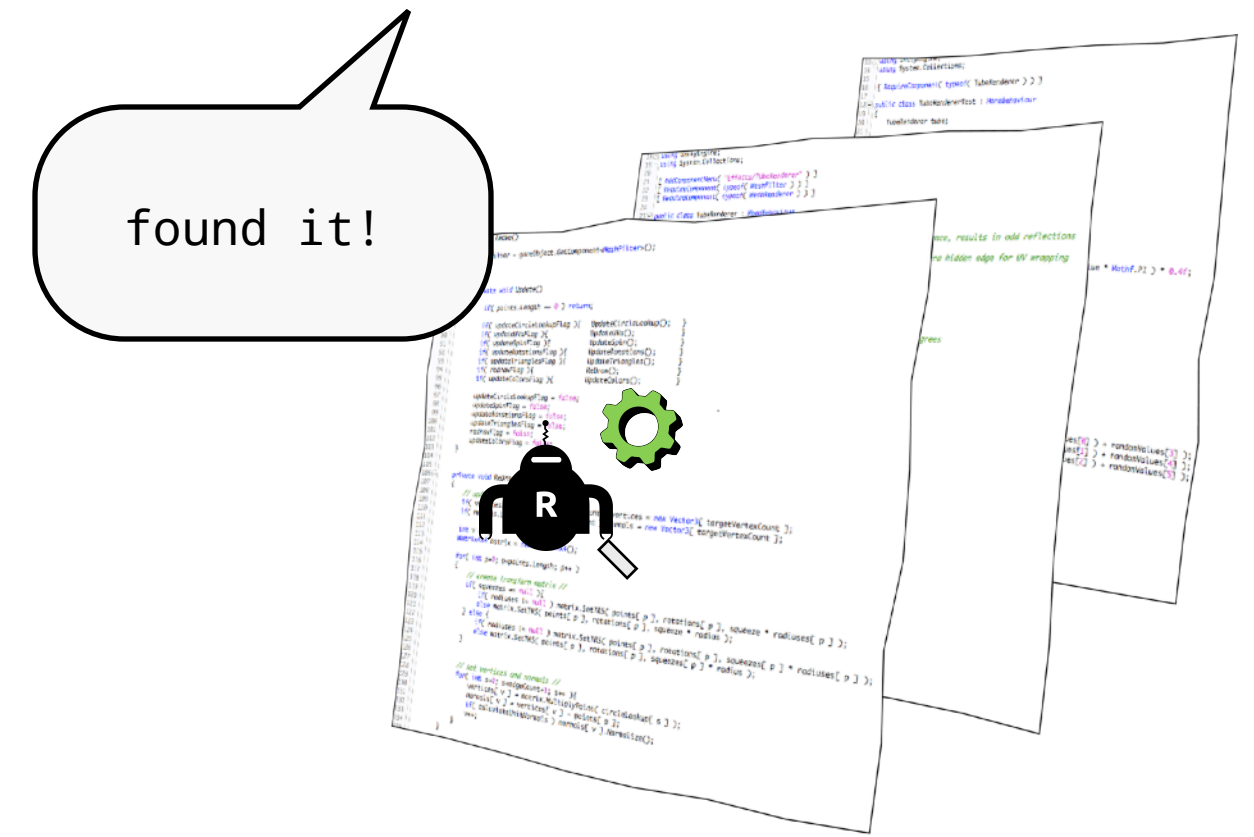
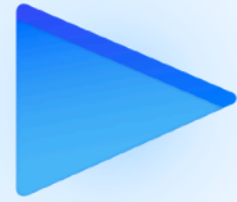
calling a function



```
void Start(){  
    SayHi();  
}  
  
void SayHi(){  
    Debug.Log("hi!");  
}
```



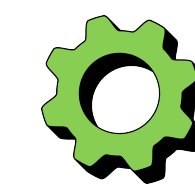
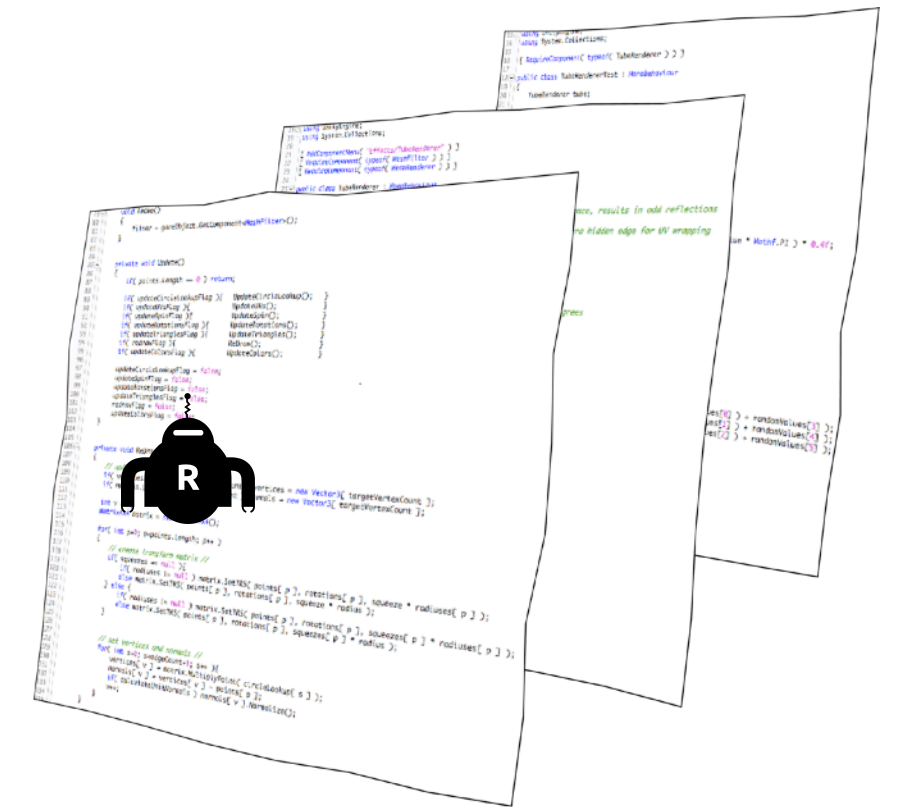
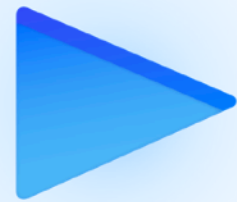
calling a function



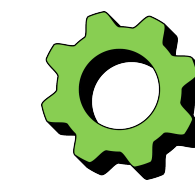
```
void Start(){
    SayHi();
}

void SayHi(){
    Debug.Log("hi!");
}
```

calling a function

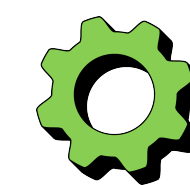
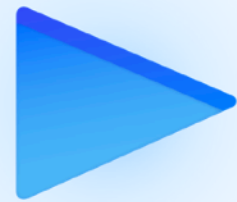


```
void Start(){  
    SayHi();  
}
```

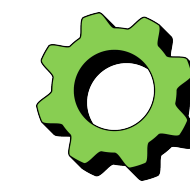


```
void SayHi(){  
    Debug.Log("hi!");  
}
```

calling a function

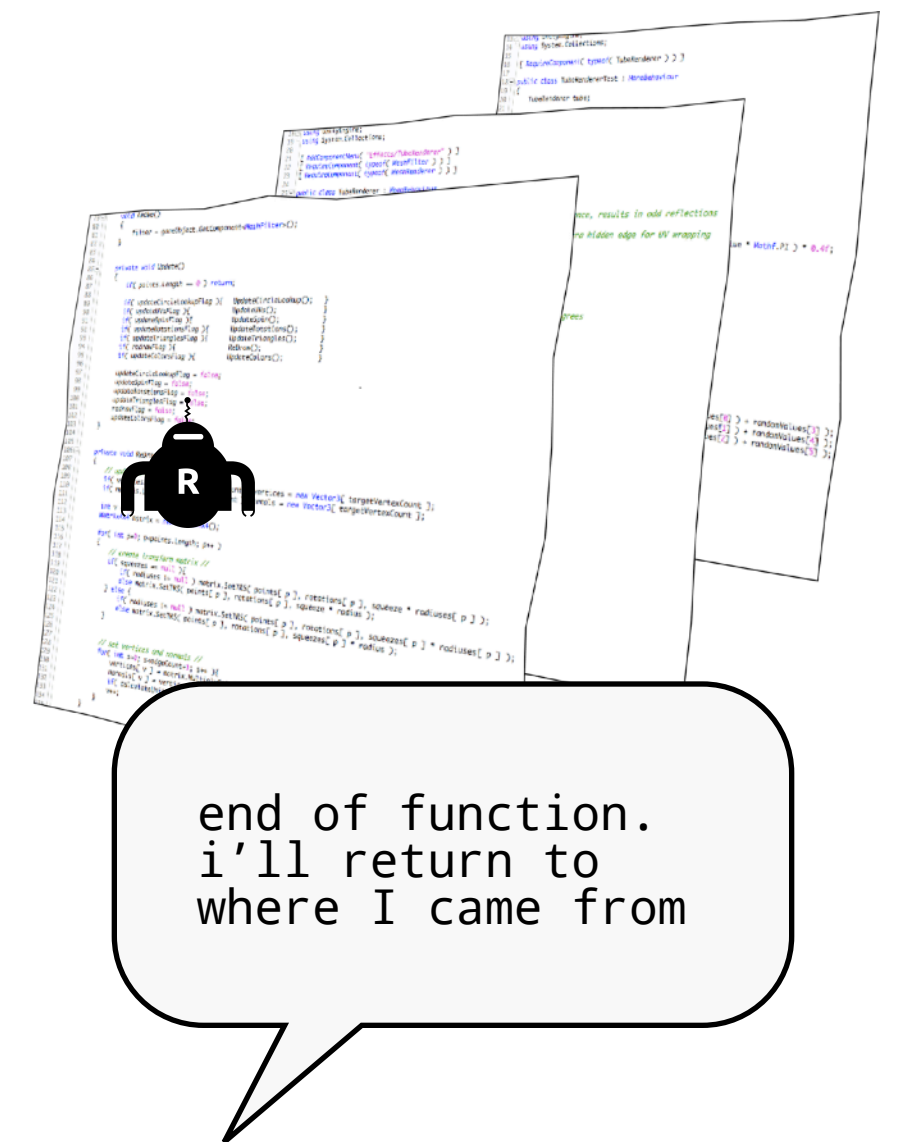


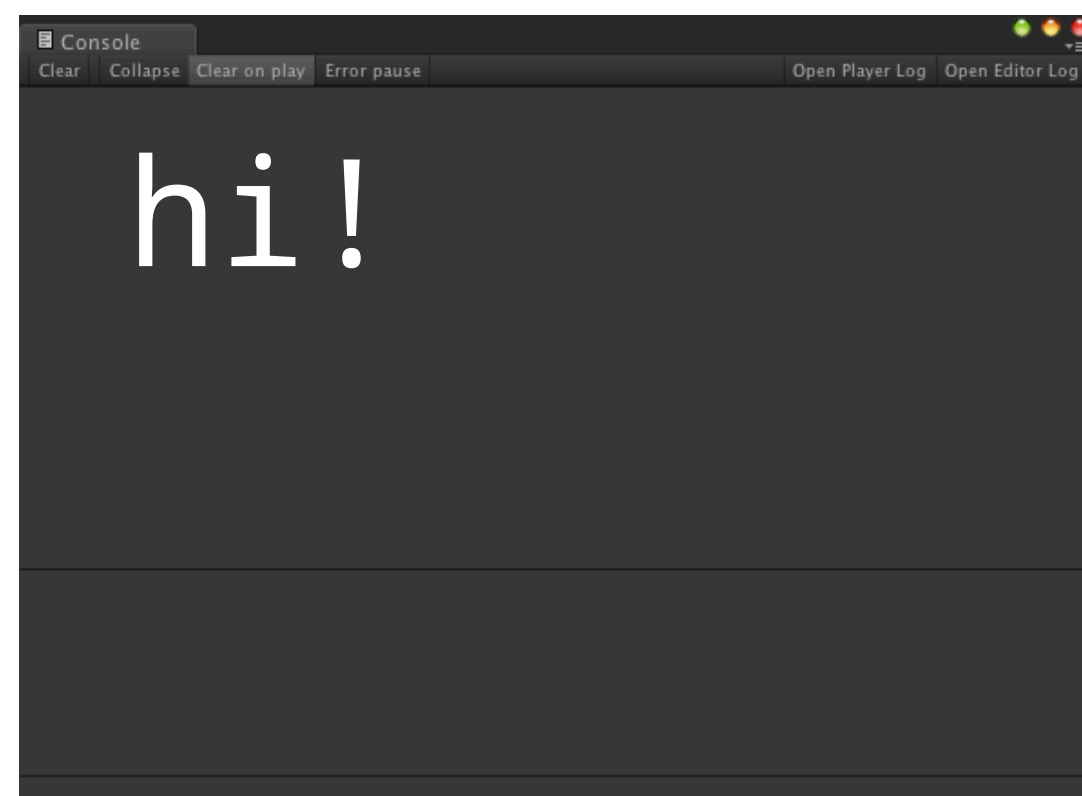
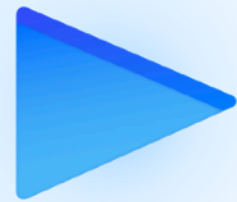
```
void Start(){  
    SayHi();  
}
```





```
void SayHi(){  
    Debug.Log("hi!");  
}
```

calling a function

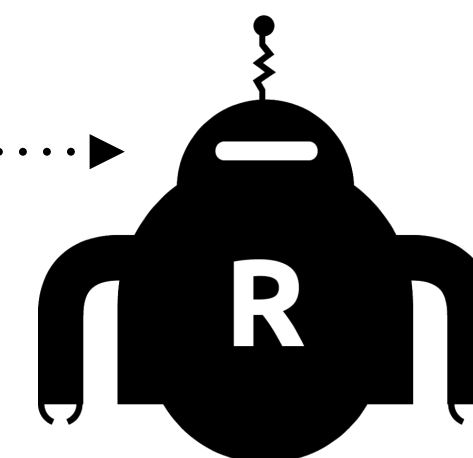




 `void Start(){
 SayHi();
}`

 `void SayHi(){
 Debug.Log("hi!");
}`

end of function.
i'll return to
where I came from.

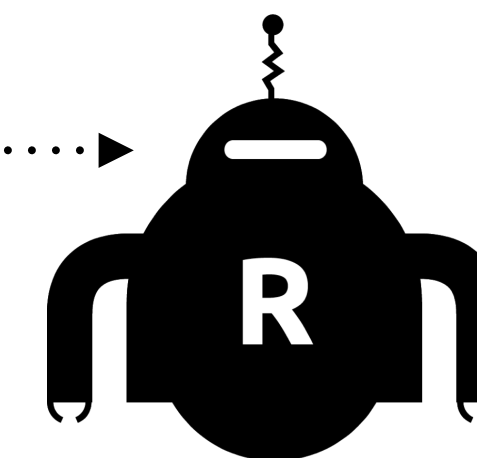


calling a function



```
void Start(){  
    SayHi();  
}  
  
void SayHi(){  
    Debug.Log("hi!");  
}
```

i'm done here

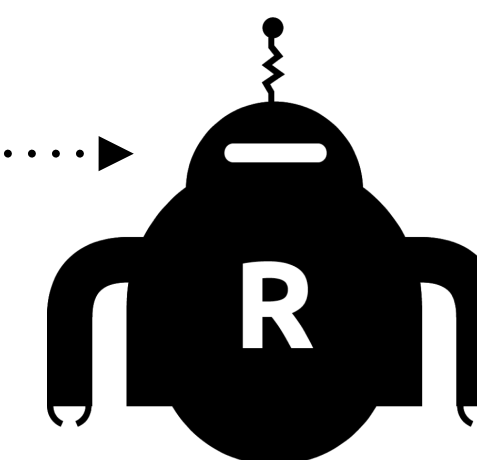


calling a function

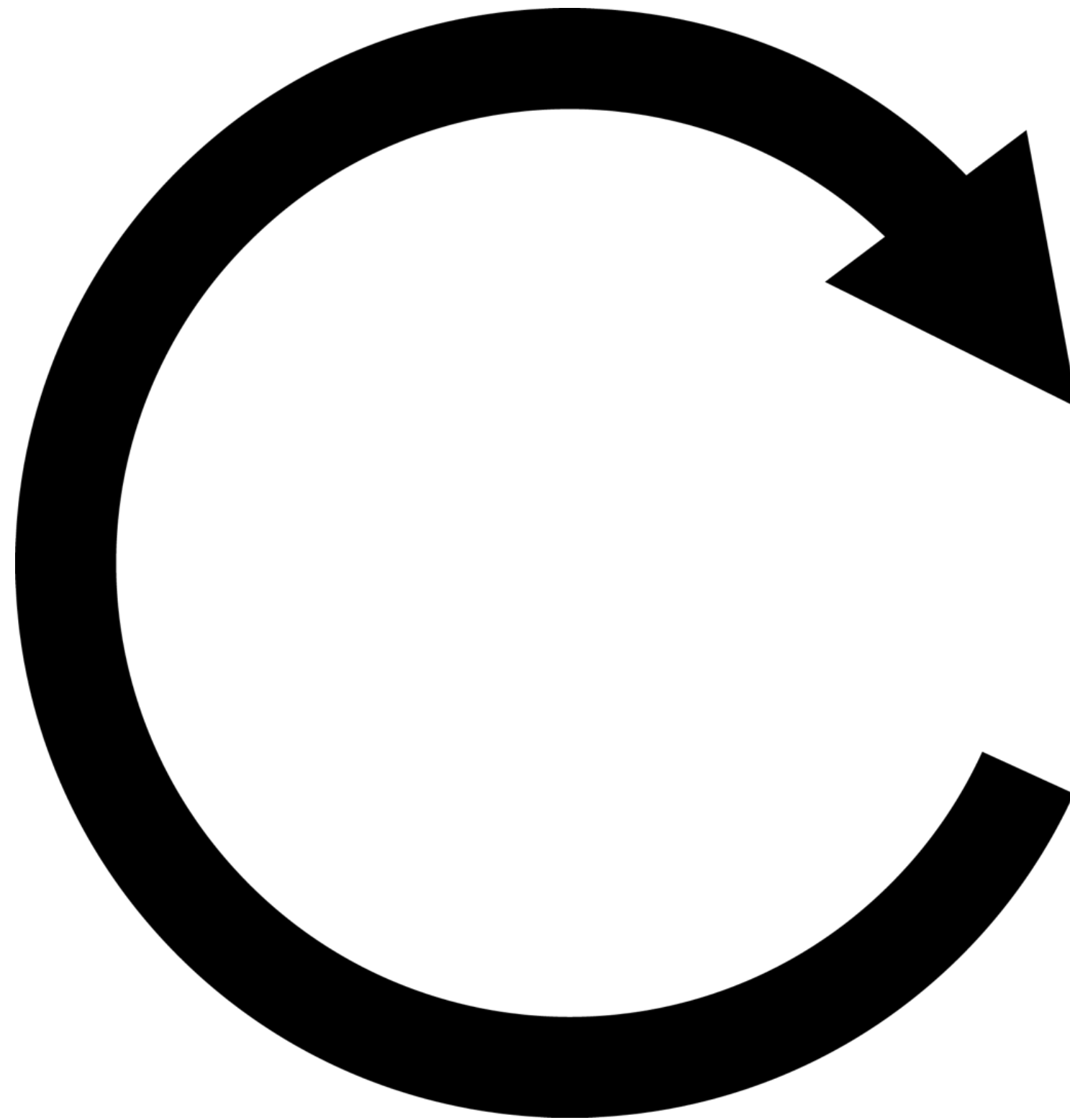


```
void Start(){  
    SayHi();  
}  
  
void SayHi(){  
    Debug.Log("hi!");  
}
```

end of function.
I'll return to
wherever I came
from in the
unity system



calling a function



loops

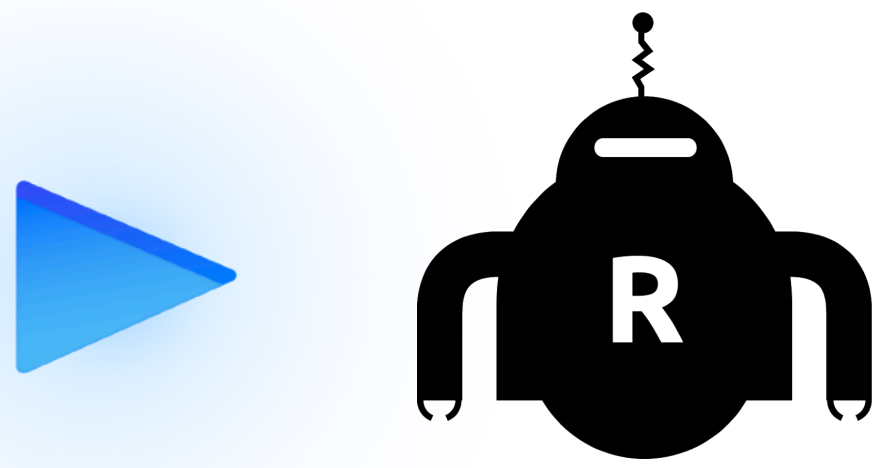
```
for( int i=0; i<2; i++ ){  
    // repeat this code  
}
```

the for-statement

start with the variable `i` at value 0
repeat as long as condition is true
increase `i` by 1 every iteration

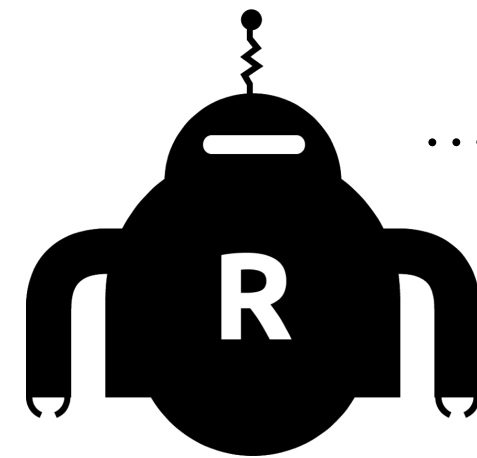
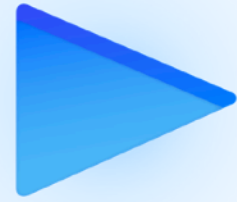
```
for( int i=0; i<2; i++ ){  
    // repeat this code  
}
```

the for-statement



```
for( int i=0; i<2; i++ ){  
    Debug.Log( i );  
}
```

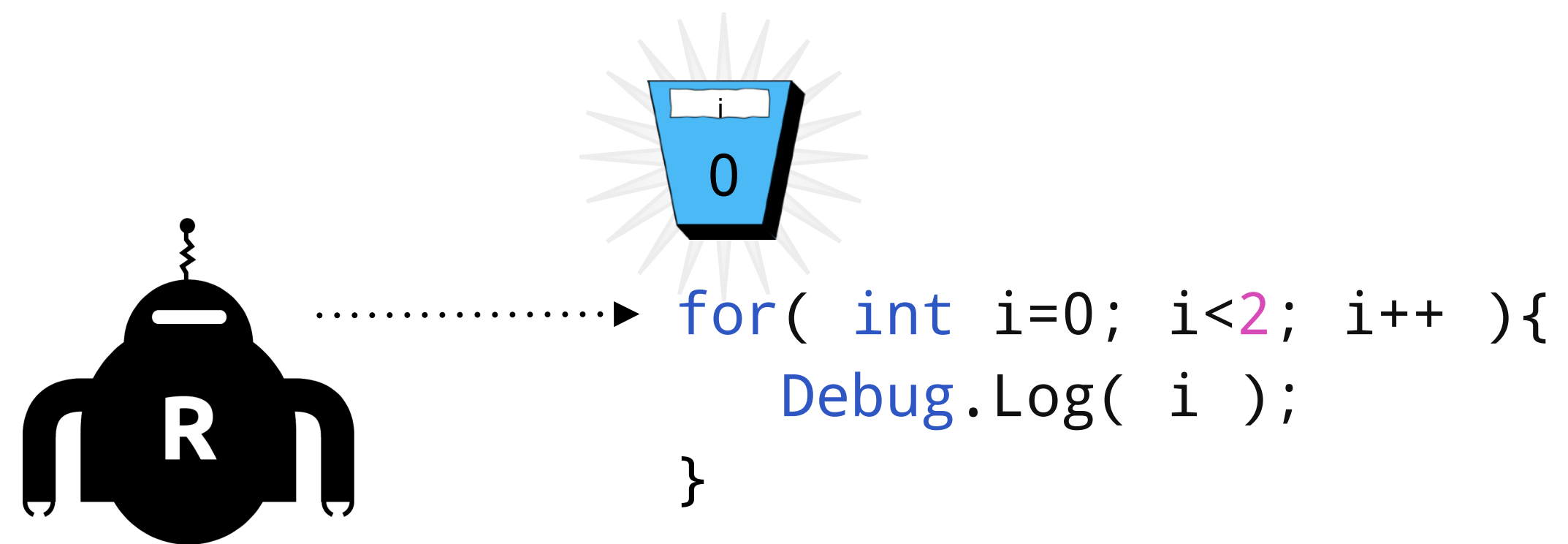
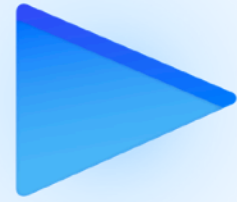
the for-statement



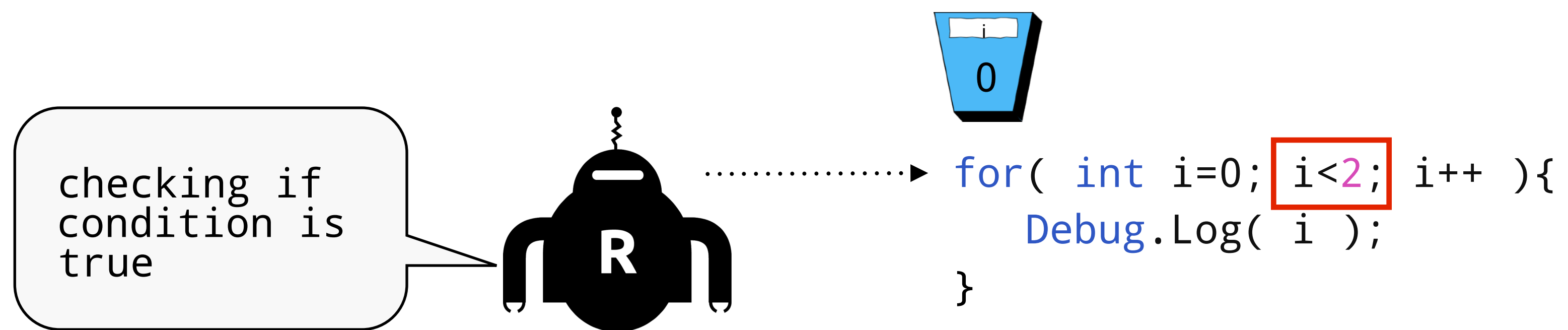
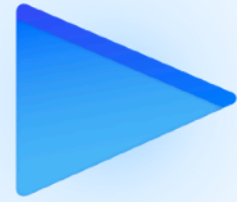
.....▶ `for(int i=0; i<2; i++){
 Debug.Log(i);
}`

creating a local
variable 'i' and
setting it to 0

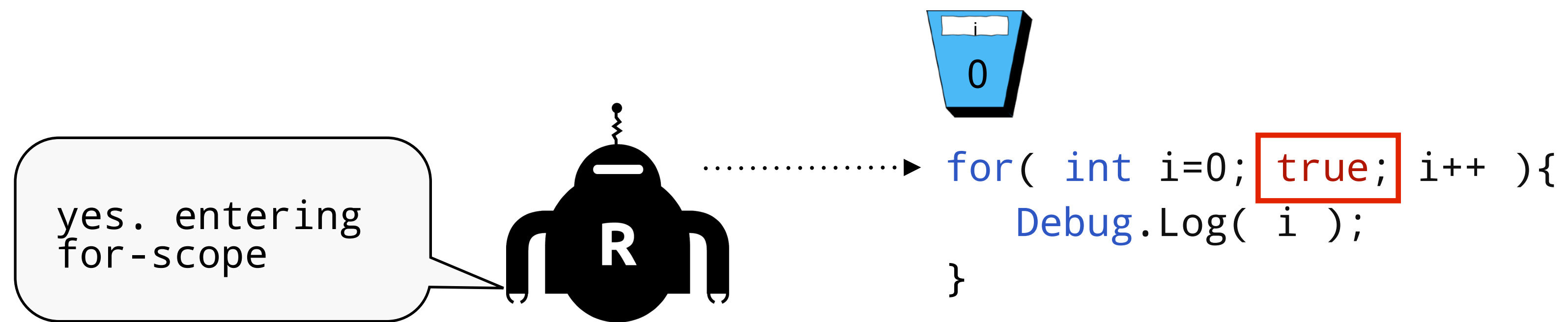
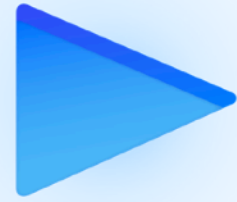
the for-statement



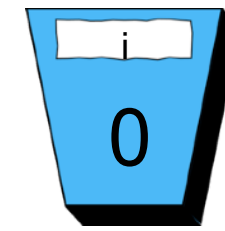
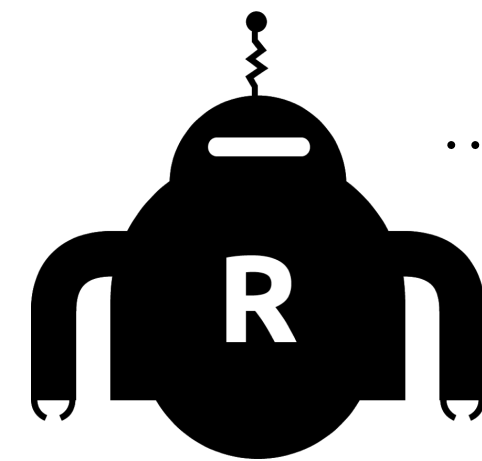
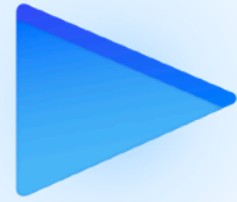
the for-statement



the for-statement



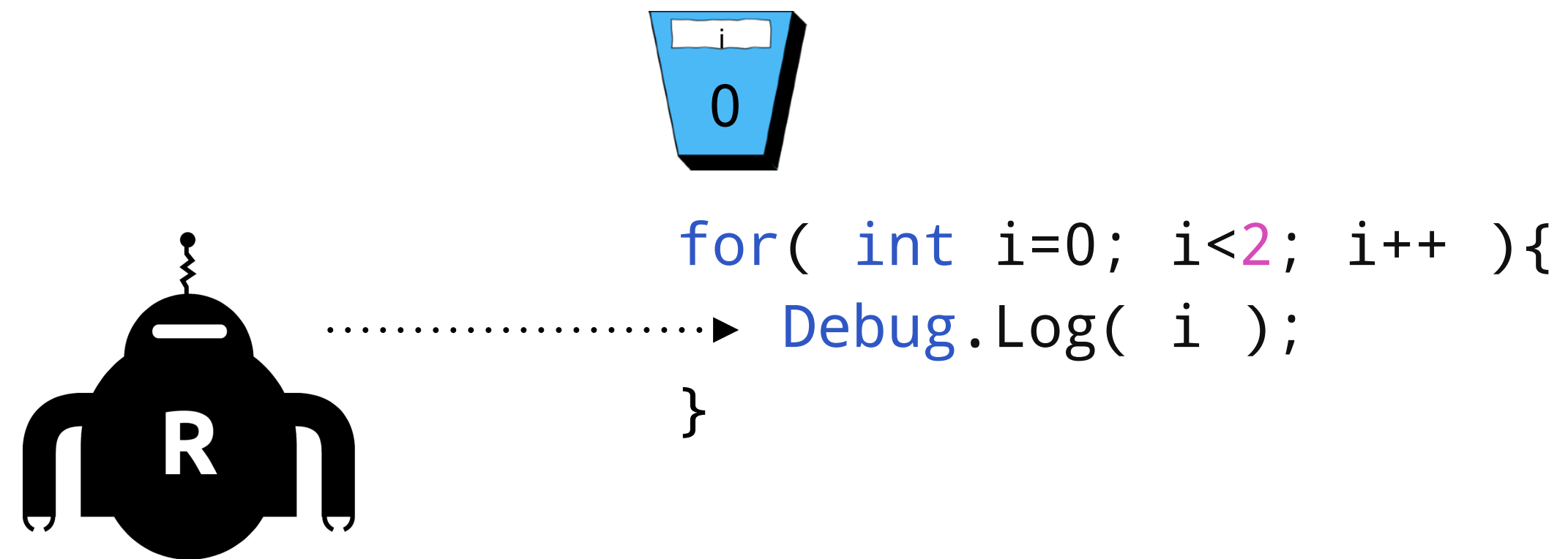
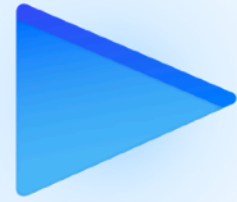
the for-statement



```
for( int i=0; i<2; i++ ){  
    Debug.Log( i );  
}
```

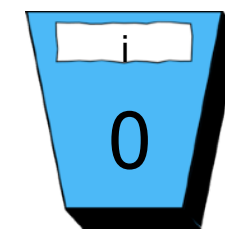
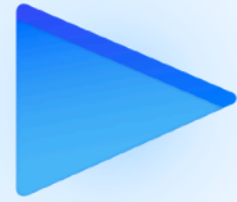
logging the
value of 'i'

the for-statement

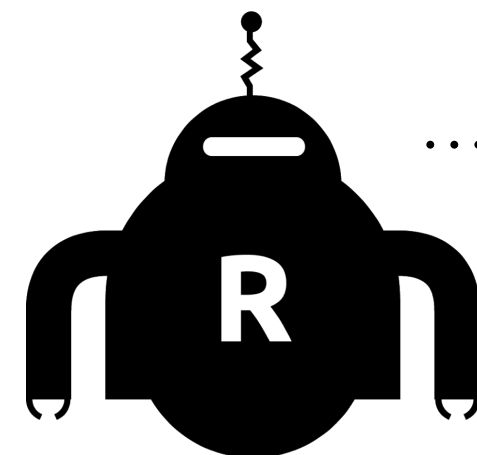


the for-statement



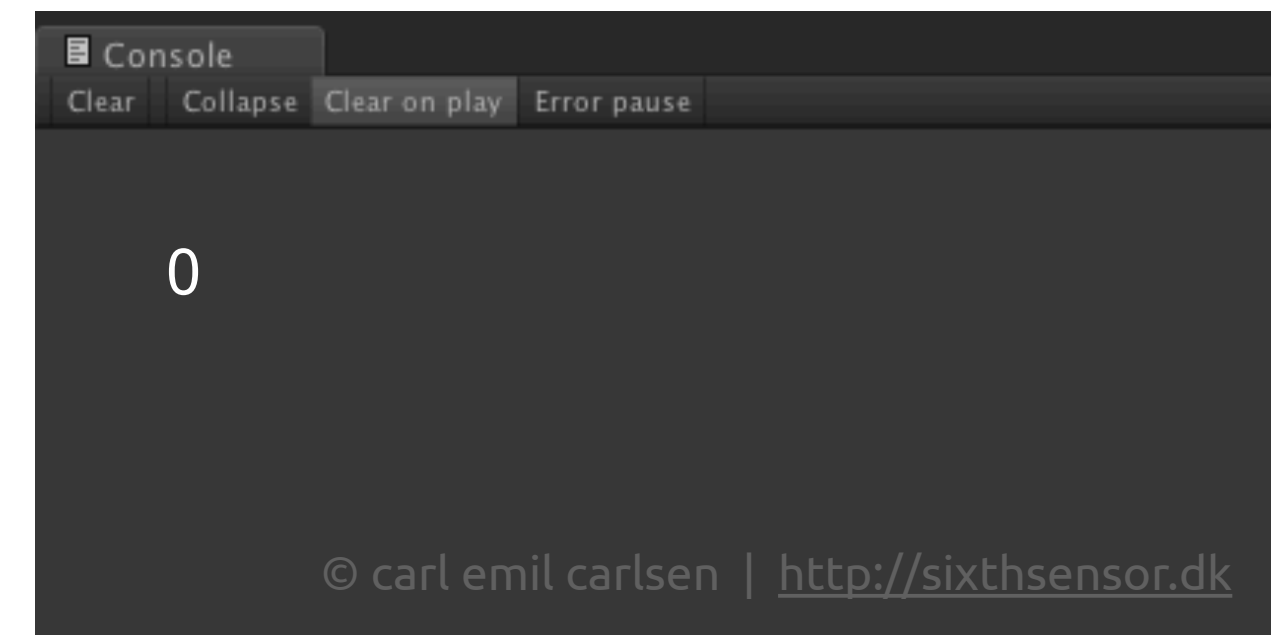


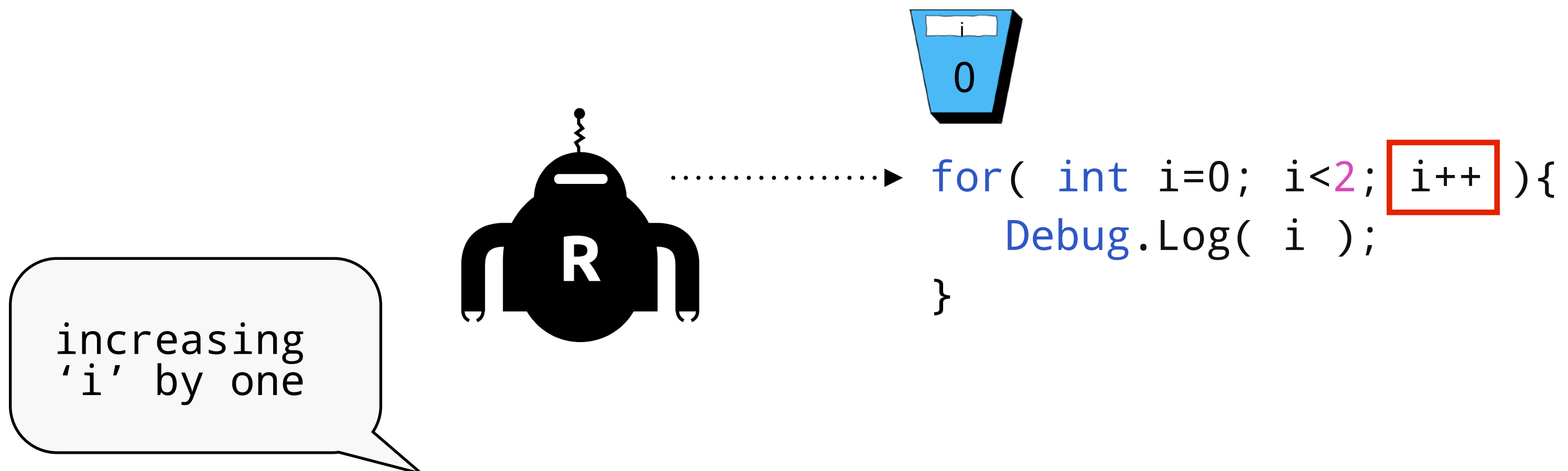
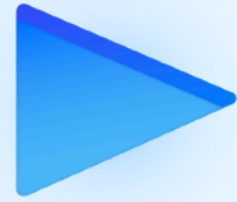
```
for( int i=0; i<2; i++ ){  
    Debug.Log( i );  
}
```



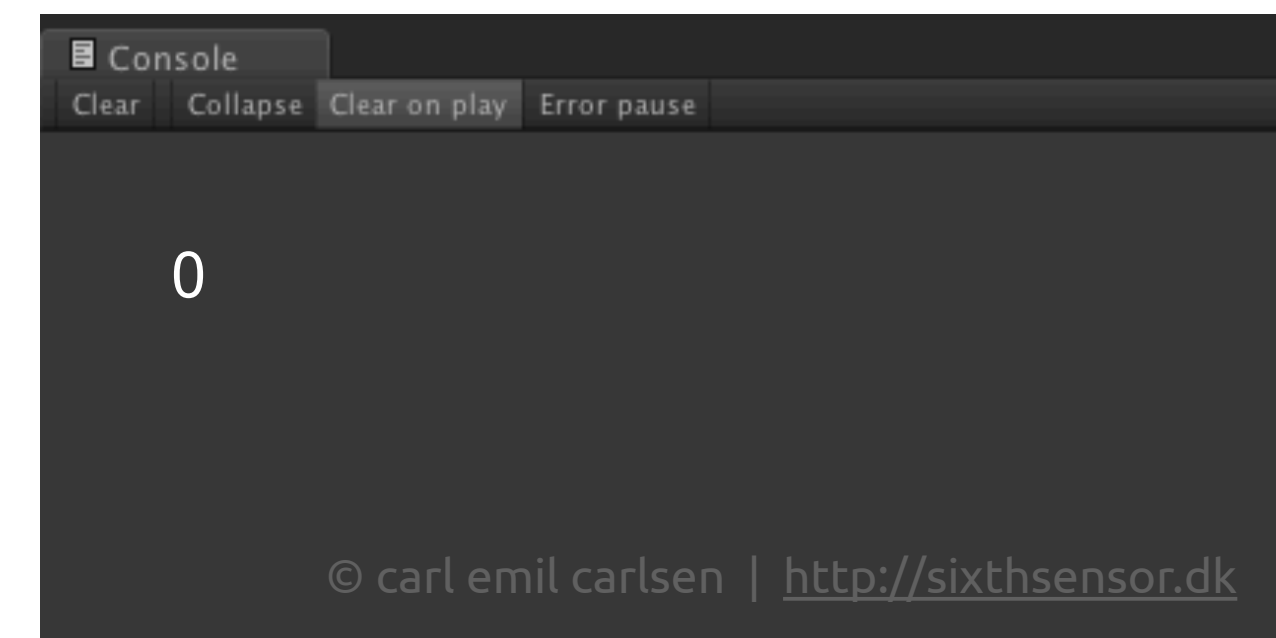
end of for scope!
i'll go back and
check for repeat

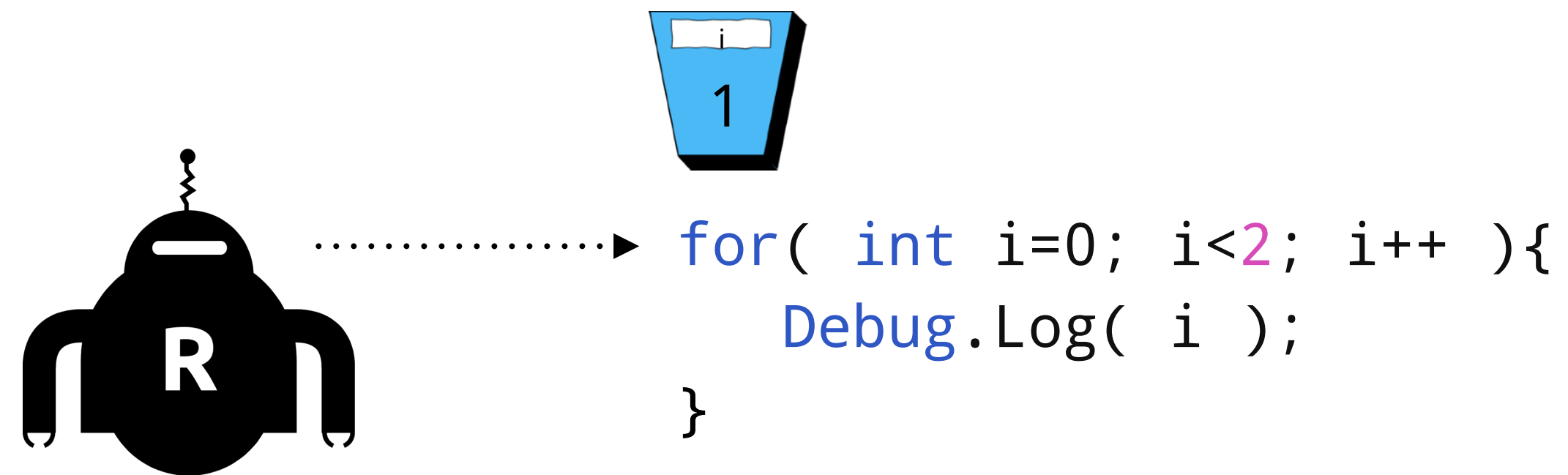
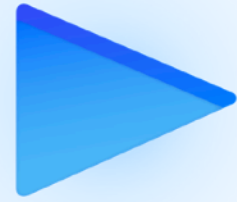
the for-statement



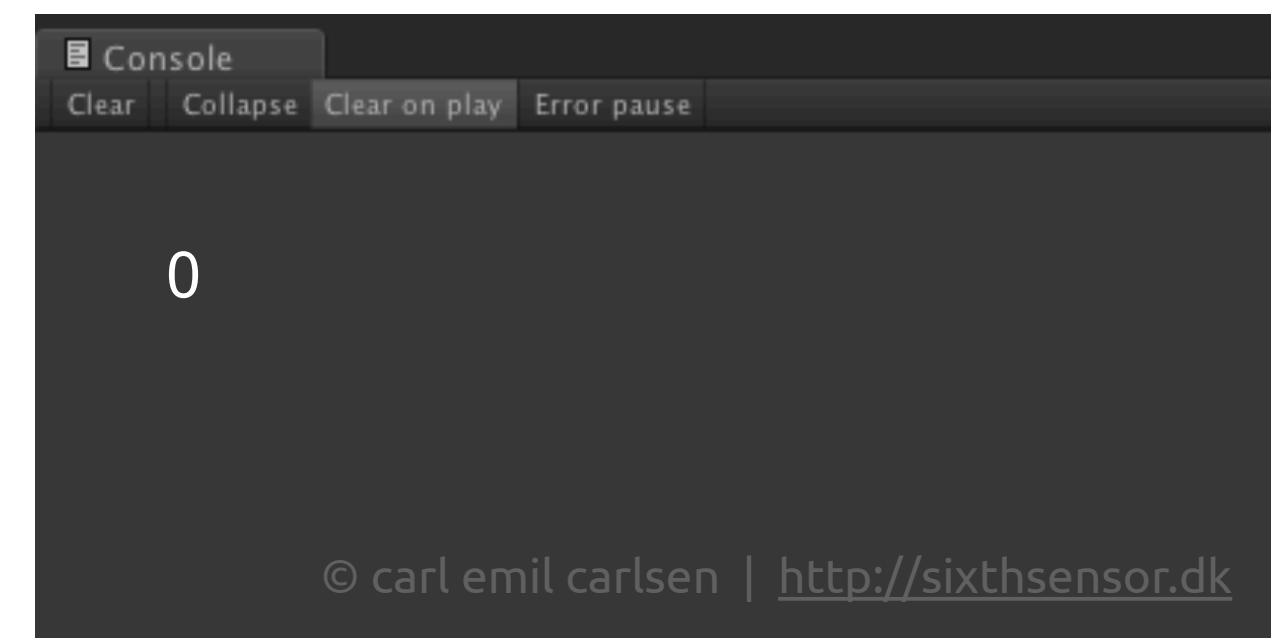


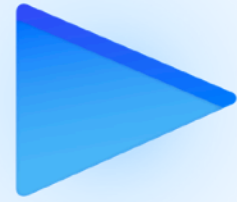
the for-statement



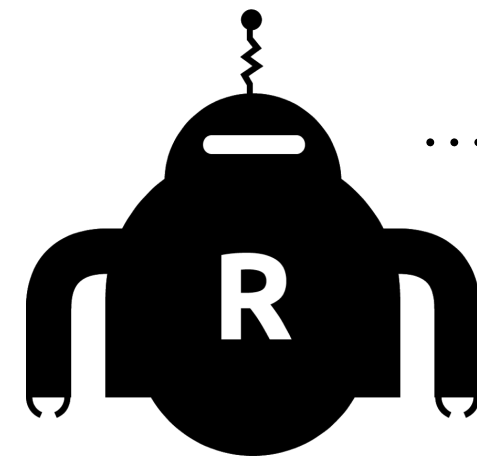


the for-statement



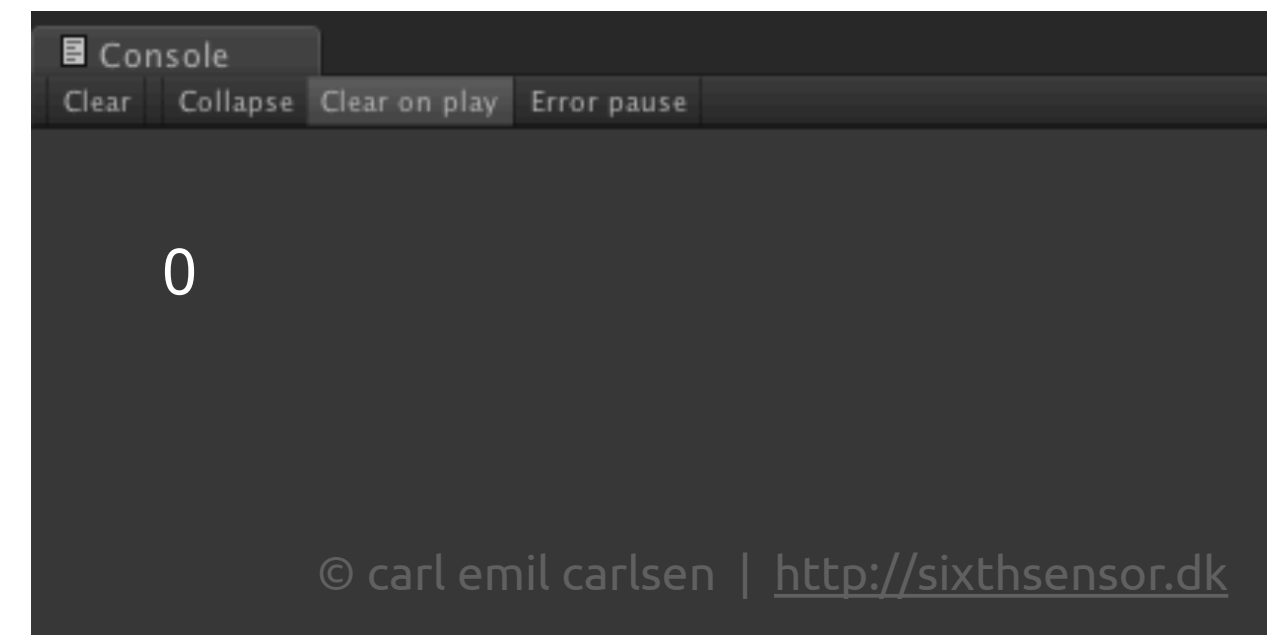


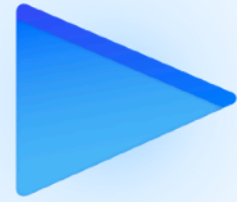
checking if
condition is true



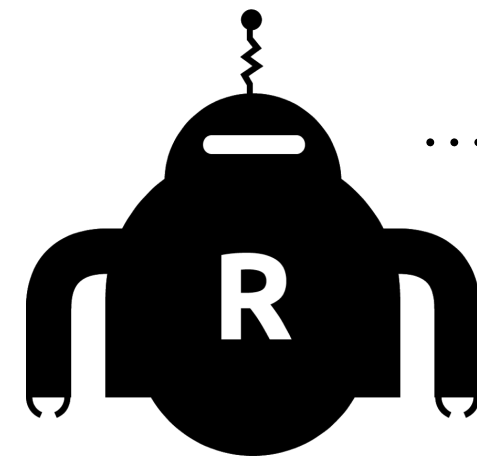
```
for( int i=0; i<2; i++ ){  
    Debug.Log( i );  
}
```

the for-statement



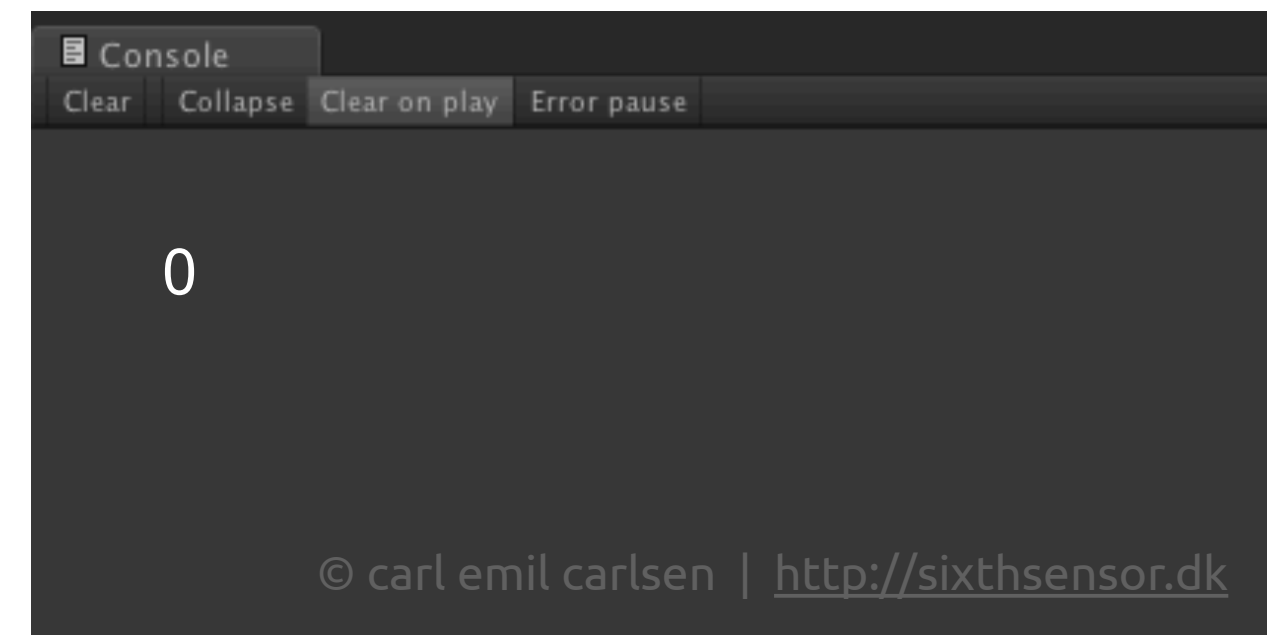


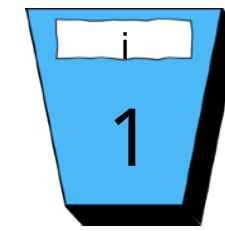
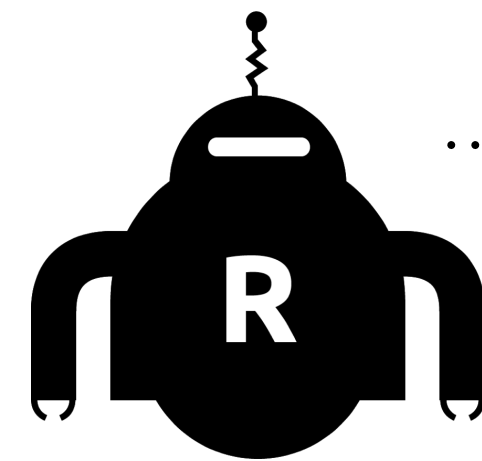
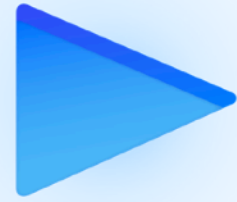
yes.
repeating the
scope



```
for( int i=0; true; i++ ){  
    Debug.Log( i );  
}
```

the for-statement

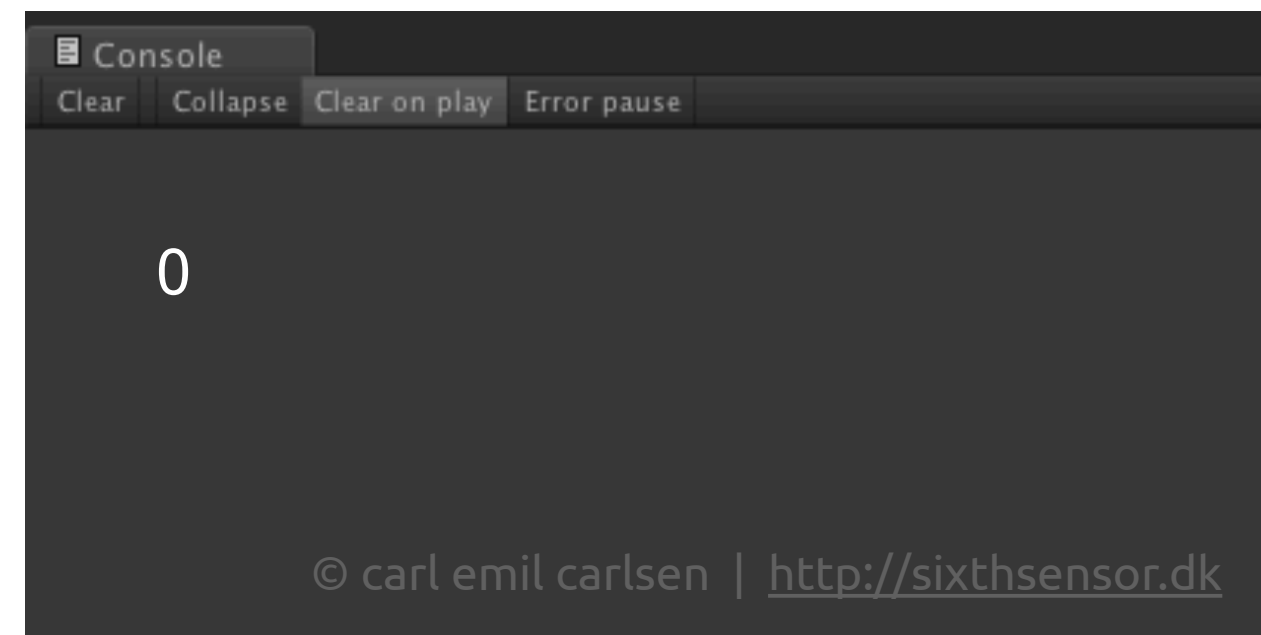


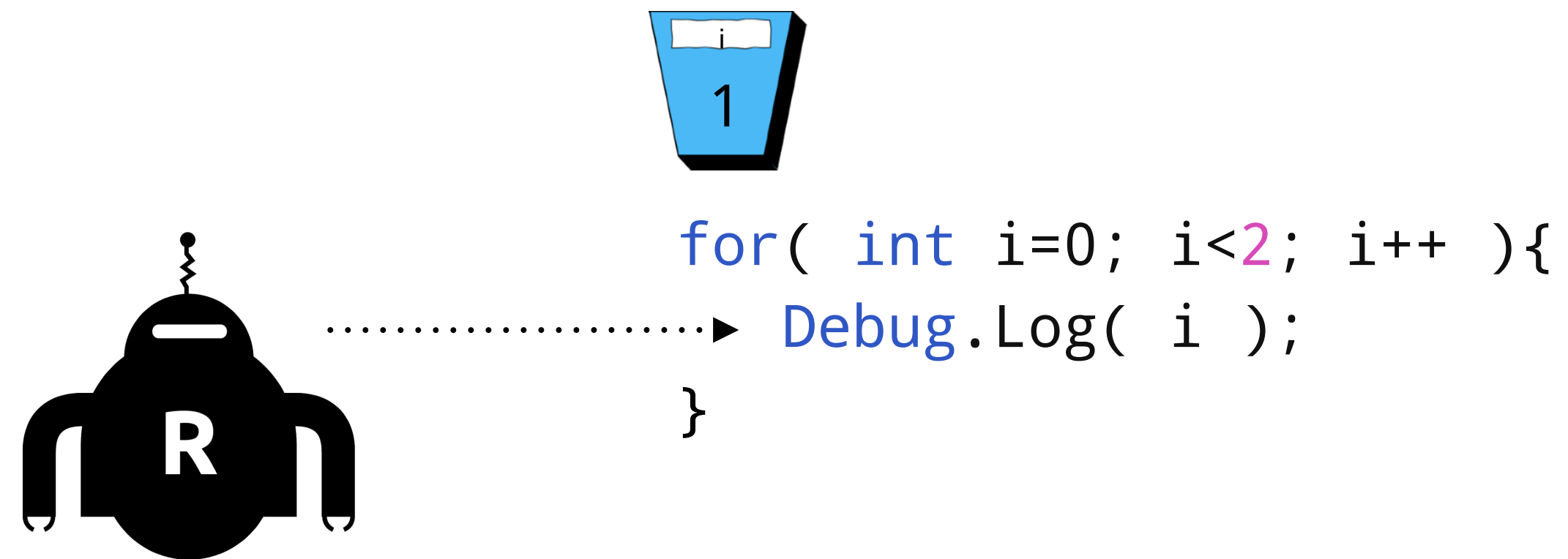
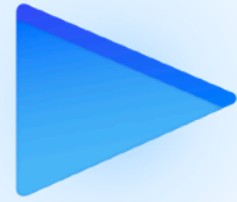


```
for( int i=0; i<2; i++ ){  
    Debug.Log( i );  
}
```

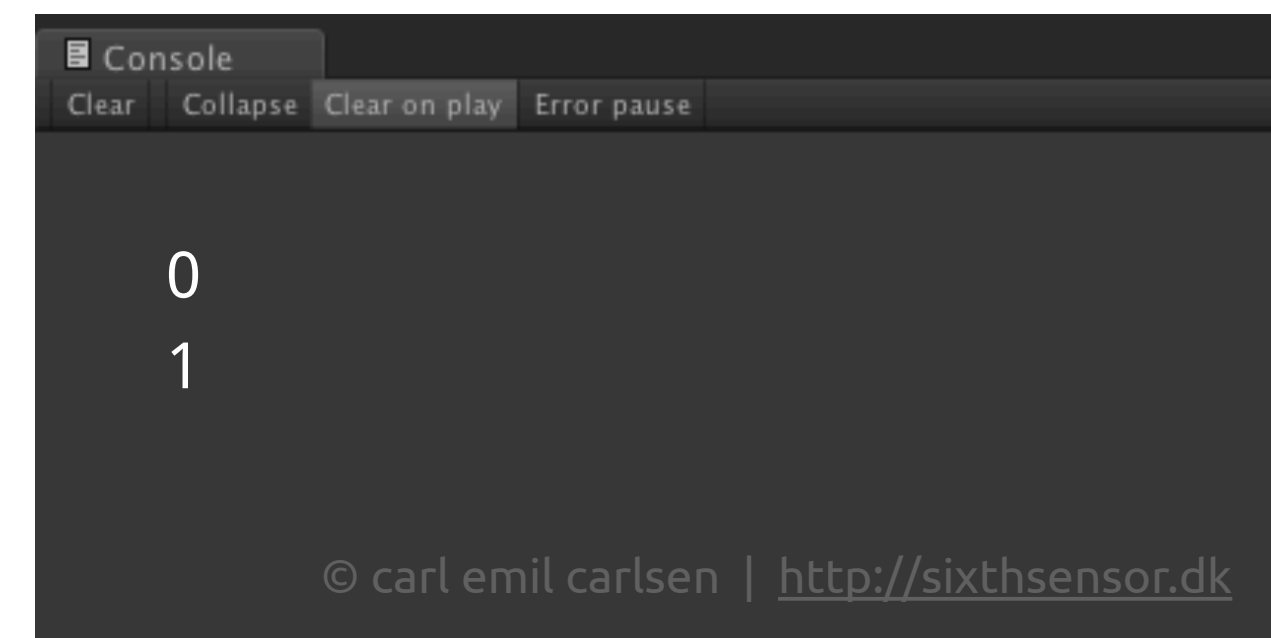
logging the
value of 'i'

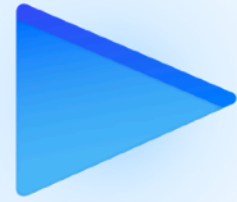
the for-statement



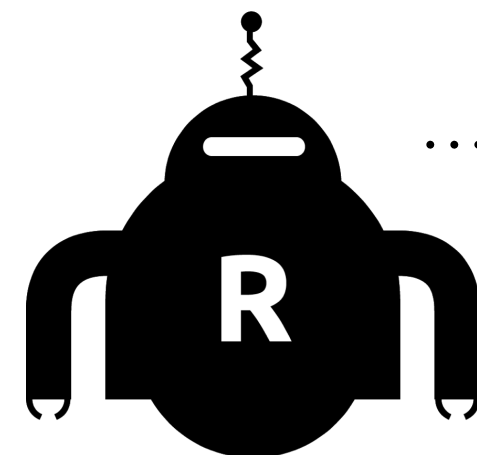


the for-statement



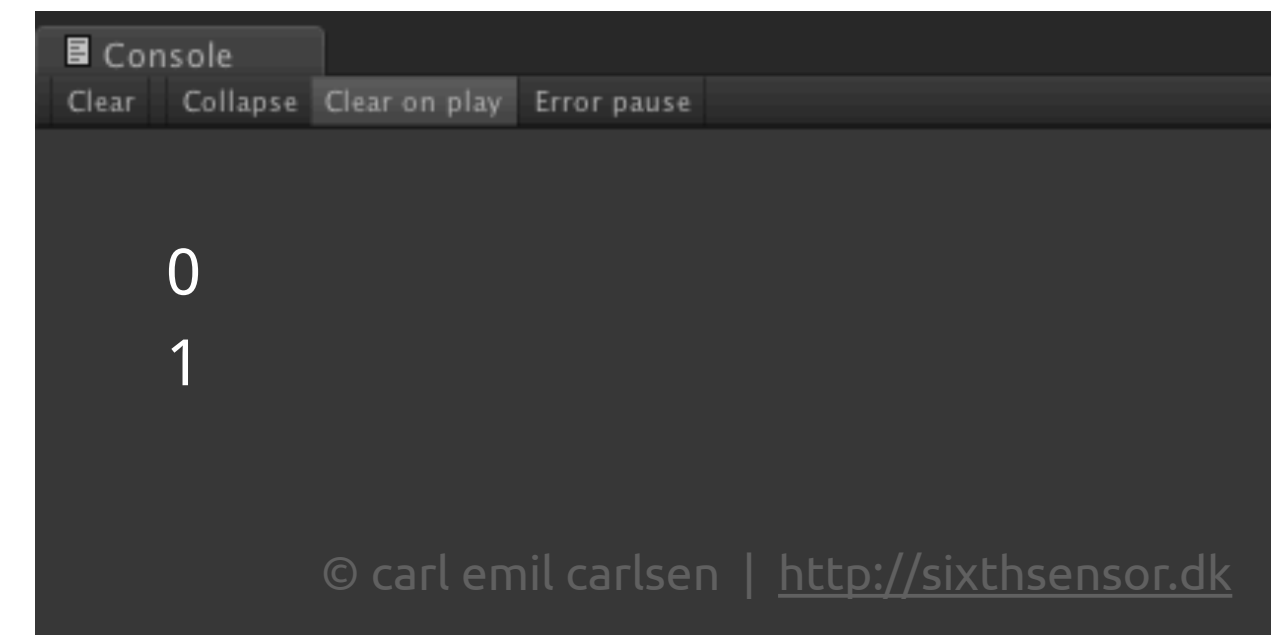


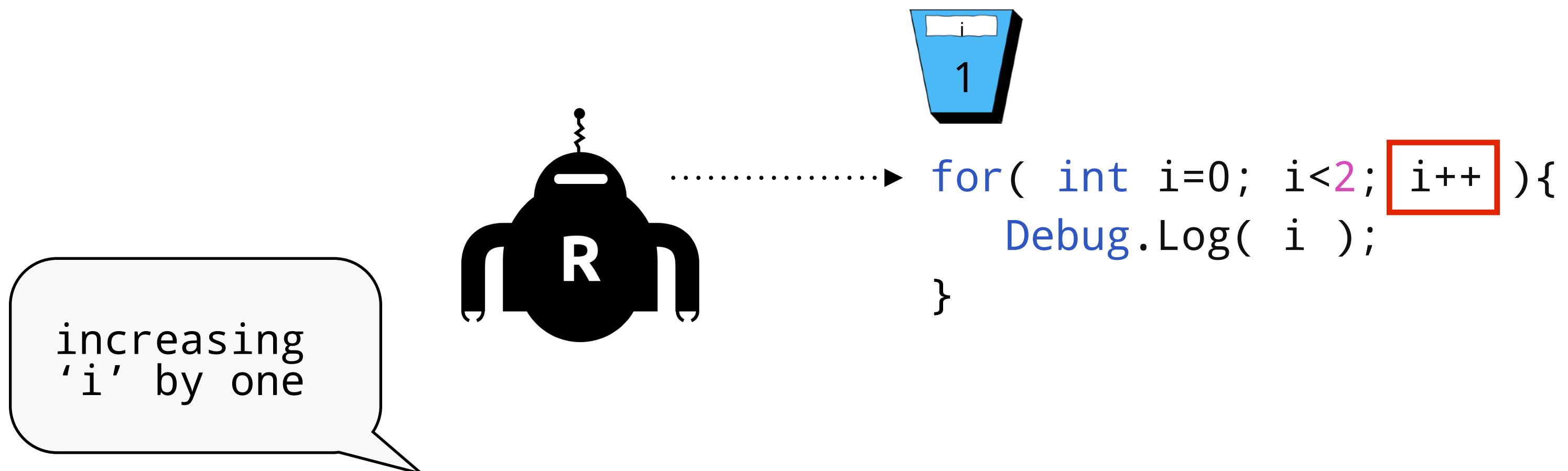
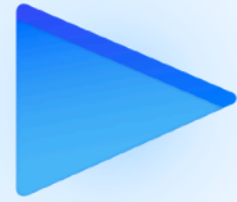
```
for( int i=0; i<2; i++ ){  
    Debug.Log( i );  
}
```



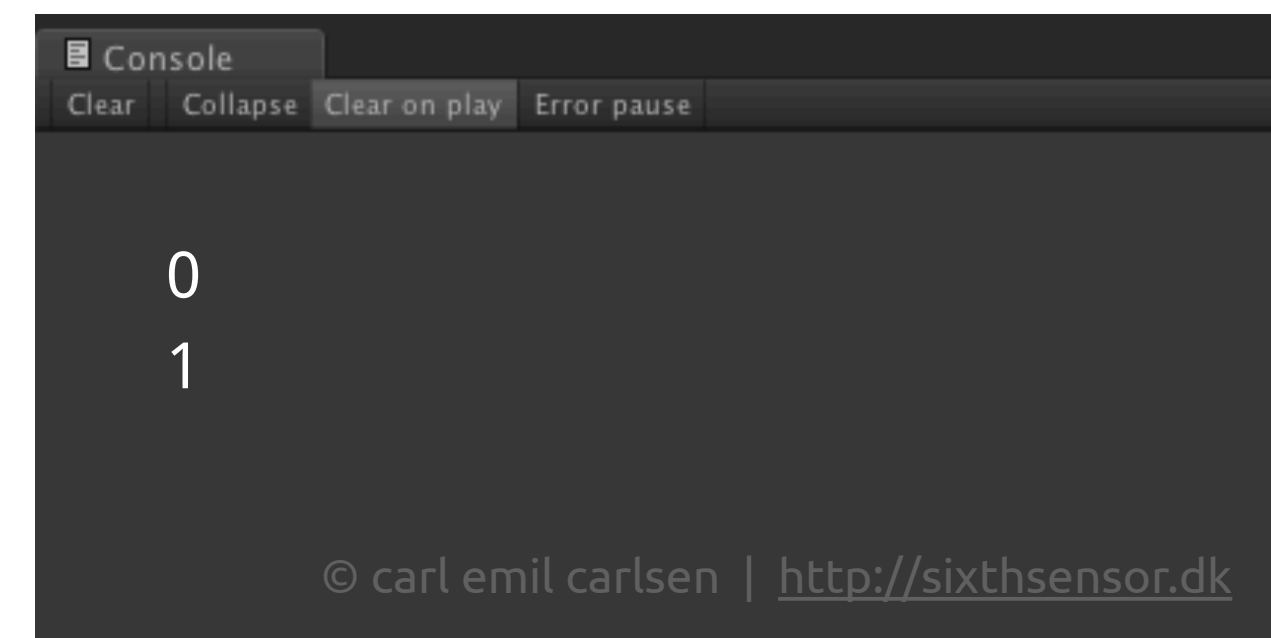
back to the top
once again

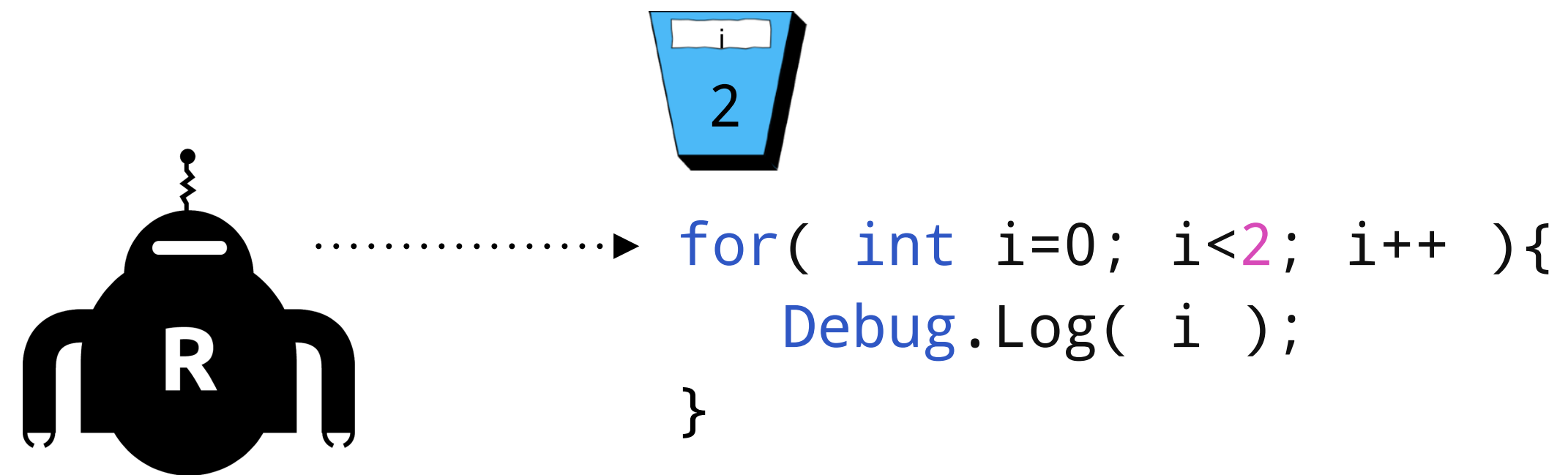
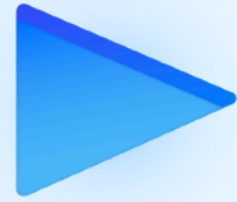
the for-statement



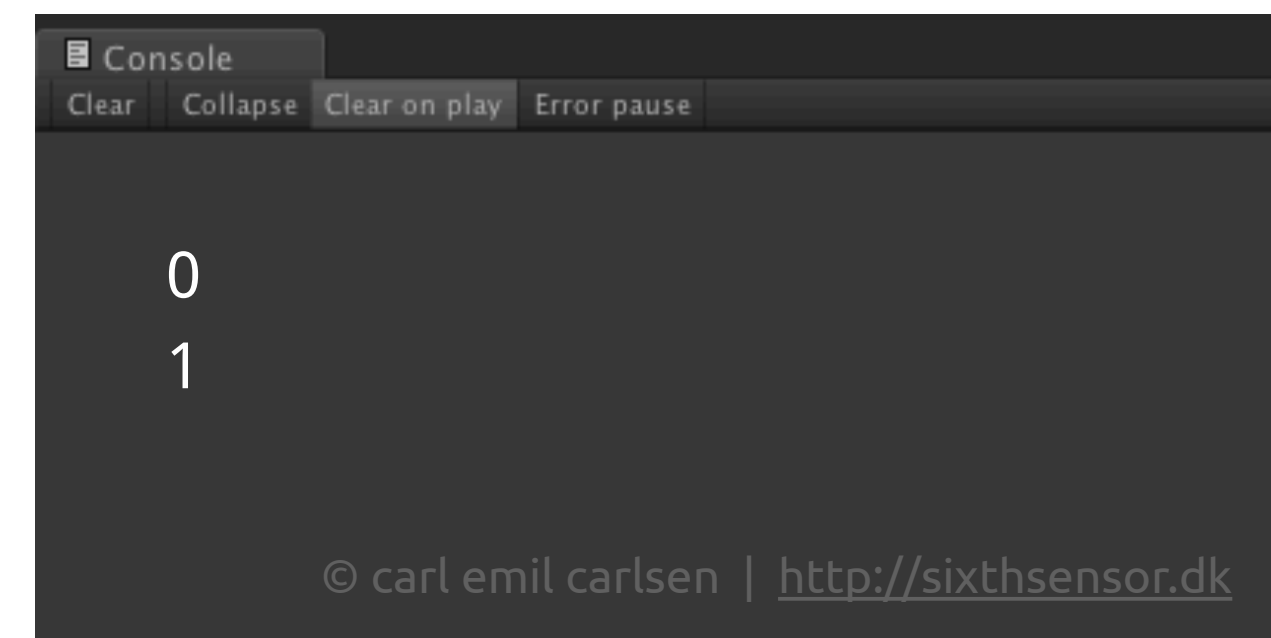


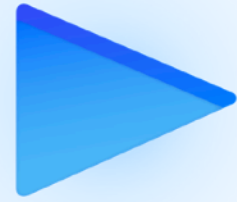
the for-statement



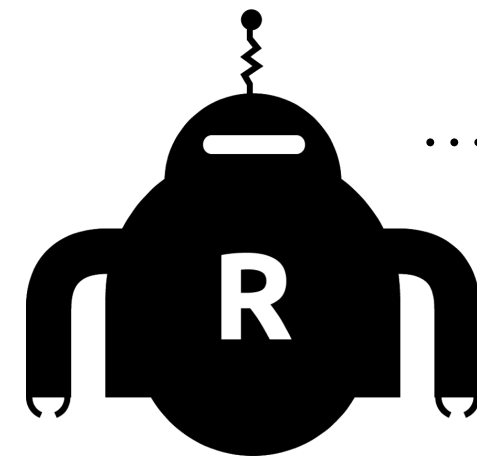


the for-statement





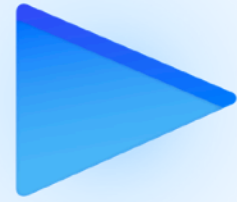
checking if
condition is true



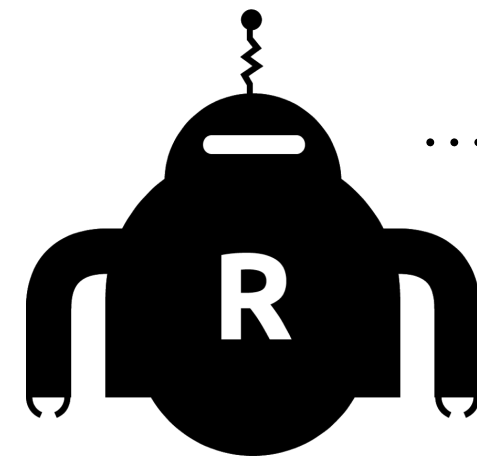
```
for( int i=0; i<2; i++ ){  
    Debug.Log( i );  
}
```

the for-statement



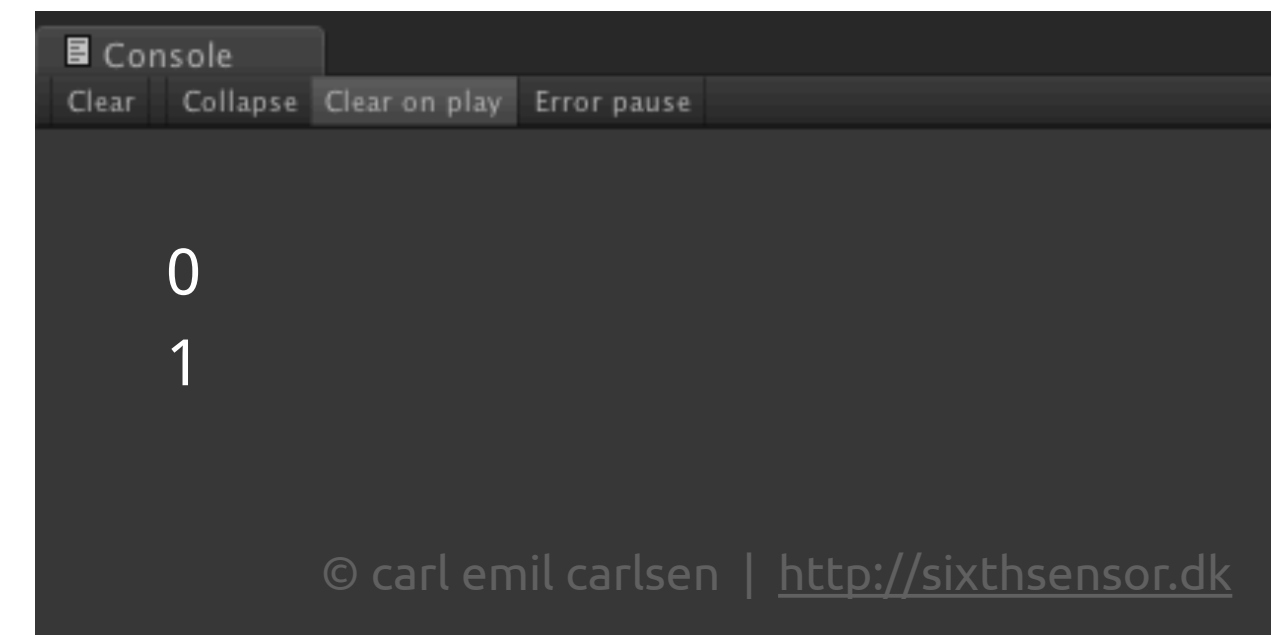


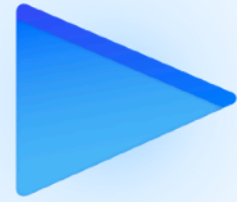
no. the loop
has finished,
moving on now



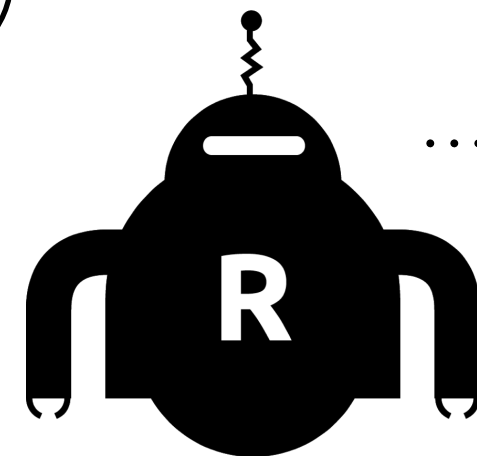
```
for( int i=0; false; i++ ){  
    Debug.Log( i );  
}
```

the for-statement



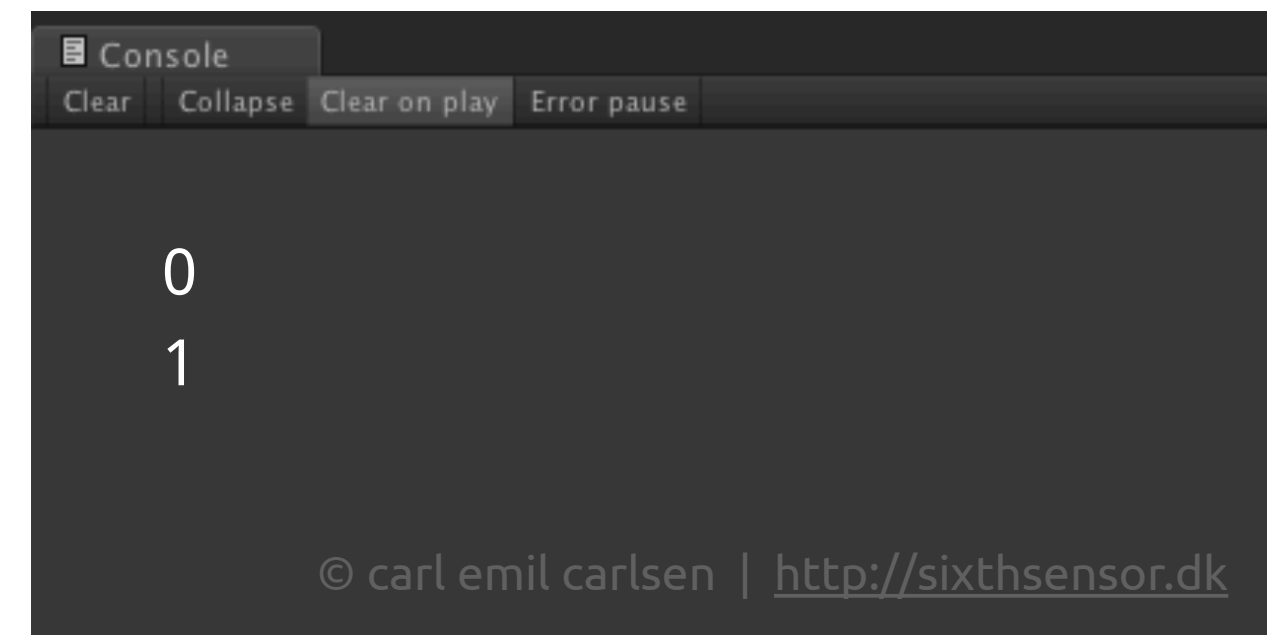


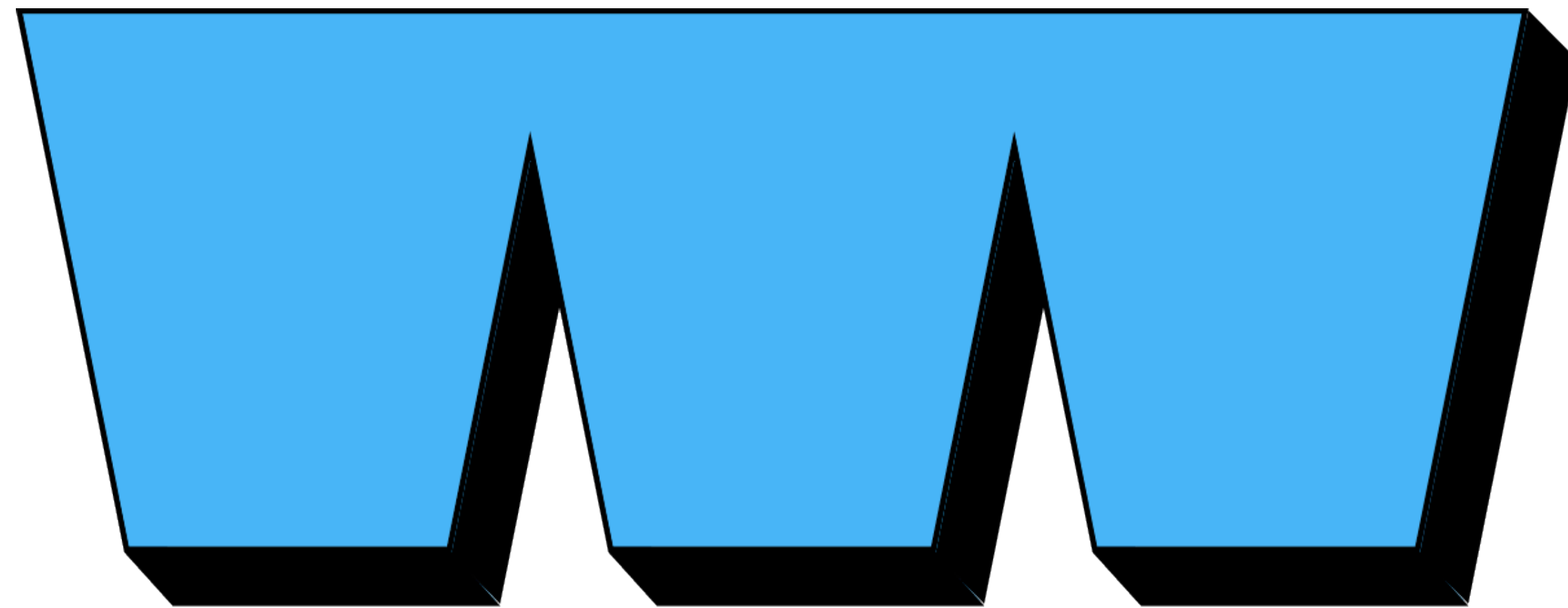
nothing here.
job done!



```
for( int i=0; i<2; i++ ){  
    Debug.Log( i );  
}
```

the for-statement





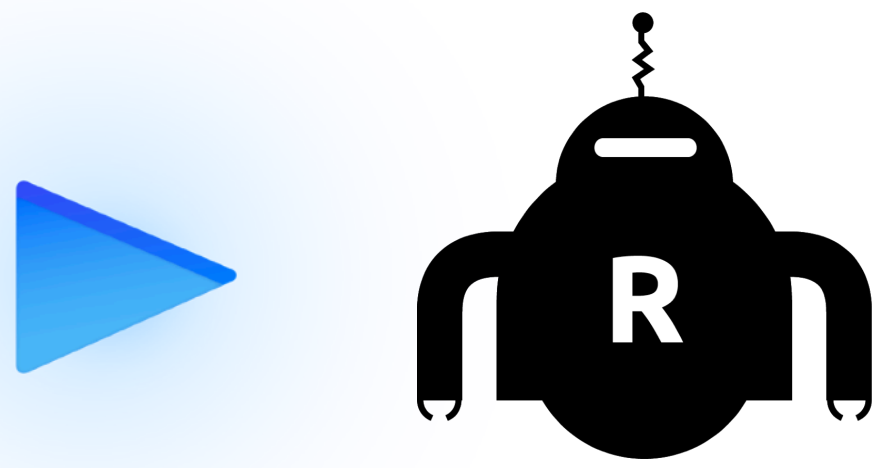
arrays

```
int[] ages = new int[ 3 ];
```

defining an array

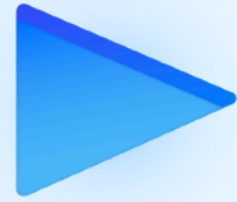
data type	array!	name	assignment operator	make a new ...	data type	number of elements
int[]		ages	=	new	int[3];

defining an array

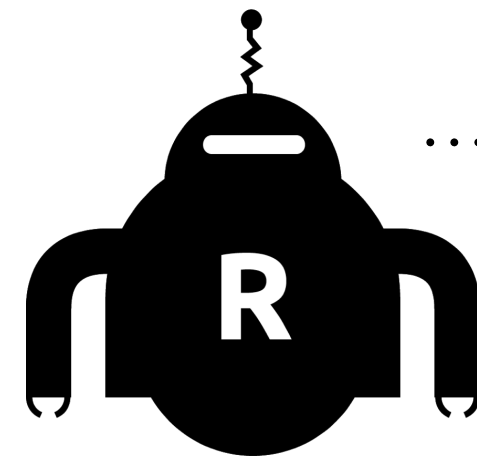


```
int[] ages = new int[ 3 ];
```

defining an array

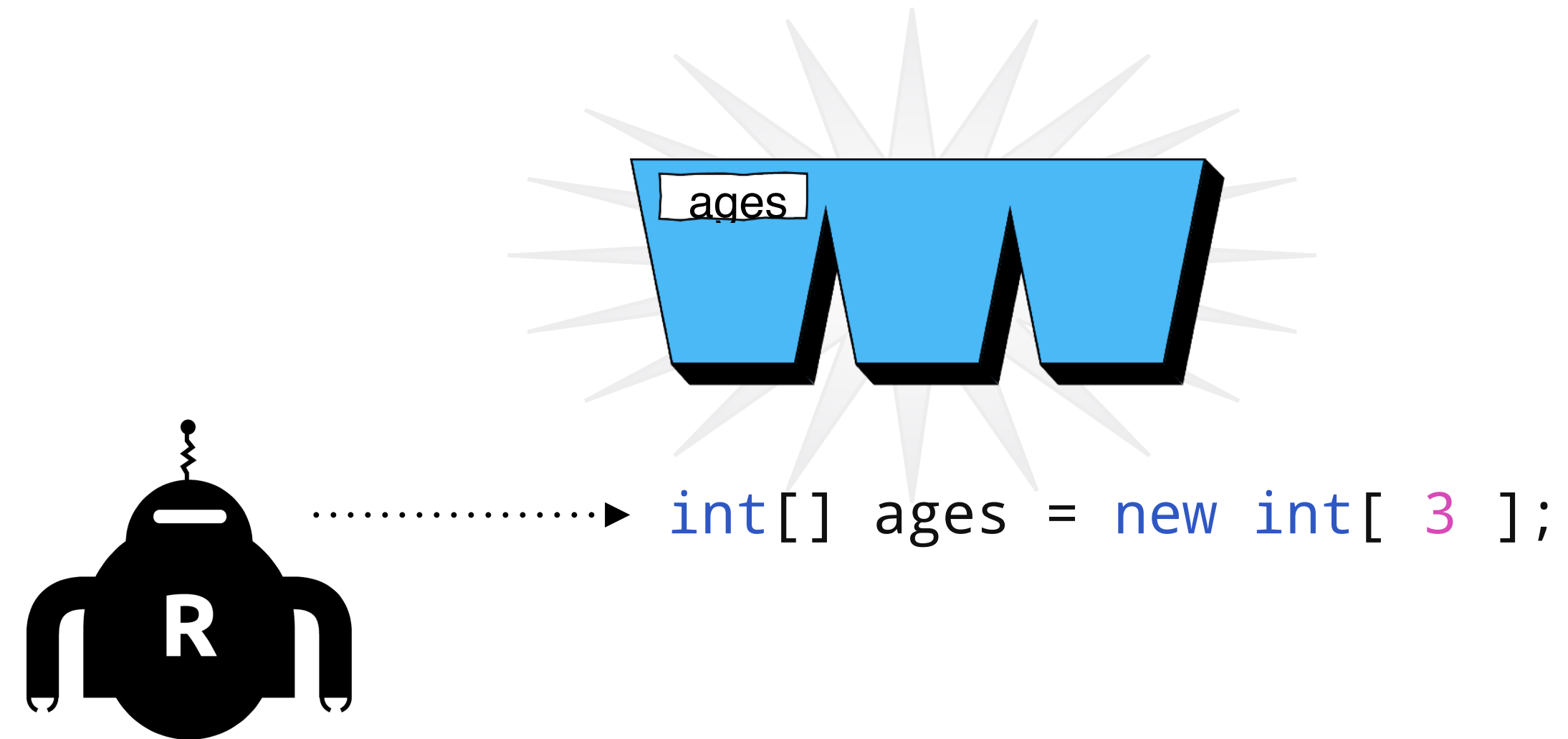
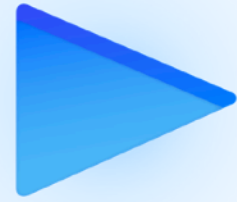


creating an array
named 'ages' with
space for three
values



.....▶ `int[] ages = new int[3];`

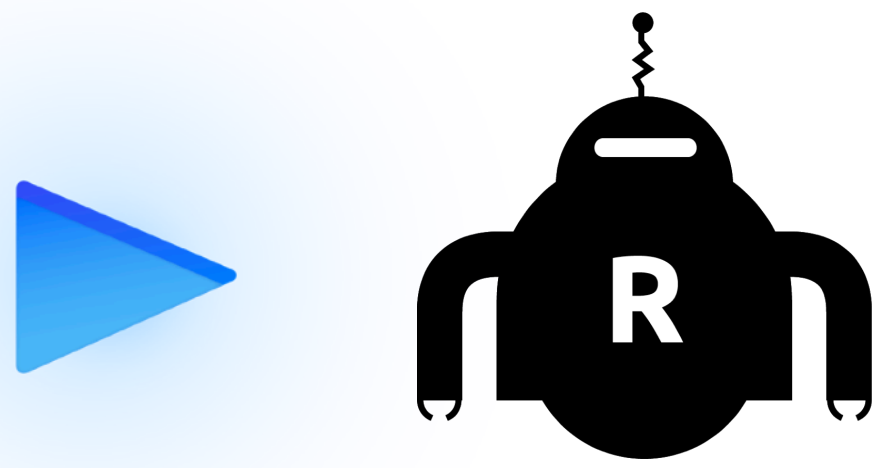
defining an array



defining an array

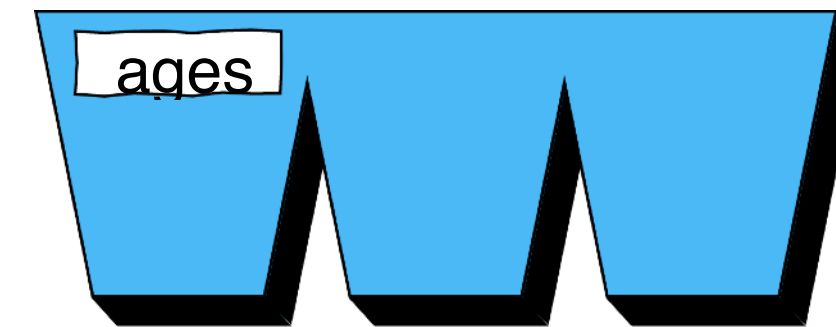
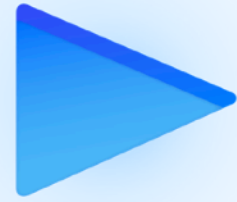
```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;
```

setting values in an array

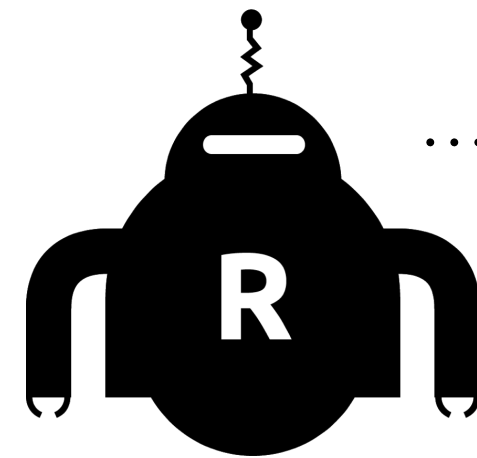


```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;
```

setting values in an array

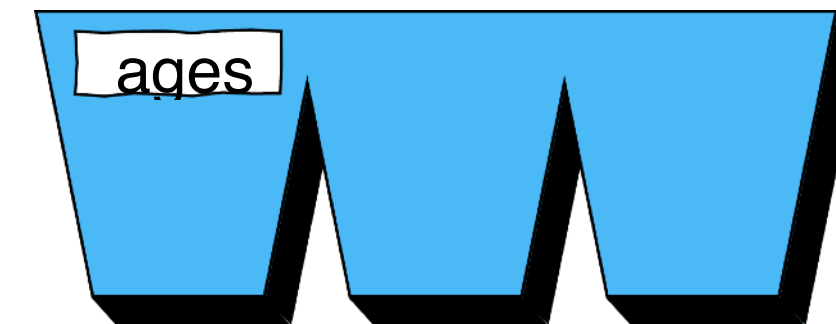
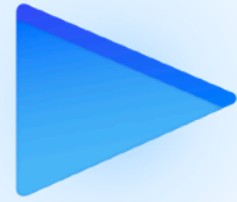


created an array
named 'ages' with
space for three
values

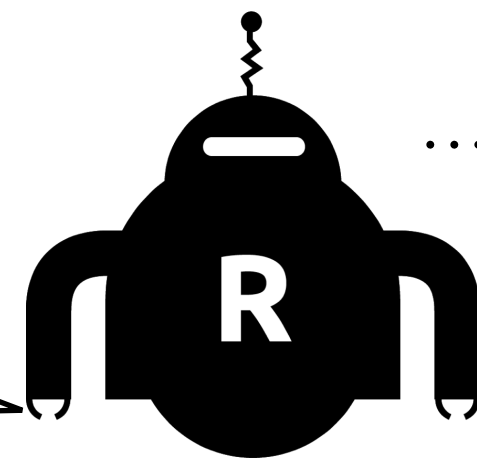


```
.....▶ int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;
```

setting values in an array

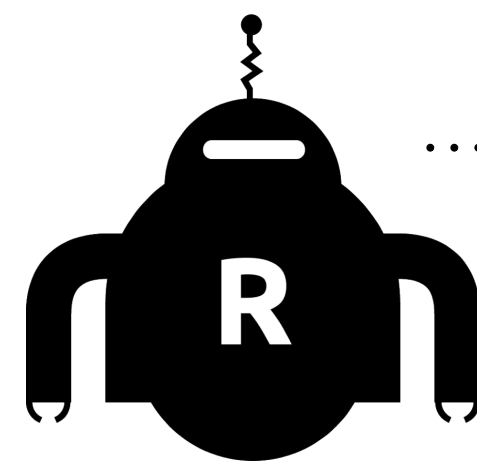
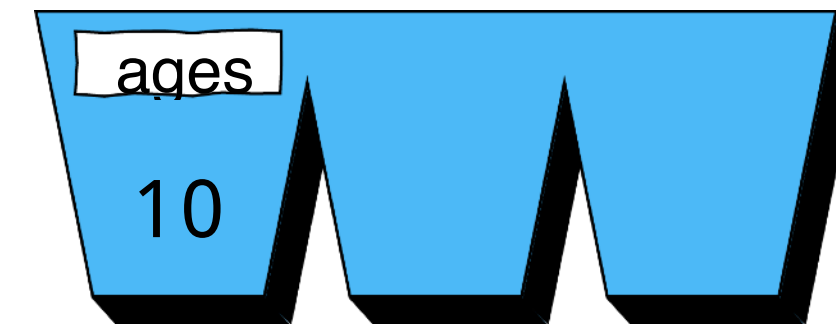
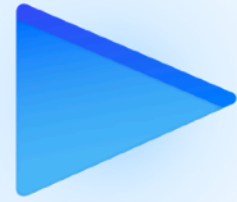


setting value at
index 0 in the
array 'ages' to 10



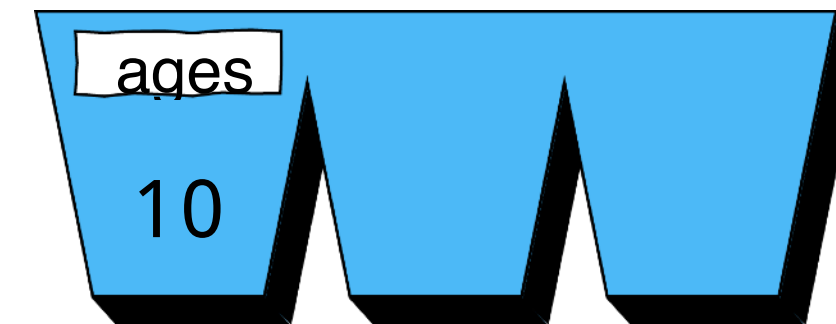
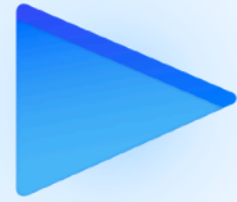
```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;
```

setting values in an array

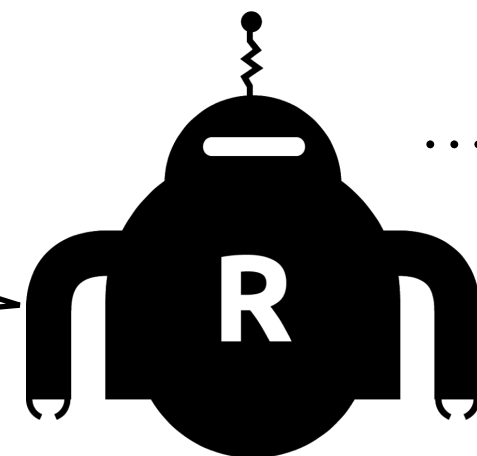


```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;
```

setting values in an array

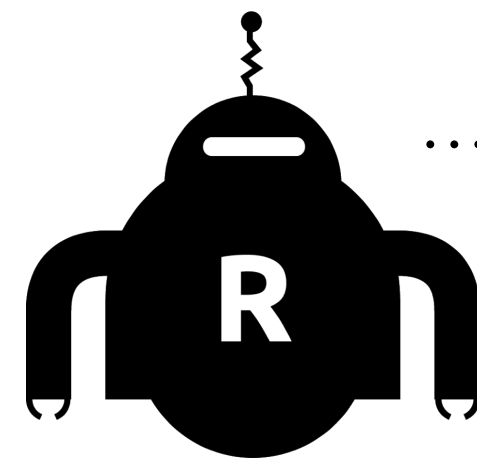
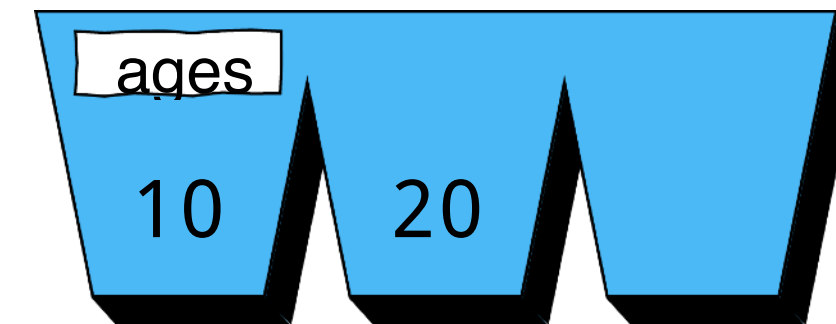
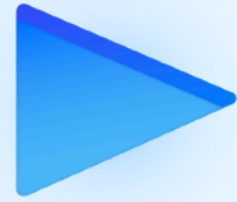


setting value at
index 1 in the
array 'ages' to 20



```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;
```

setting values in an array

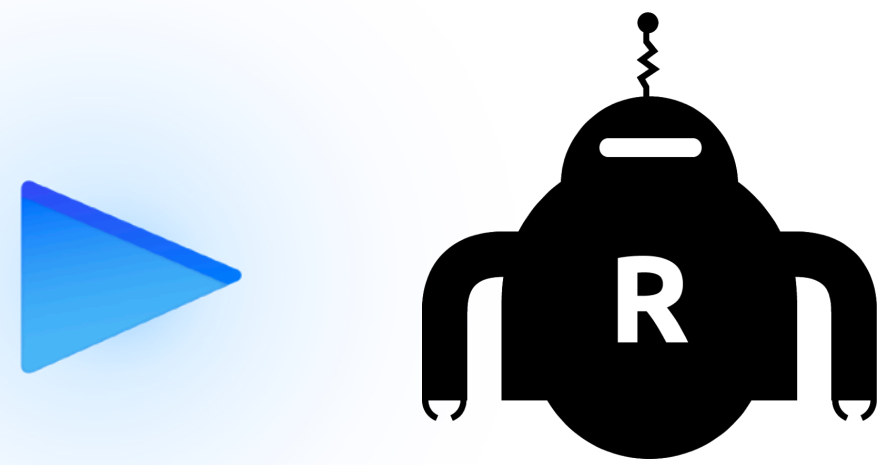


```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;
```

setting values in an array

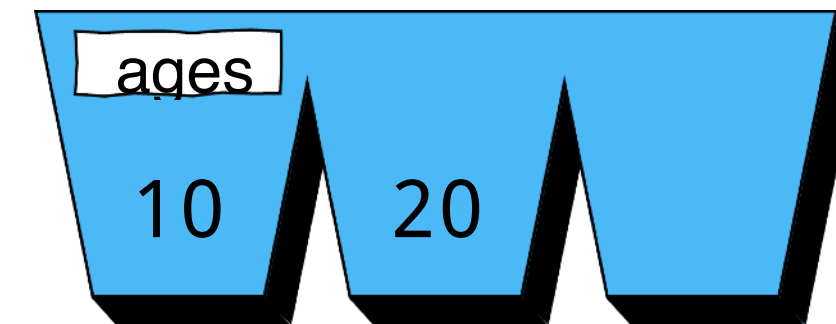
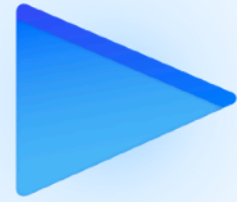
```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;  
ages[ 2 ] = ages[ 0 ] + ages[ 1 ];
```

getting values from an array

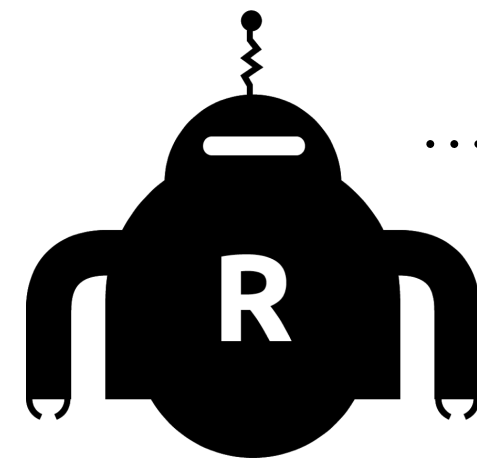


```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;  
ages[ 2 ] = ages[ 0 ] + ages[ 1 ];
```

getting values from an array

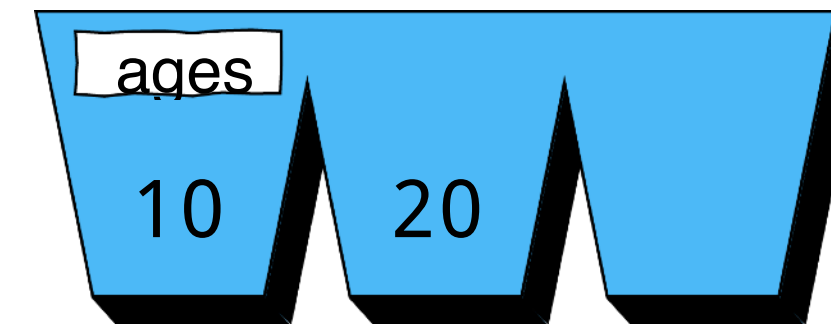
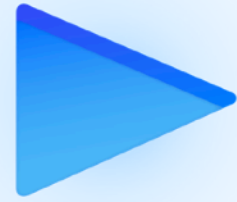


I have defined an array named 'ages' and set set 10 at index 0 and 20 at index 1

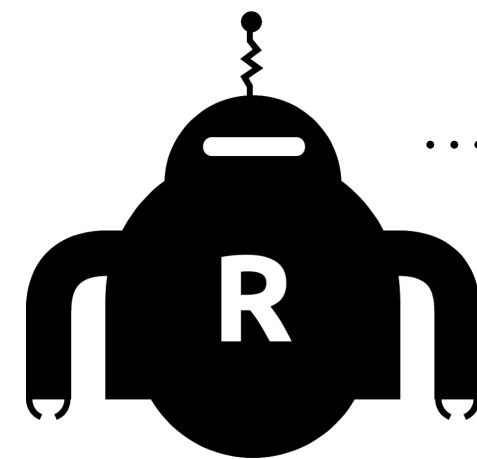


```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;  
ages[ 2 ] = ages[ 0 ] + ages[ 1 ];
```

getting values from an array

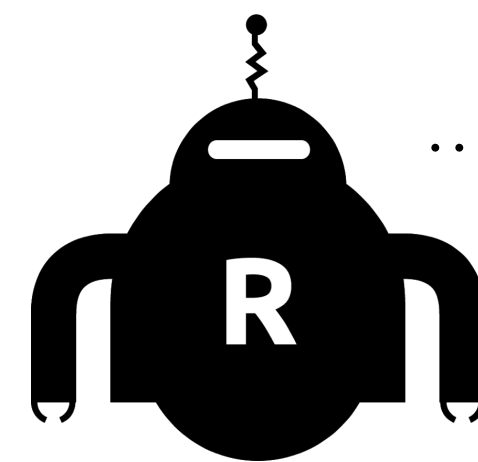
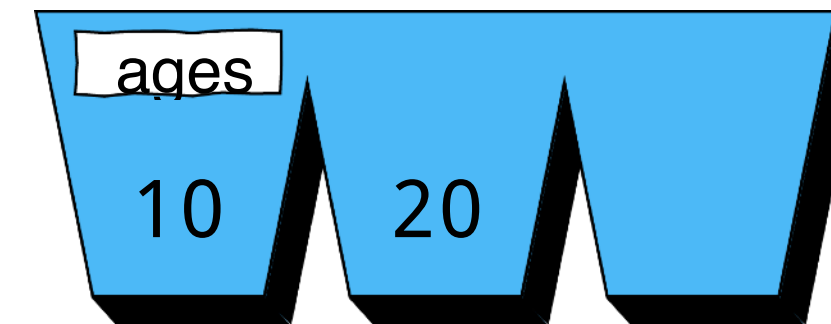
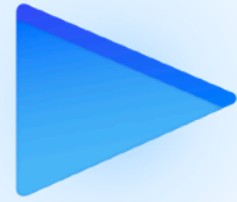


getting value from
index 0 and index 1 in
the array 'ages'



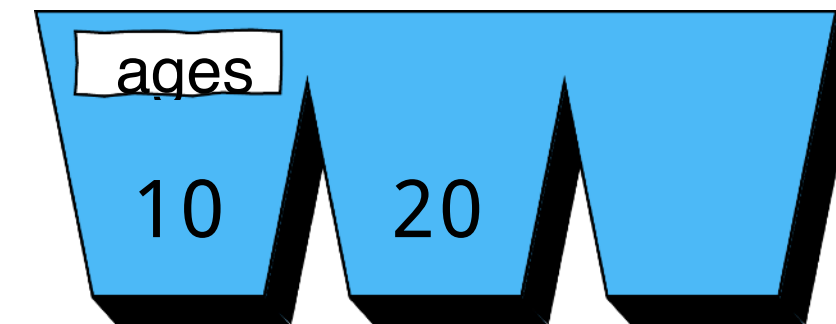
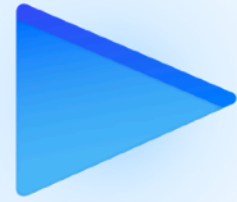
```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;  
ages[ 2 ] = ages[ 0 ] + ages[ 1 ];
```

getting values from an array

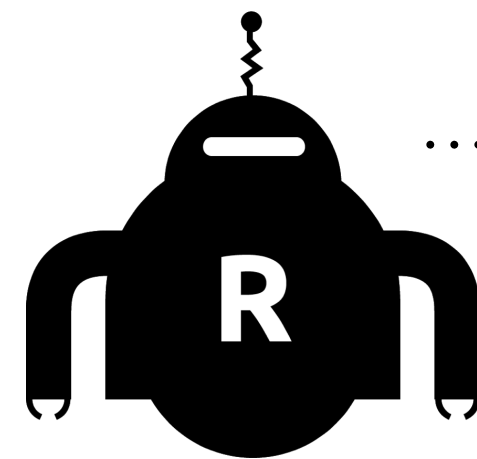


```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;  
ages[ 2 ] = 10 + 20;
```

getting values from an array

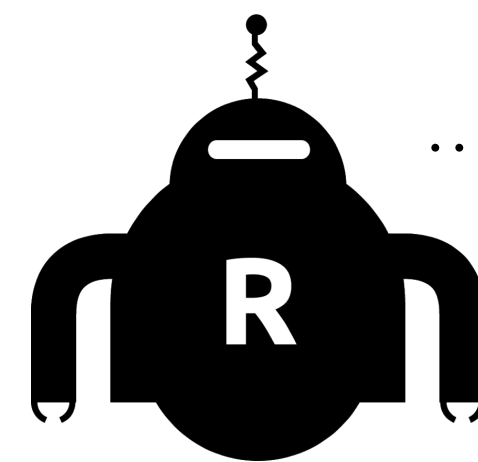
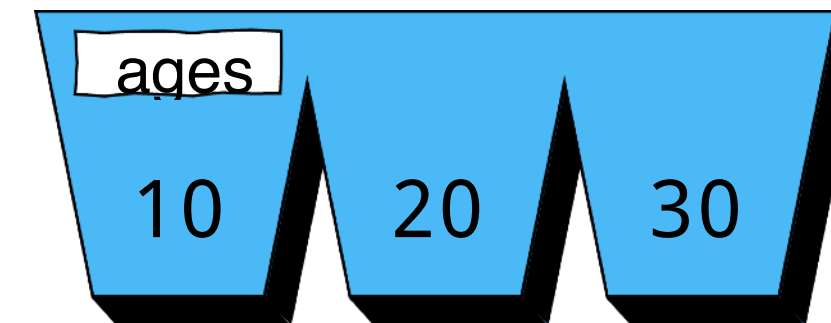
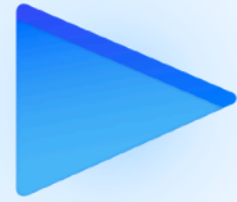


adding values and
storing them at index 2
in the array 'ages'



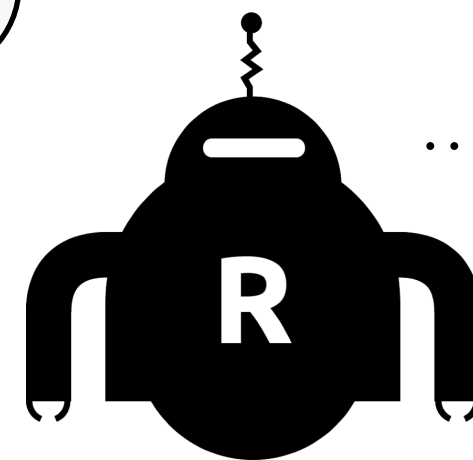
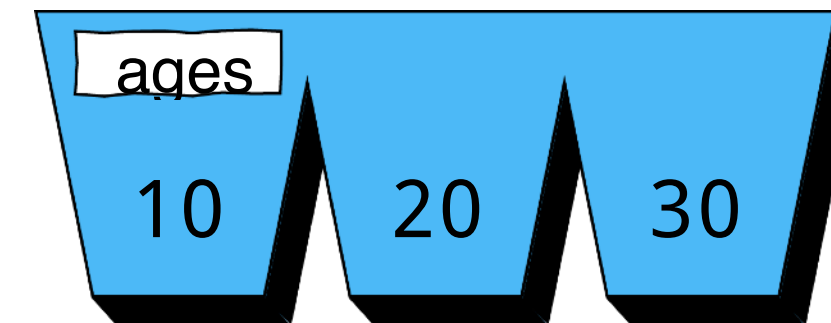
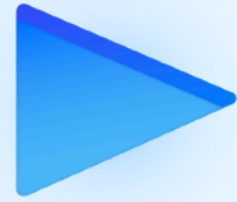
```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;  
ages[ 2 ] = 10 + 20;
```

getting values from an array



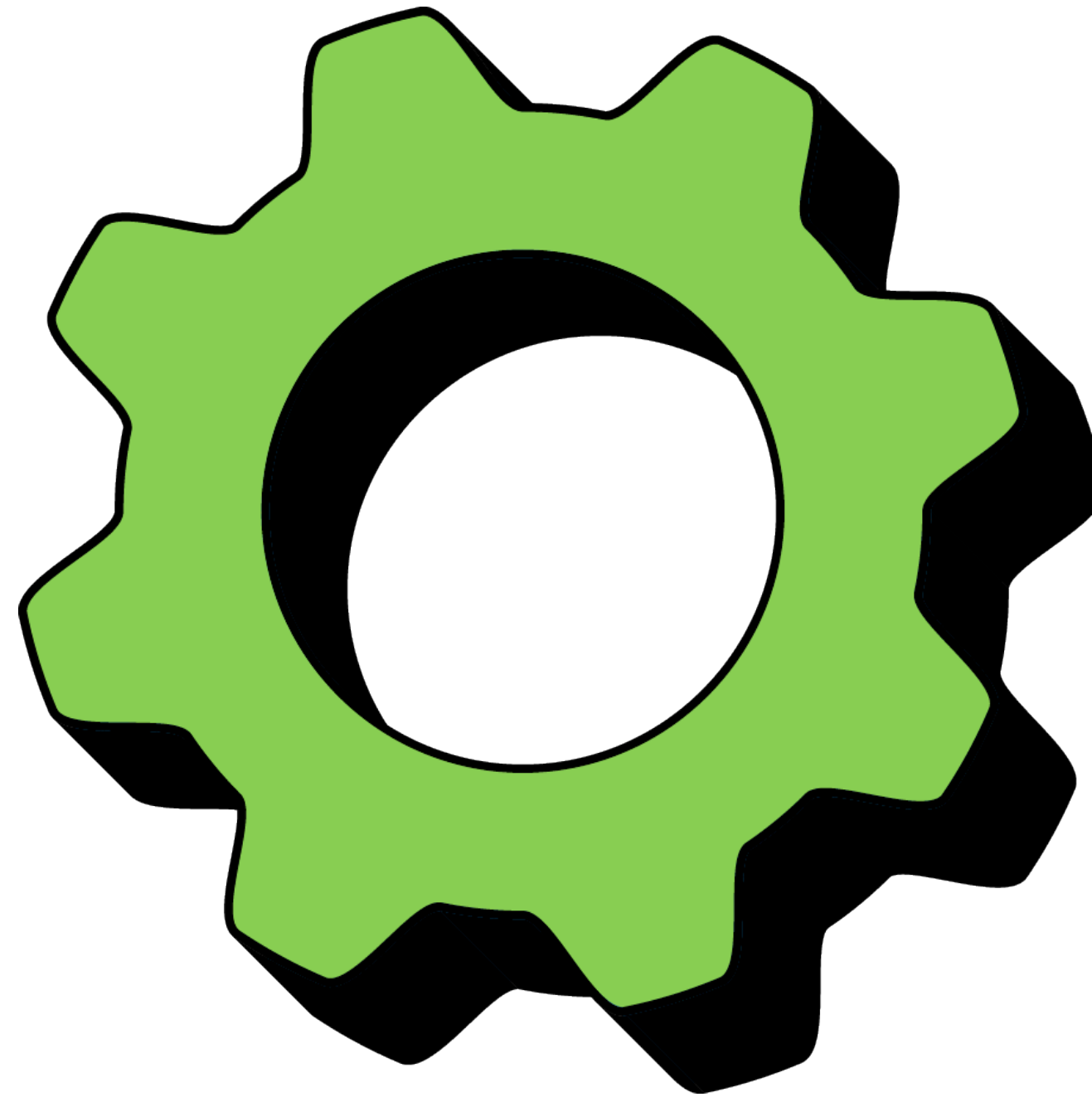
```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;  
ages[ 2 ] = 30;
```

getting values from an array




```
int[] ages = new int[ 3 ];  
ages[ 0 ] = 10;  
ages[ 1 ] = 20;  
ages[ 2 ] = ages[ 0 ] + ages[ 1 ];
```

getting values from an array




defining a function with
arguments and return value




```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```


defining a function with
arguments and return value

	return type	name	arg. 1 type	arg. 1 name	arg. 2 type	arg. 2 name
	<u>bool</u>	<u>MaybeSay</u>	<u>string</u>	<u>prose</u>	<u>float</u>	<u>chance</u>
) {
						bool doSay = Random.value < chance;
						if(doSay){
						Debug.Log(prose);
						}
						<u>return</u> doSay;
						}
						end
						return statement

defining a function with
arguments and return value

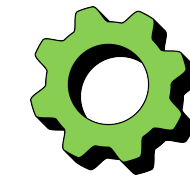
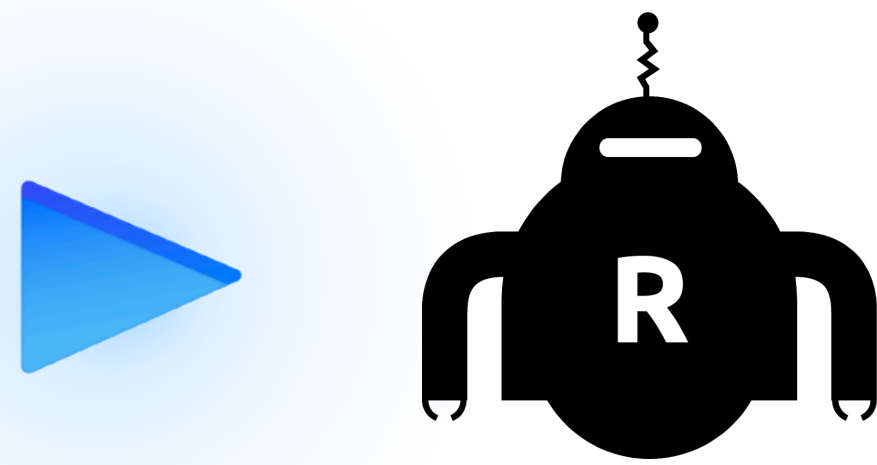


```
void Start(){
    bool didSay = MaybeSay( "hurrah", 0.5f );
    Debug.Log( "did say hurrah: " + didSay );
}
```

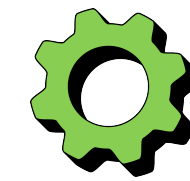


```
bool MaybeSay( string prose, float chance ){
    bool doSay = Random.value < chance;
    if(doSay){
        Debug.Log(prose);
    }
    return doSay;
}
```

defining a function with
arguments and return value

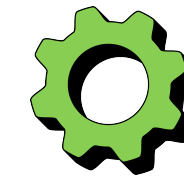
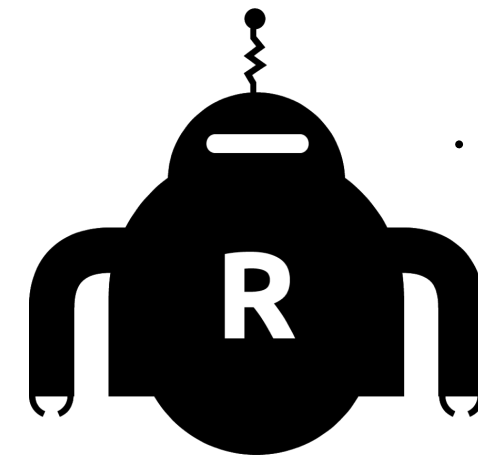


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```



```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

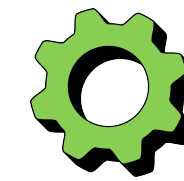
defining a function with
arguments and return value



```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

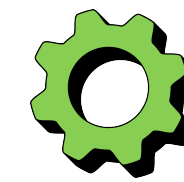
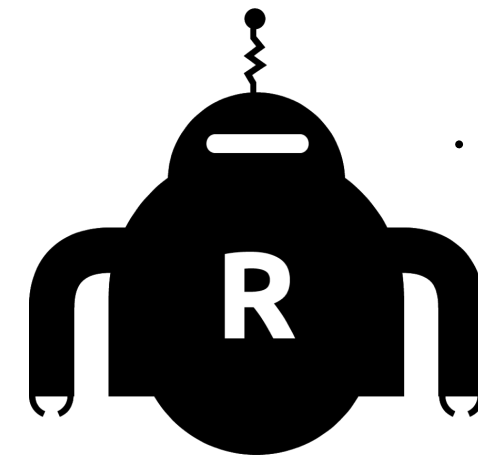
didSay

creating a
local variable



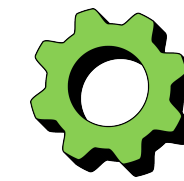
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value



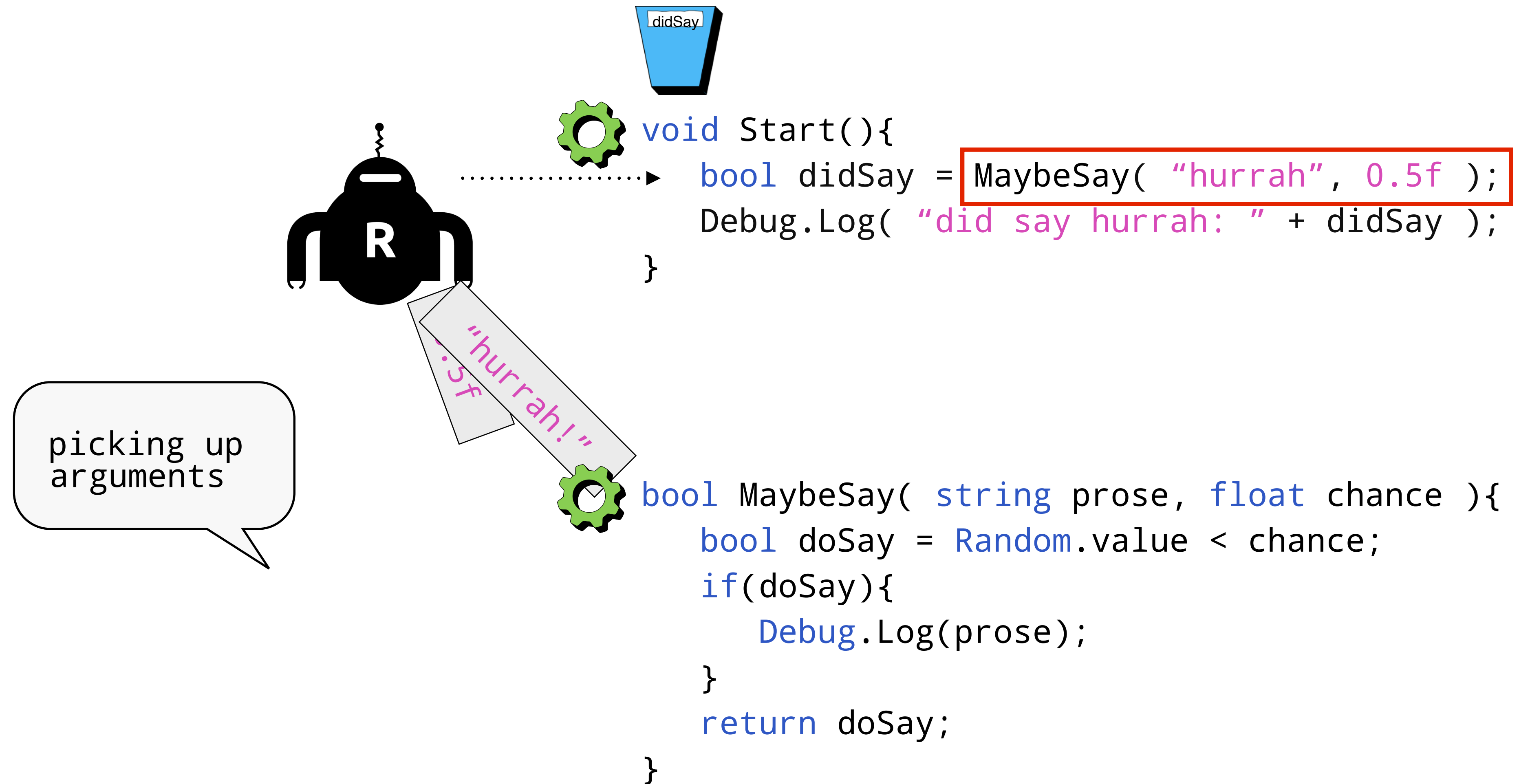
```
void Start(){  
  bool didSay = MaybeSay( "hurrah", 0.5f );  
  Debug.Log( "did say hurrah: " + didSay );  
}
```

calling MaybeSay

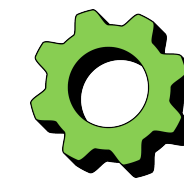
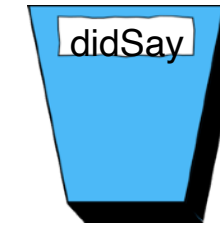


```
bool MaybeSay( string prose, float chance ){  
  bool doSay = Random.value < chance;  
  if(doSay){  
    Debug.Log(prose);  
  }  
  return doSay;  
}
```

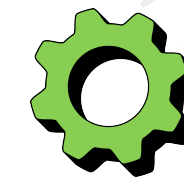
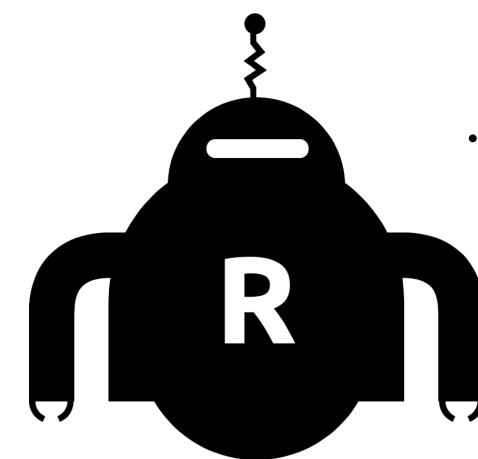
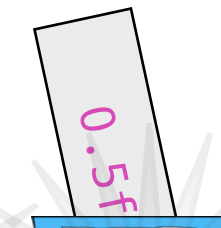
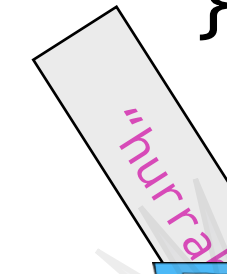
defining a function with
arguments and return value



defining a function with
arguments and return value



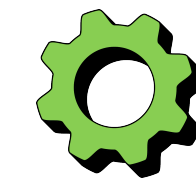
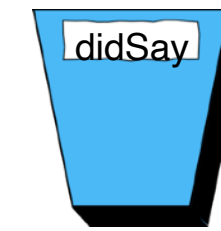
```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```



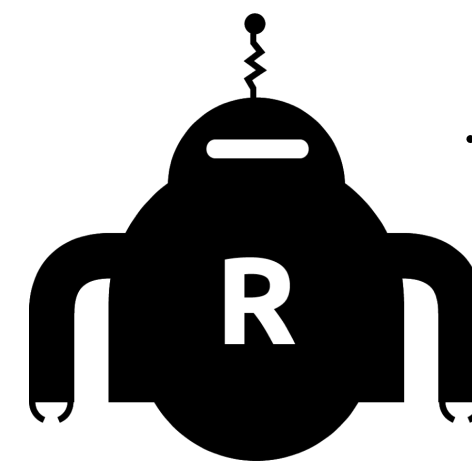
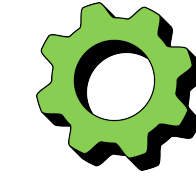
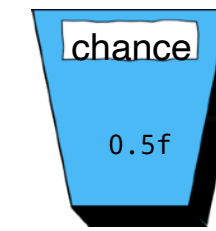
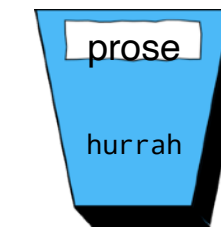
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

creating argument
variables

defining a function with
arguments and return value



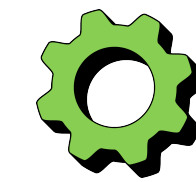
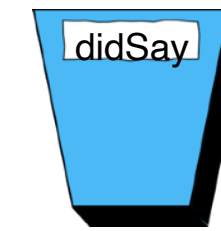
```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```



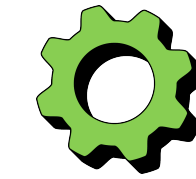
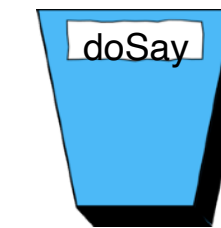
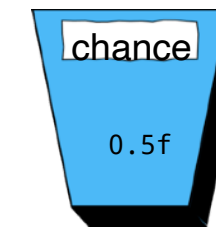
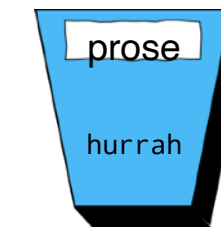
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

creating local
variable

defining a function with
arguments and return value

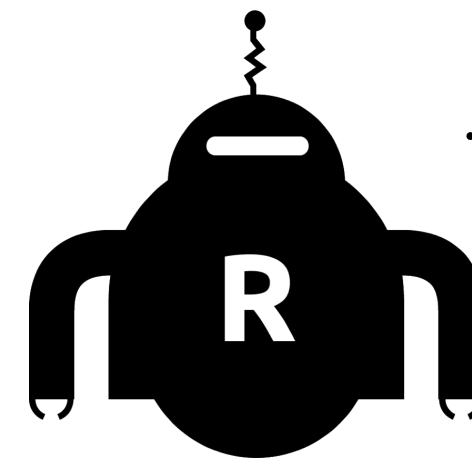


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

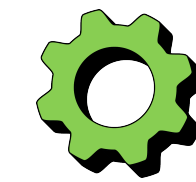


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

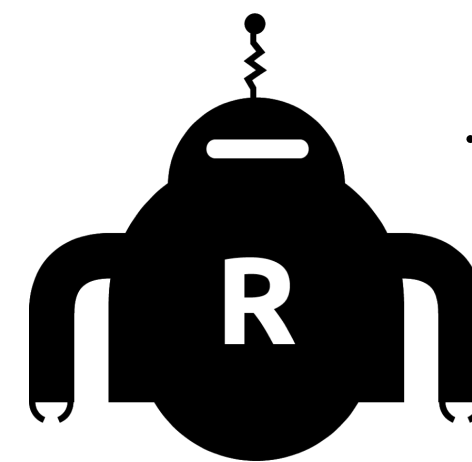
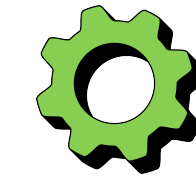
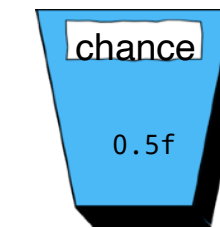
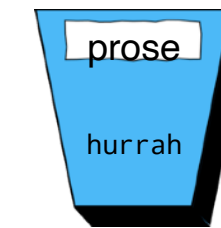
getting a random
value between 0 and
1 from the class
named Random



defining a function with
arguments and return value

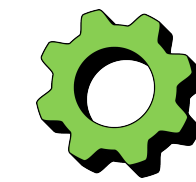
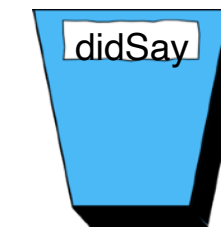


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

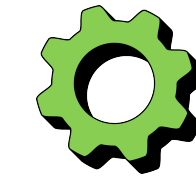
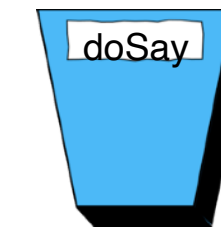
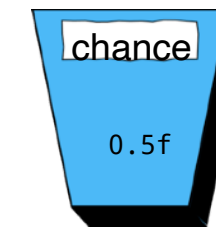
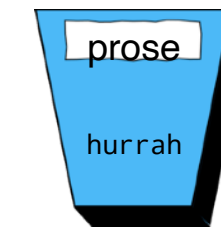


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = 0.243254315f < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value

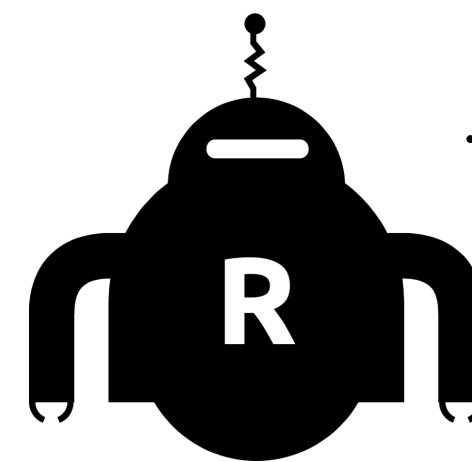


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

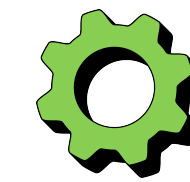


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = 0.243254315f < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

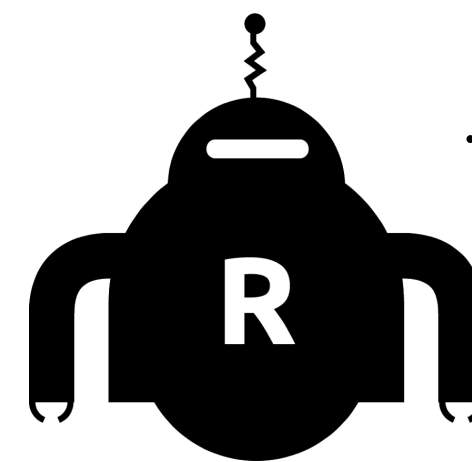
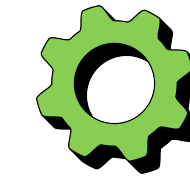
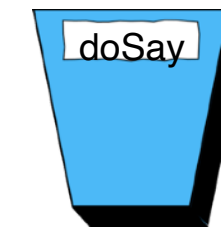
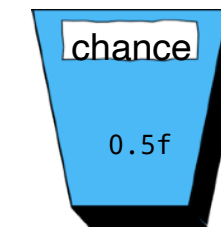
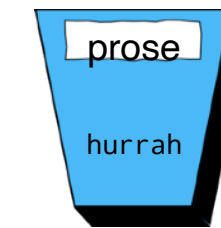
getting the value
of the variable
'chance'



defining a function with
arguments and return value

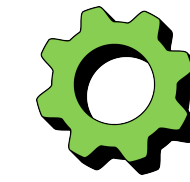
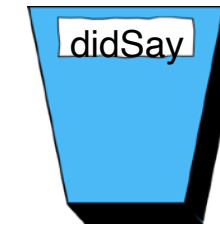


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

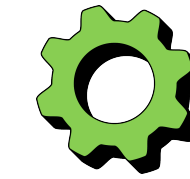
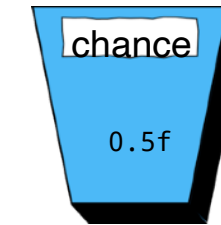
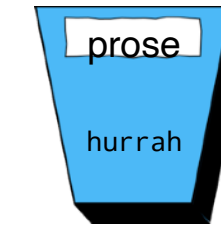


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = 0.243254315f < 0.5f;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value

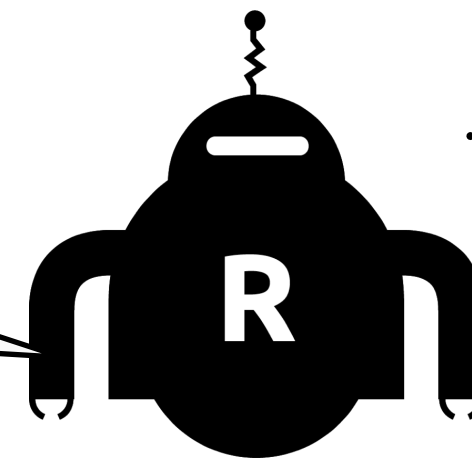


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

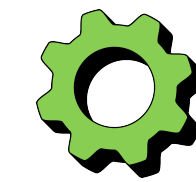
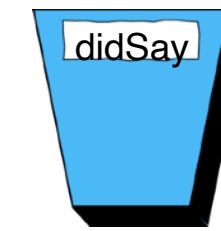


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = 0.243254315f < 0.5f;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

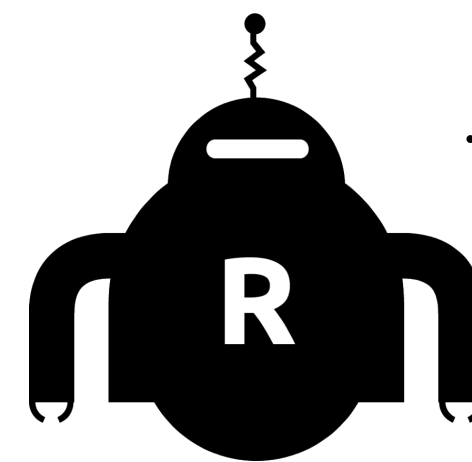
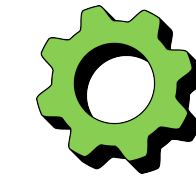
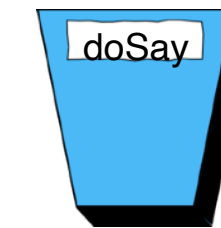
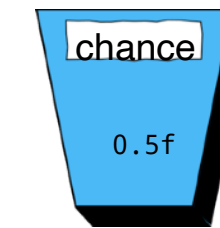
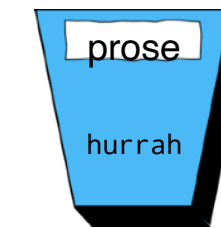
evaluating
logical
statement



defining a function with
arguments and return value

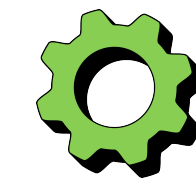


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

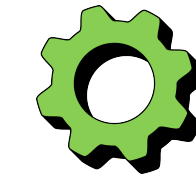
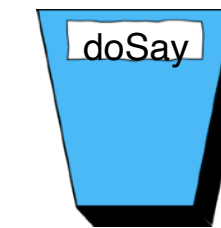
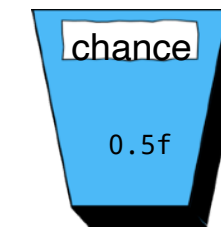
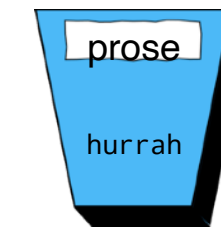


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = true;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value

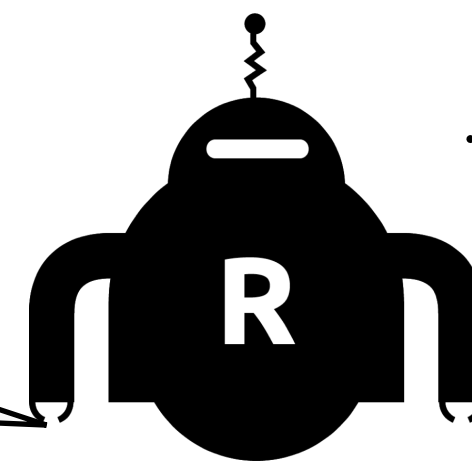


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

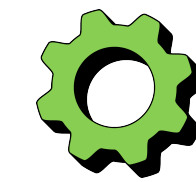
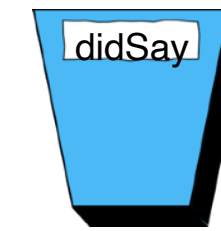


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = true;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

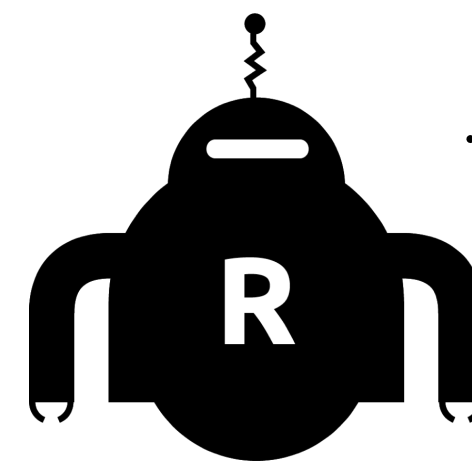
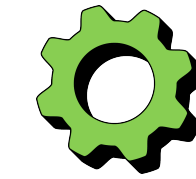
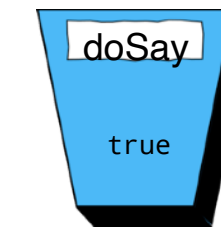
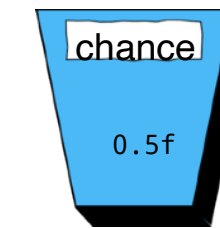
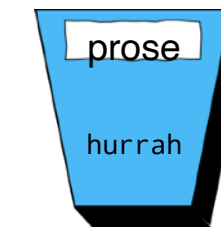
setting the
variable 'doSay' to



defining a function with
arguments and return value

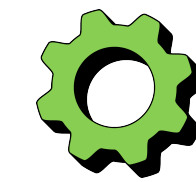
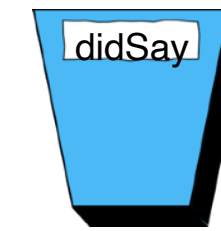


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

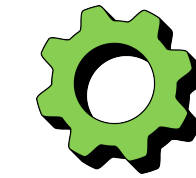
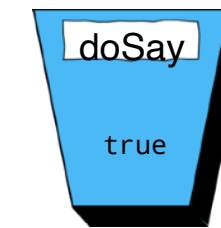
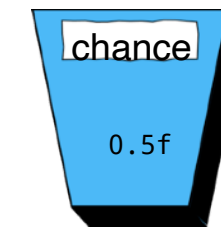
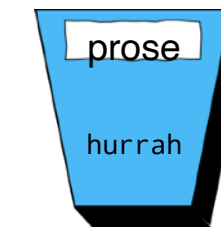


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = true;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value

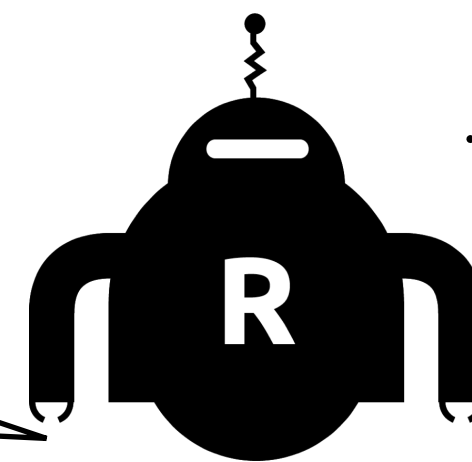


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

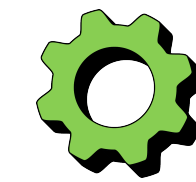


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

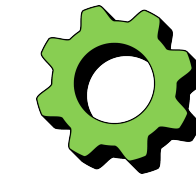
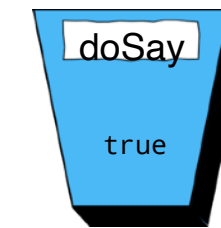
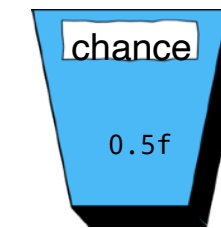
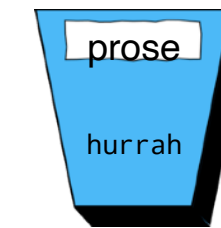
getting the value
of the variable



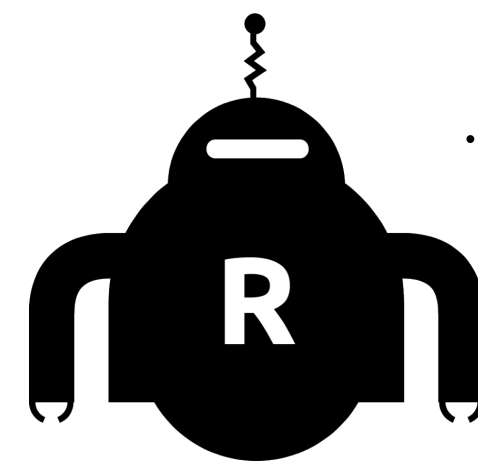
defining a function with
arguments and return value



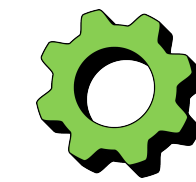
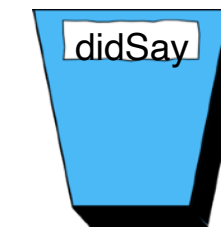
```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```



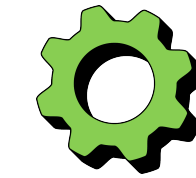
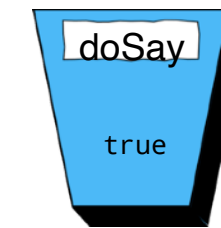
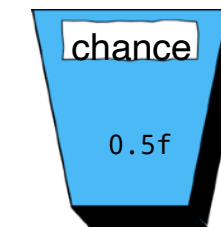
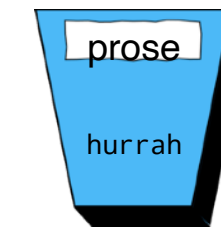
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(true){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```



defining a function with
arguments and return value

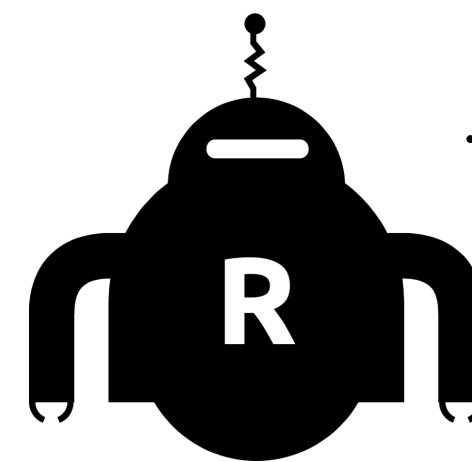


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

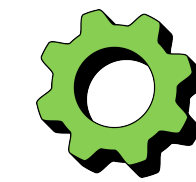
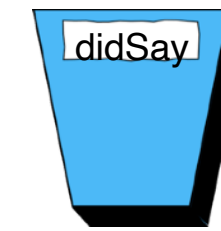


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(true){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

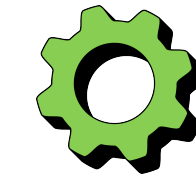
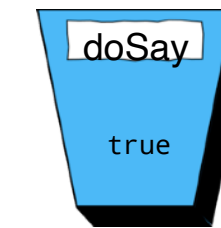
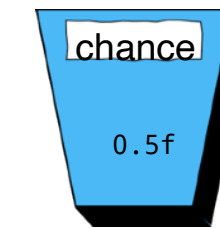
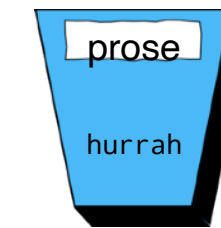
it's true, so I'll
execute the code
block marked by
brackets.



defining a function with
arguments and return value

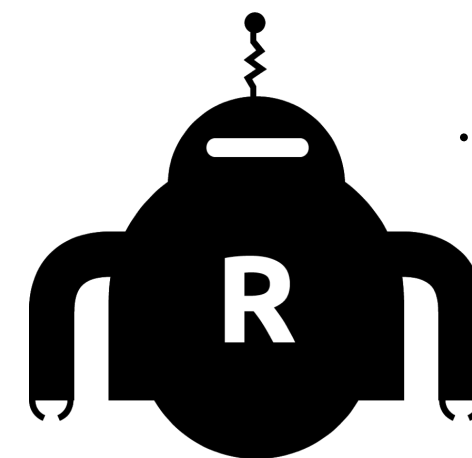


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

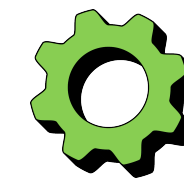


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(true){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

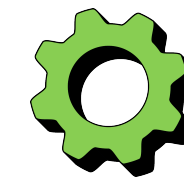
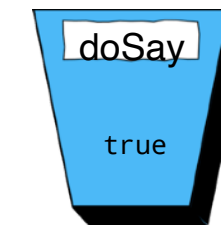
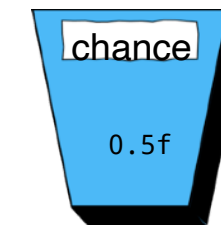
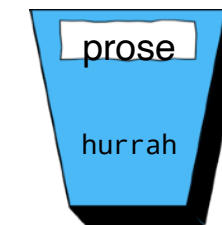
logging the value
of the variable
'prose'



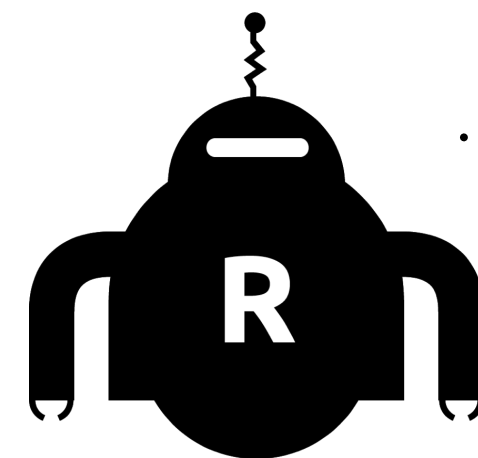
defining a function with
arguments and return value



```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

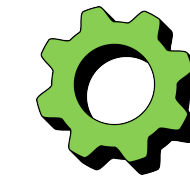
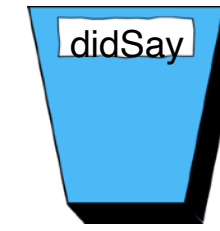


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(true){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

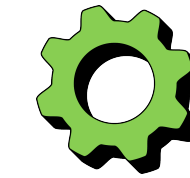
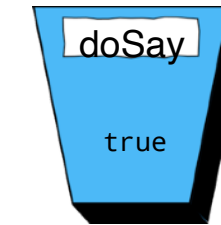
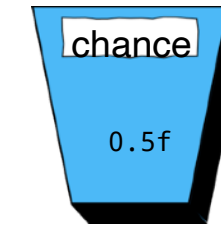
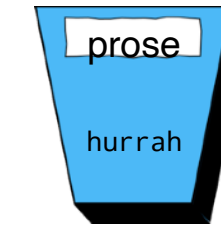


defining a function with
arguments and return value



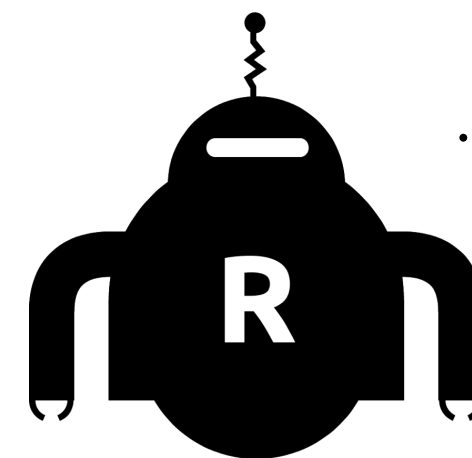


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

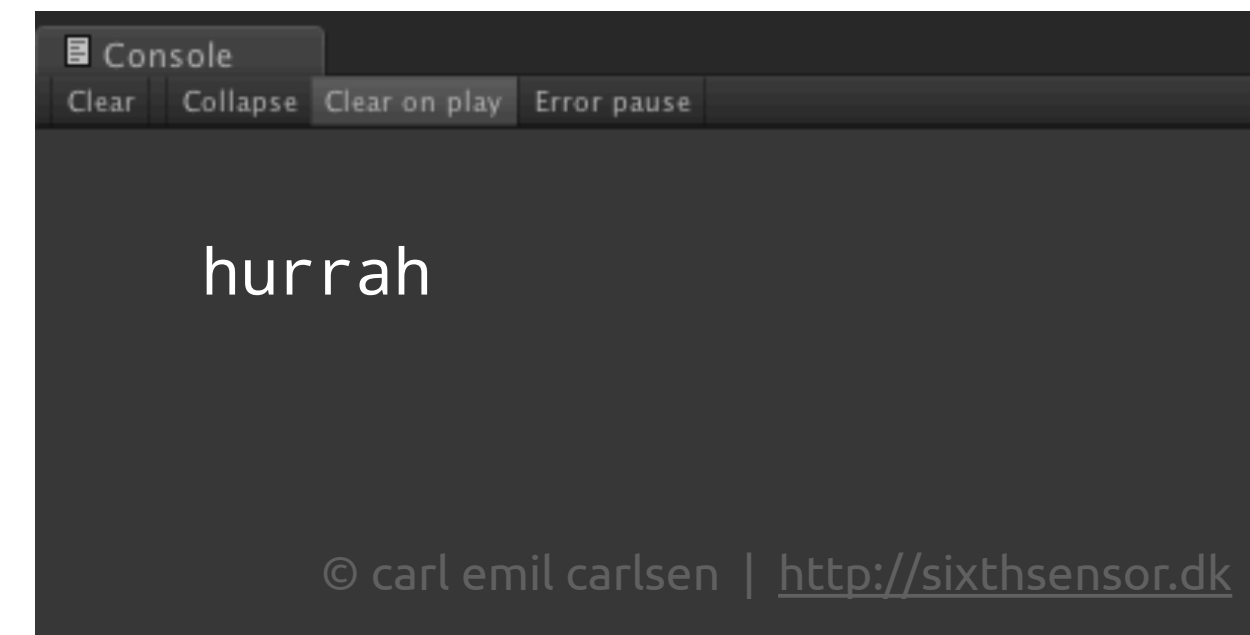


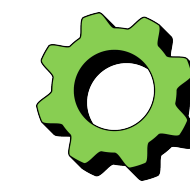
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(true){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

end of code block

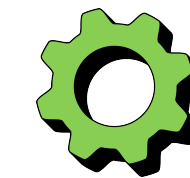
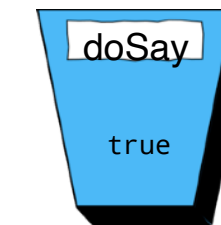
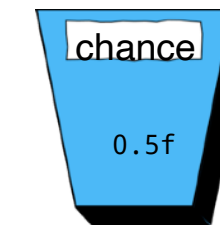
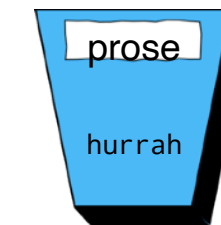


defining a function with
arguments and return value



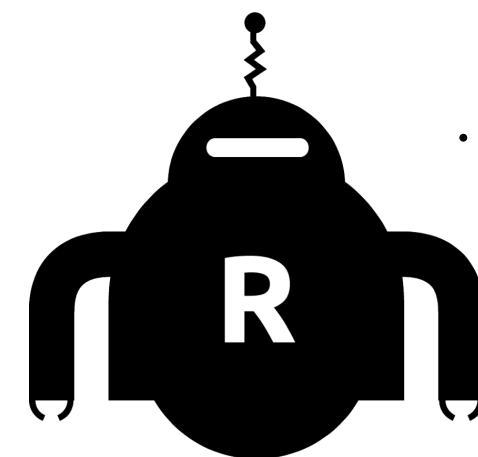


```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

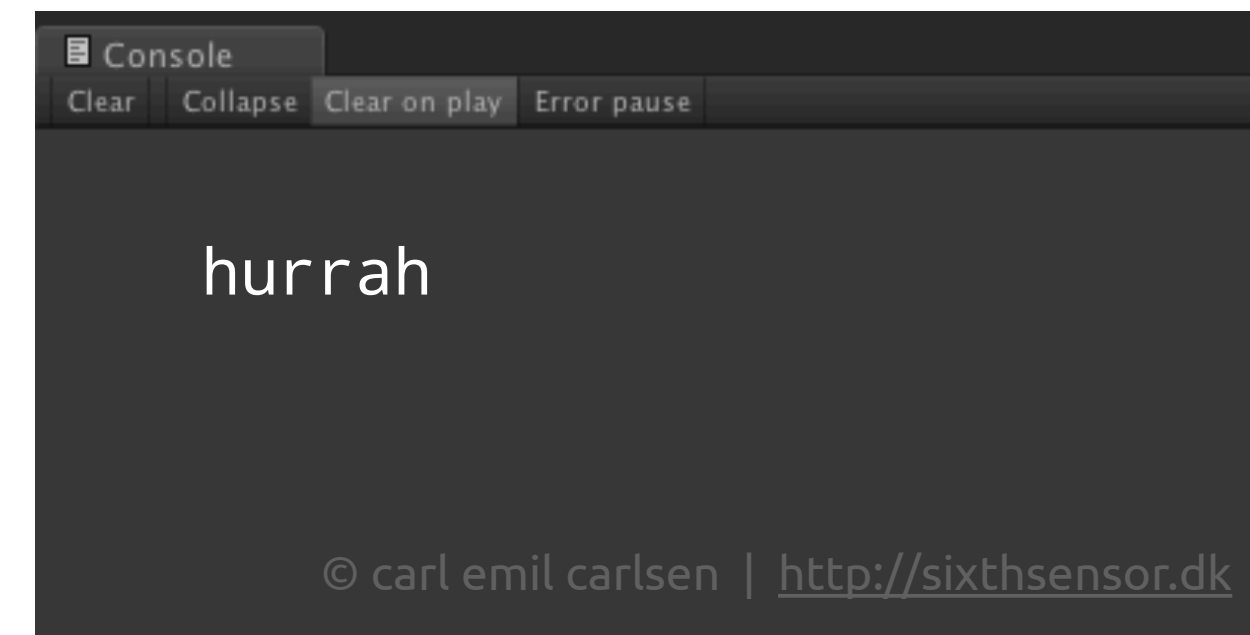


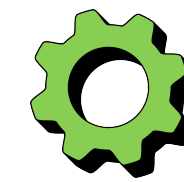
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

returning to where
I came from with
the value of the
variable 'doSay'

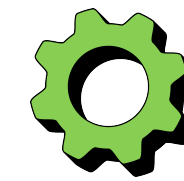
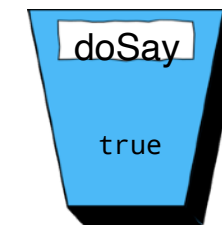
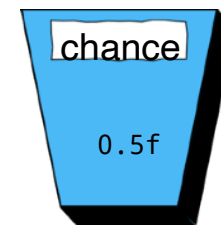
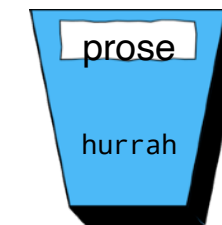


defining a function with
arguments and return value

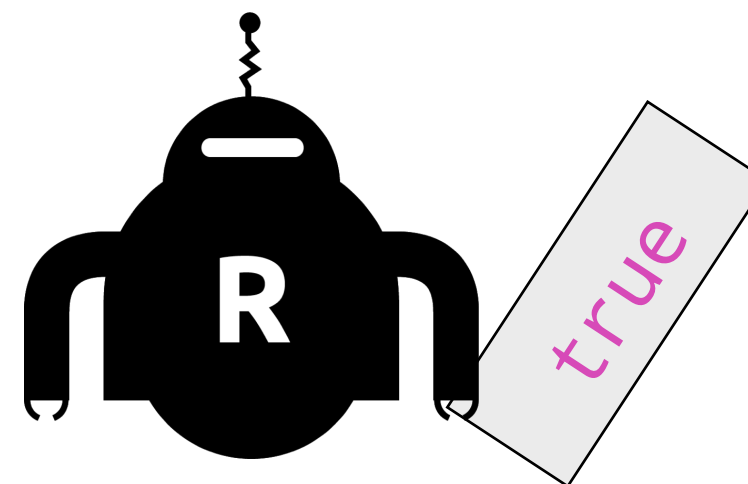




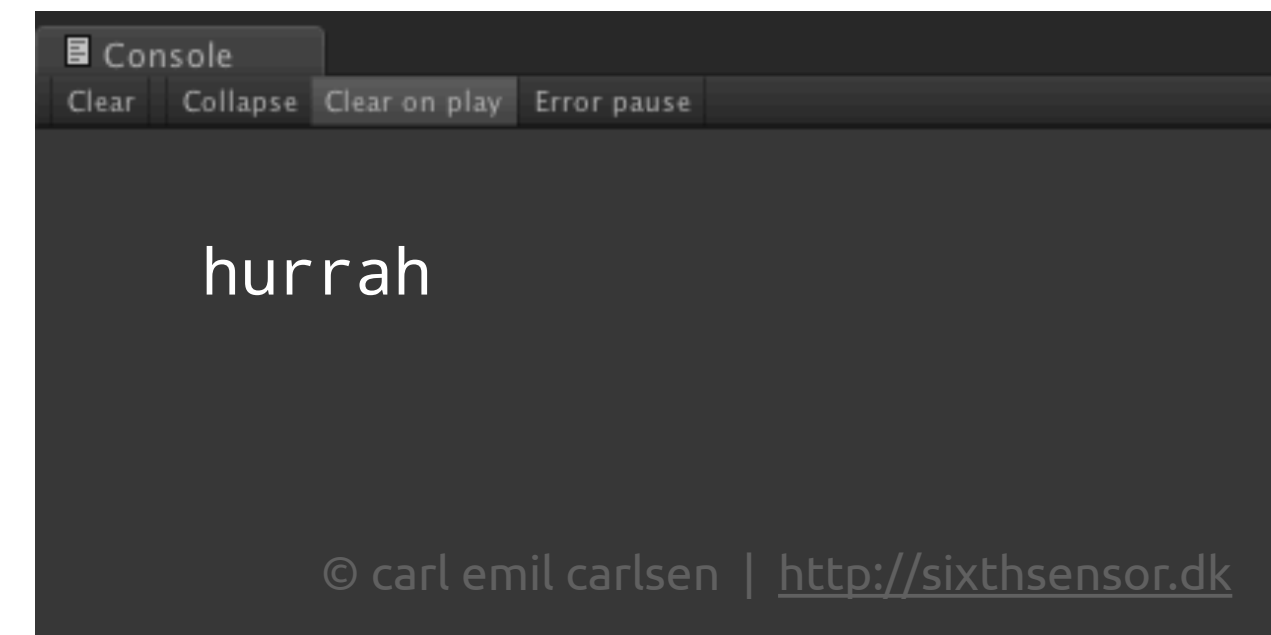
```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

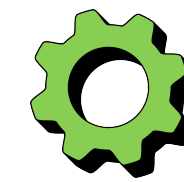


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

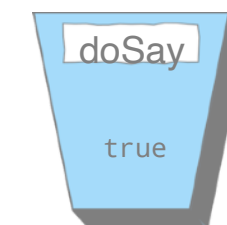
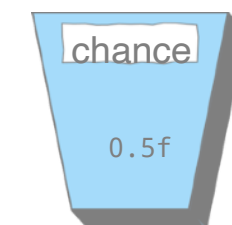
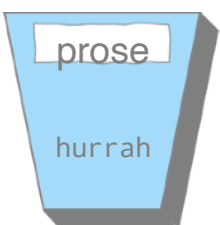
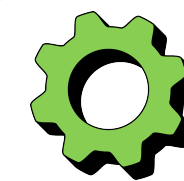
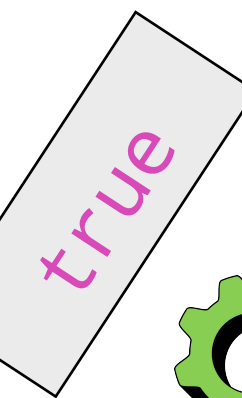
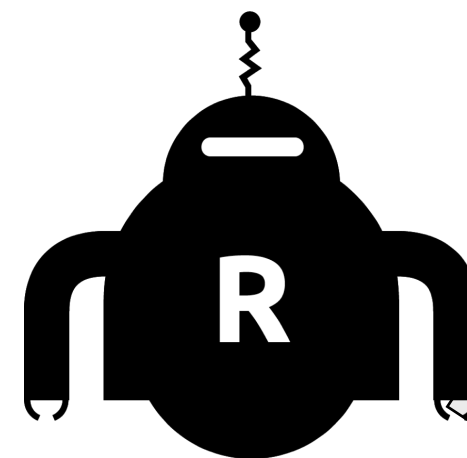


defining a function with
arguments and return value





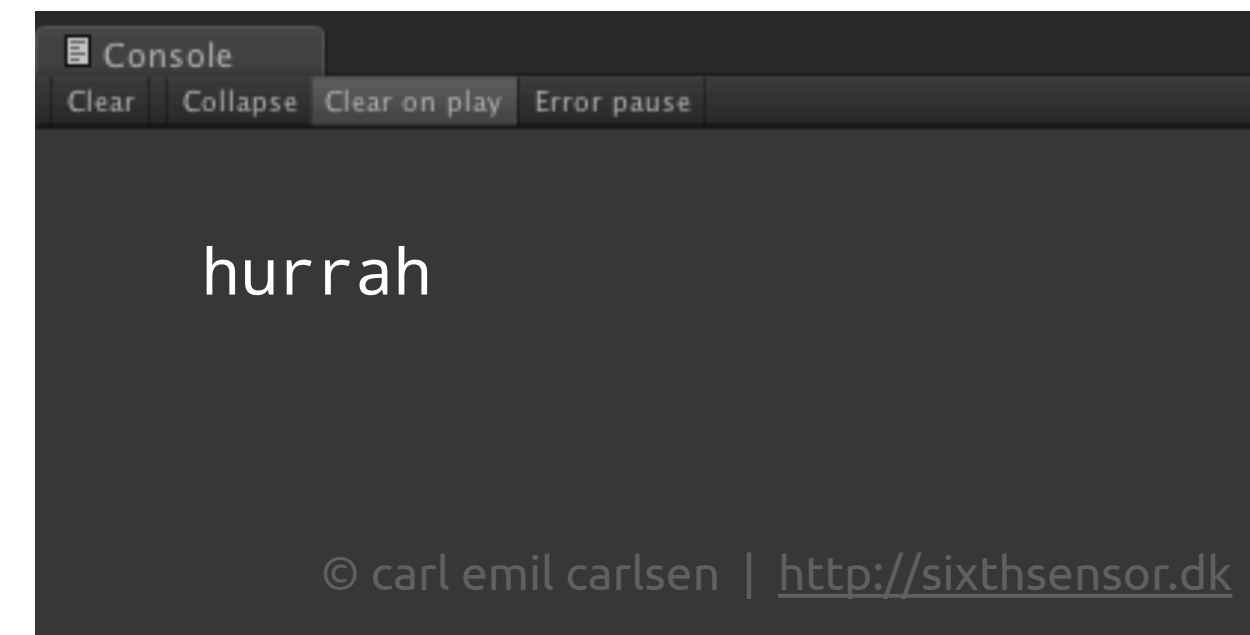
```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

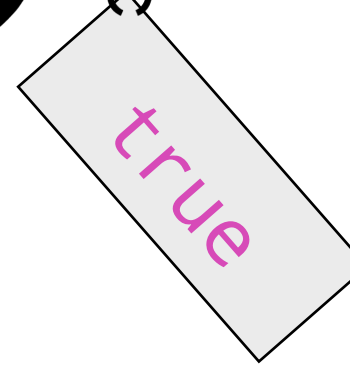
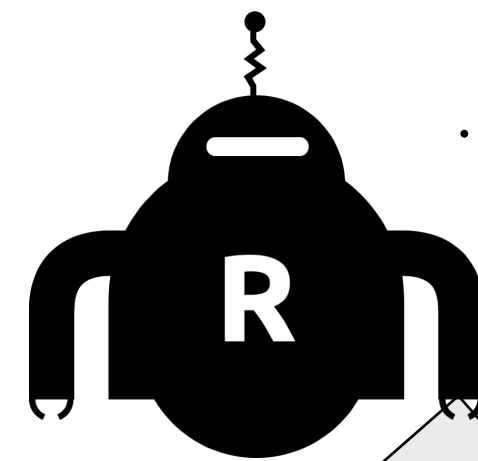


```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

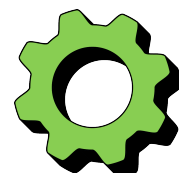
forgetting the
local variables in
the function
MaybeSay

defining a function with
arguments and return value





placing value

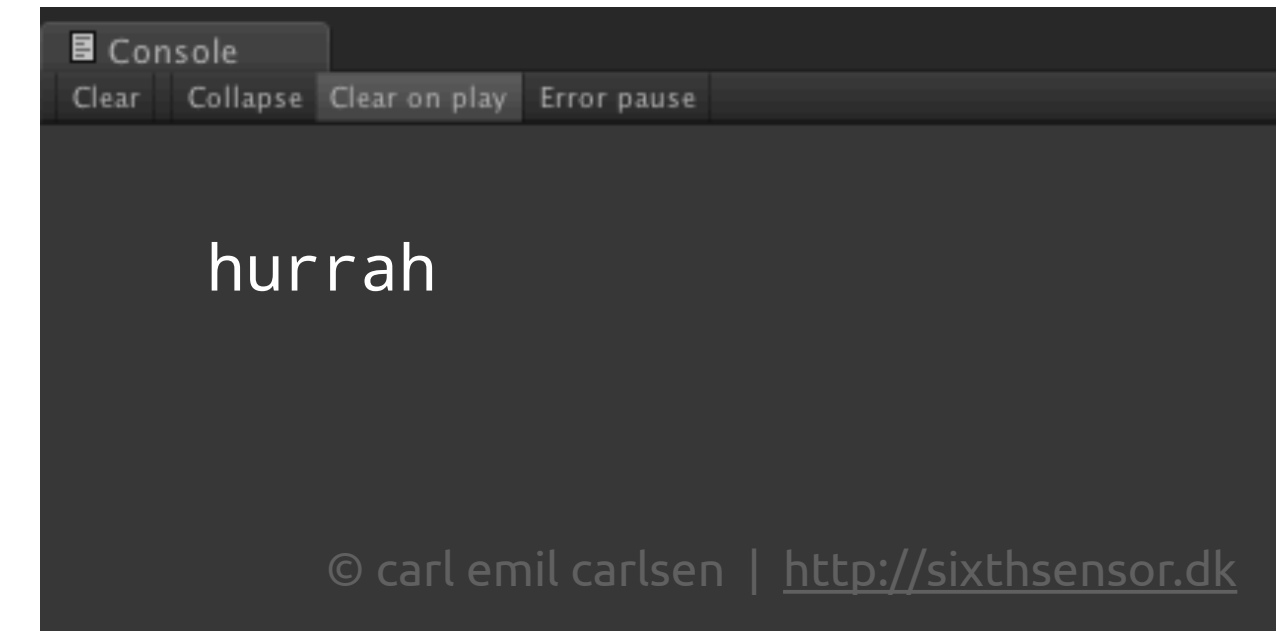


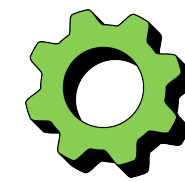
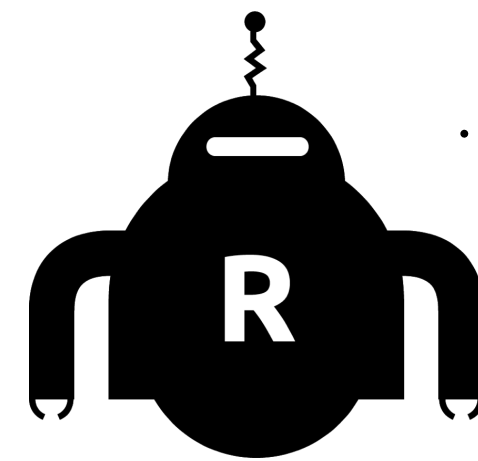
```
void Start(){  
    bool didSay = MaybeSay( "hurrah", 0.5f );  
    Debug.Log( "did say hurrah: " + didSay );  
}
```



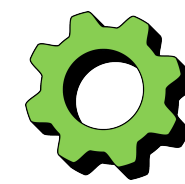
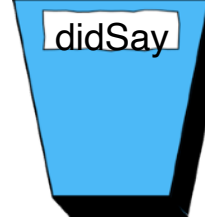
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with arguments and return value



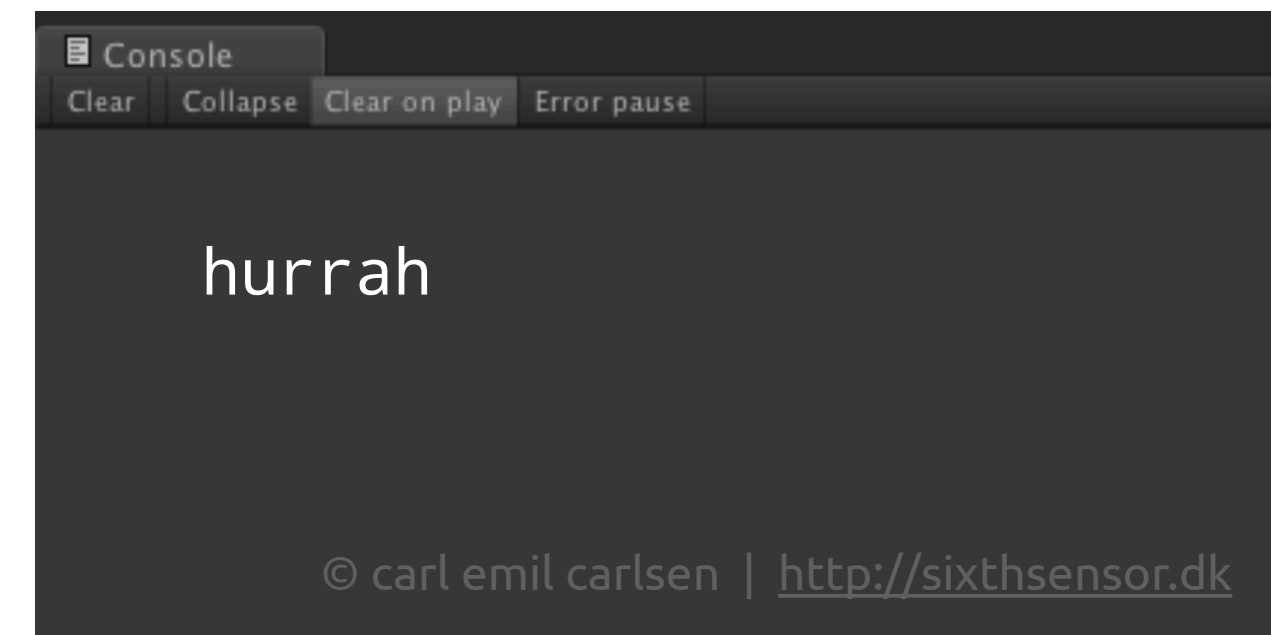


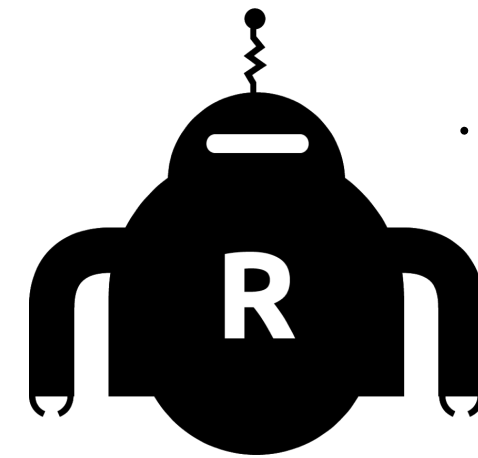
```
void Start(){  
  bool didSay = true;  
  Debug.Log( "did say hurrah: " + didSay );  
}
```



```
bool MaybeSay( string prose, float chance ){  
  bool doSay = Random.value < chance;  
  if(doSay){  
    Debug.Log(prose);  
  }  
  return doSay;  
}
```

defining a function with
arguments and return value





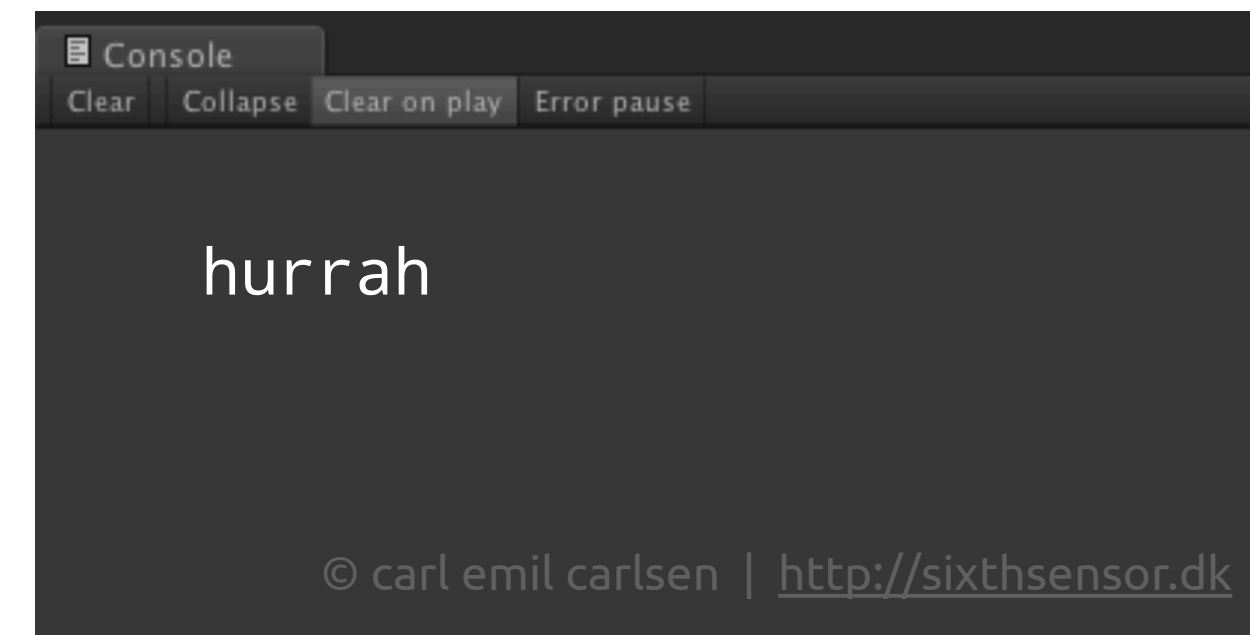
```
void Start(){  
    bool didSay = true;  
    Debug.Log( "did say hurrah: " + didSay );  
}
```

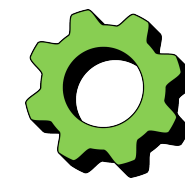
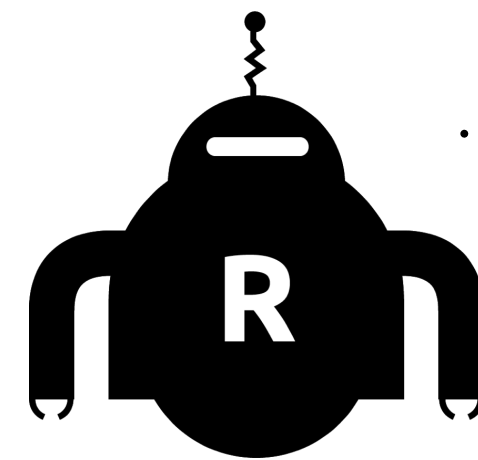
setting the variable
'didSay' to true



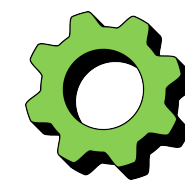
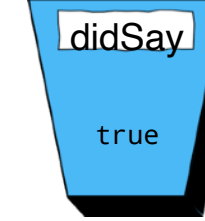
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value



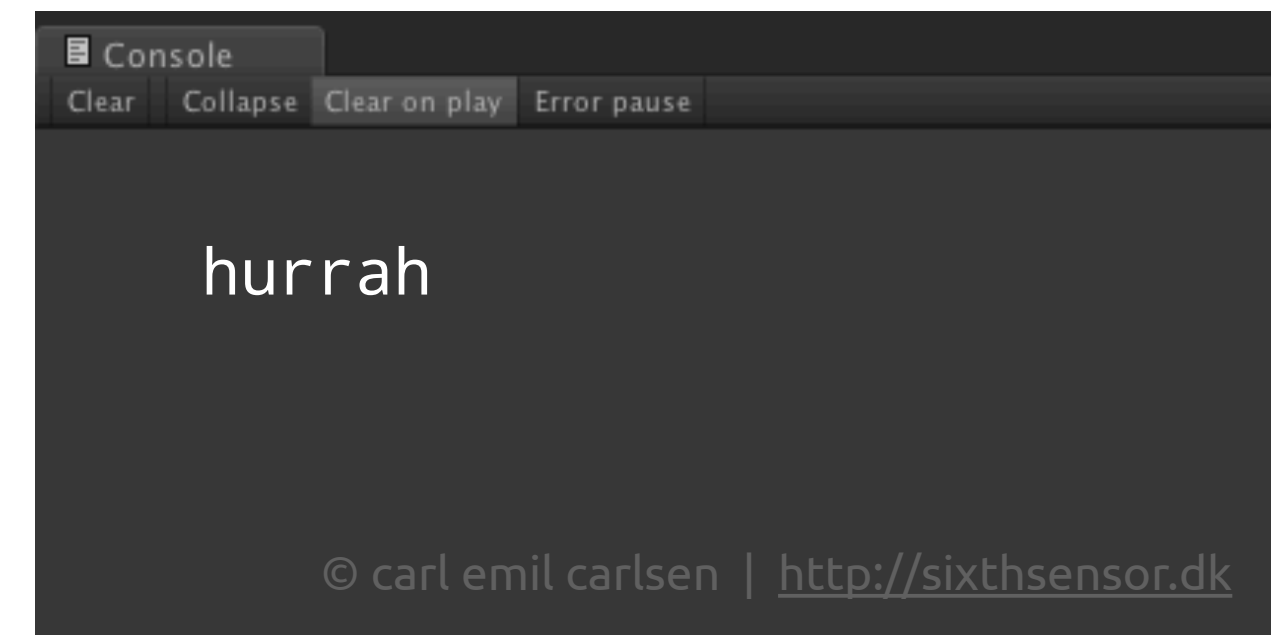


```
void Start(){  
    bool didSay = true;  
    Debug.Log( "did say hurrah: " + didSay );  
}
```



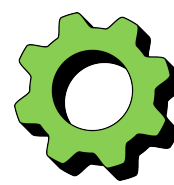
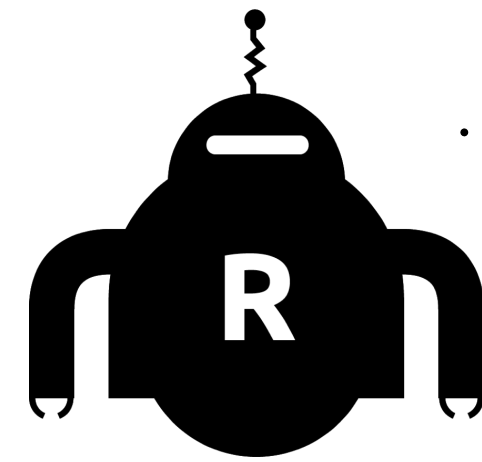
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value





getting value of the variable 'didSay'

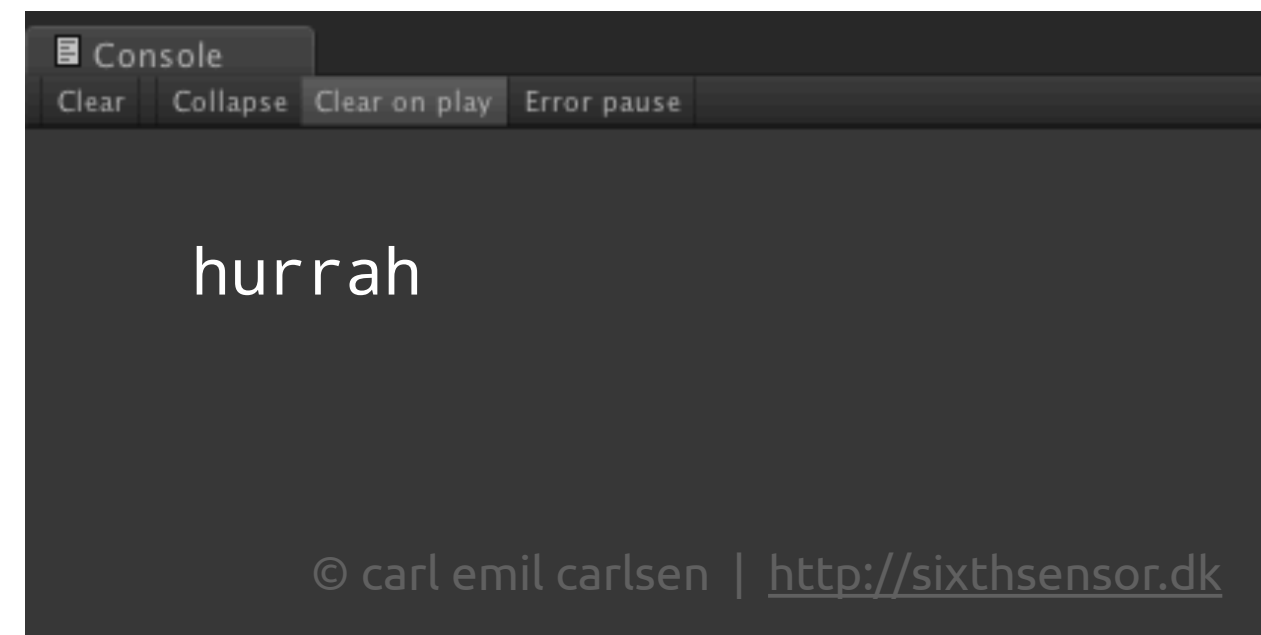


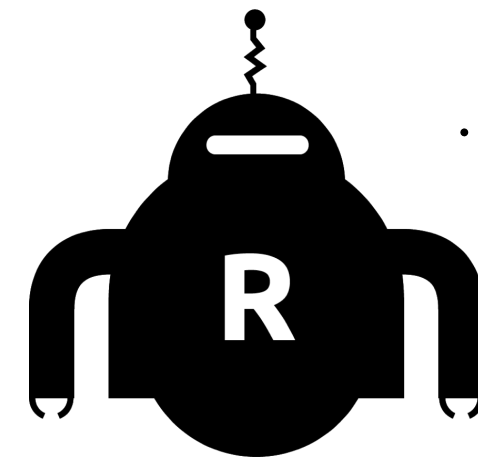
```
void Start(){  
    bool didSay = true;  
    Debug.Log( "did say hurrah: " + didSay );  
}
```



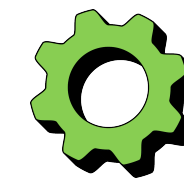
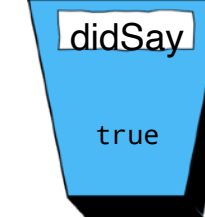
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with arguments and return value

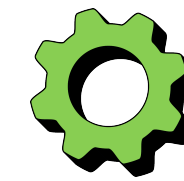




converting
boolean value
to string value

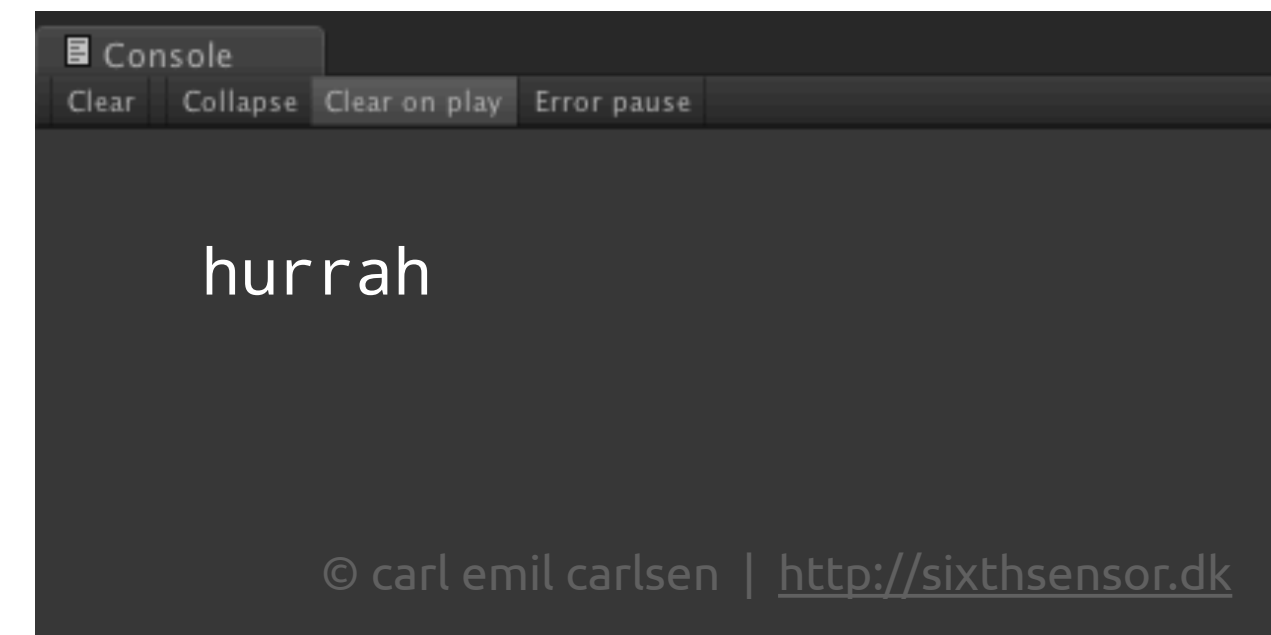


```
void Start(){  
    bool didSay = true;  
    Debug.Log( "did say hurrah: " + true );  
}
```



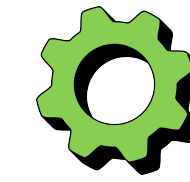
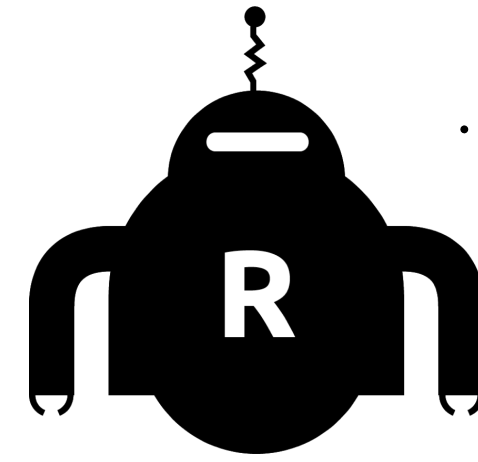
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value

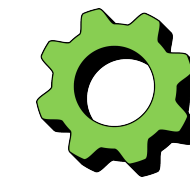
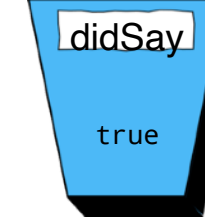




combining two
string values

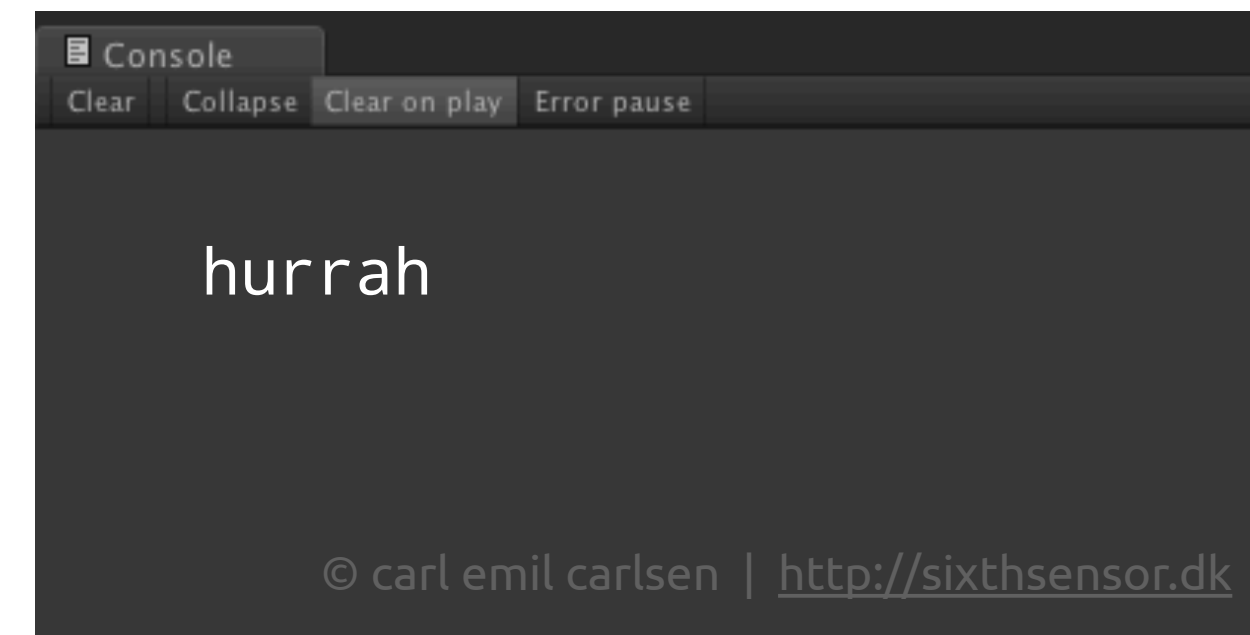


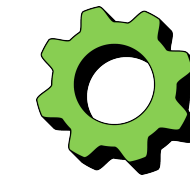
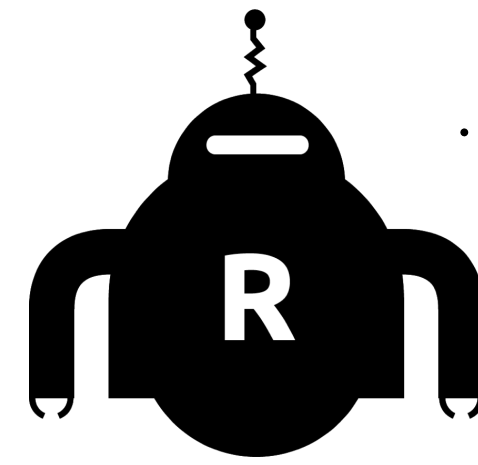
```
void Start(){  
    bool didSay = true;  
    Debug.Log( "did say hurrah: " + "true" );  
}
```



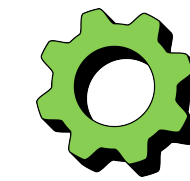
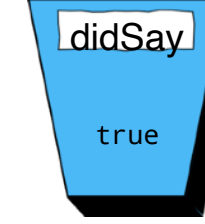
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value



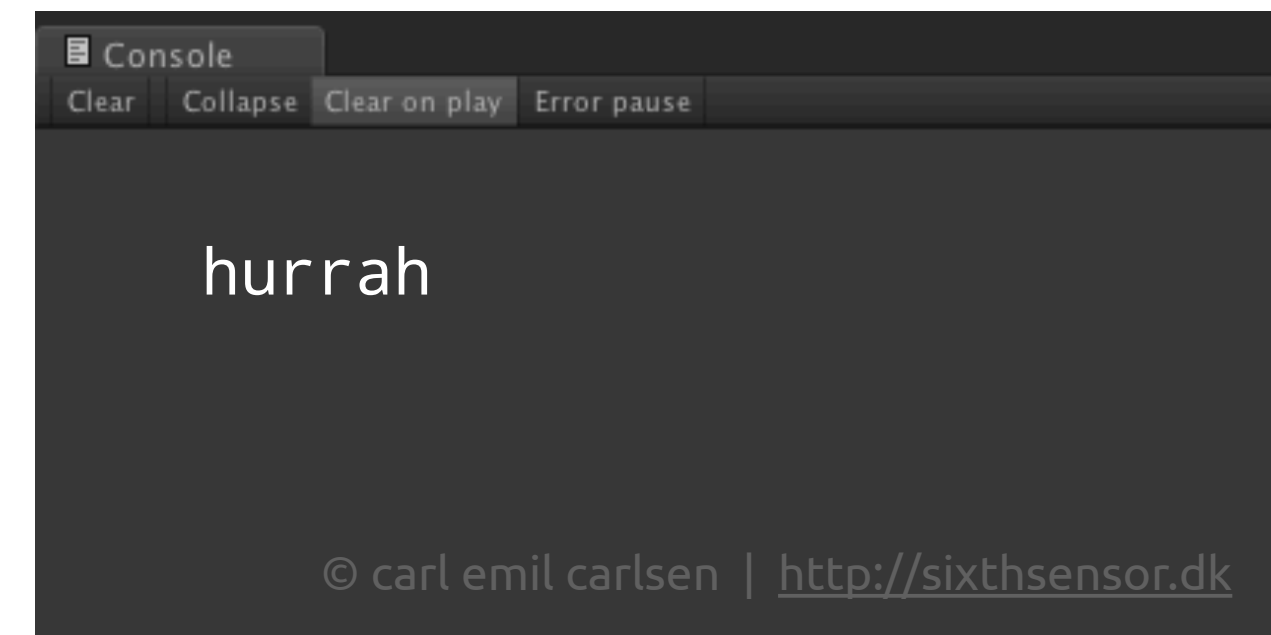


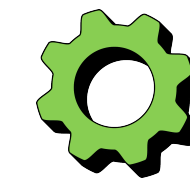
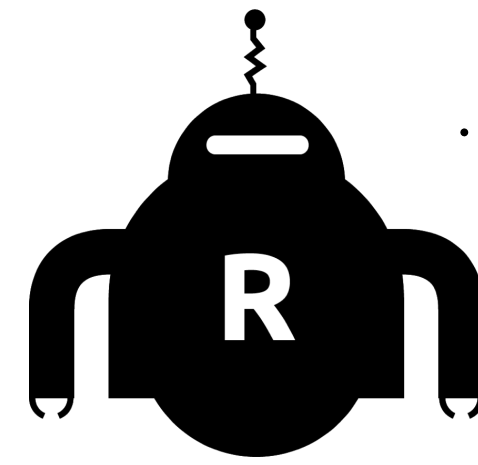
```
void Start(){  
    bool didSay = true;  
    Debug.Log( "did say hurrah: true" );  
}
```



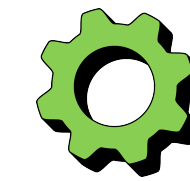
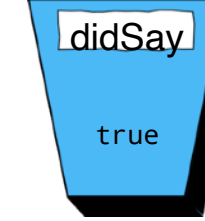
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value



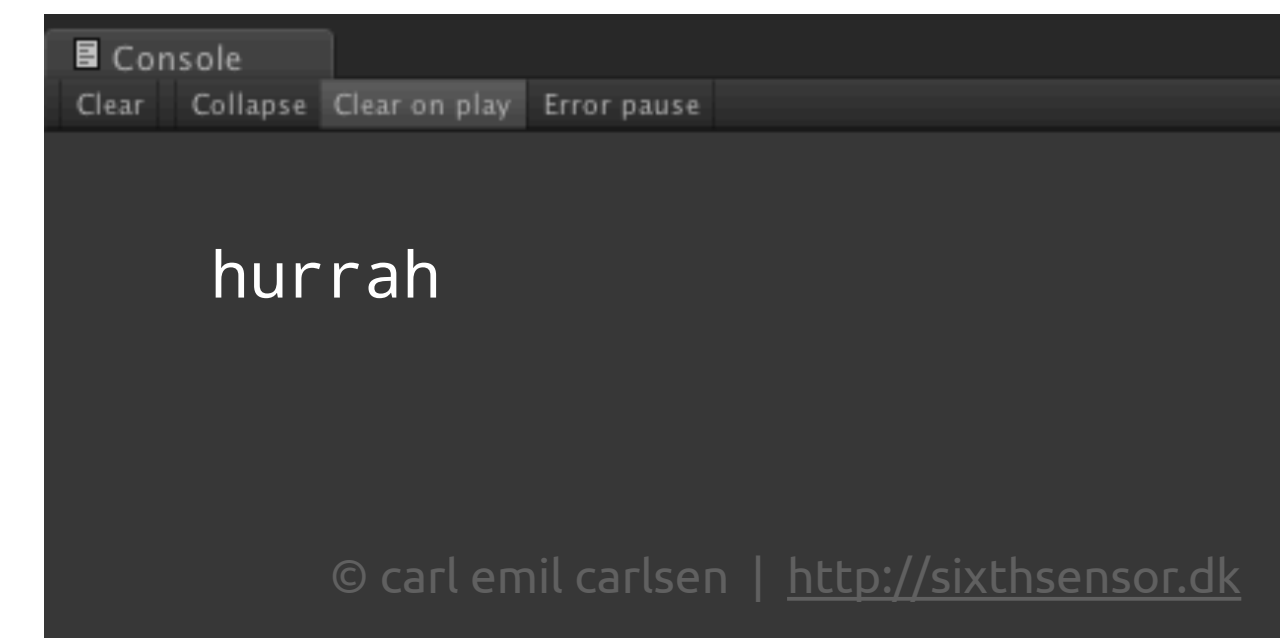


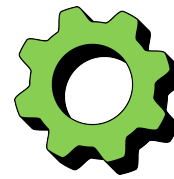
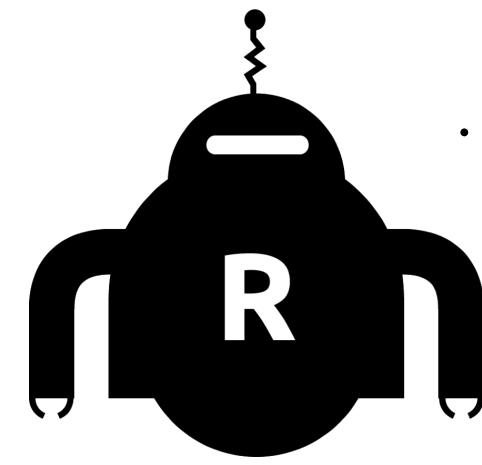
```
void Start(){  
    bool didSay = true;  
    Debug.Log( "did say hurrah: true" );  
}
```



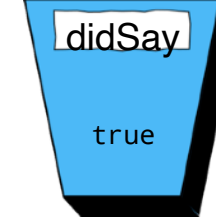
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value



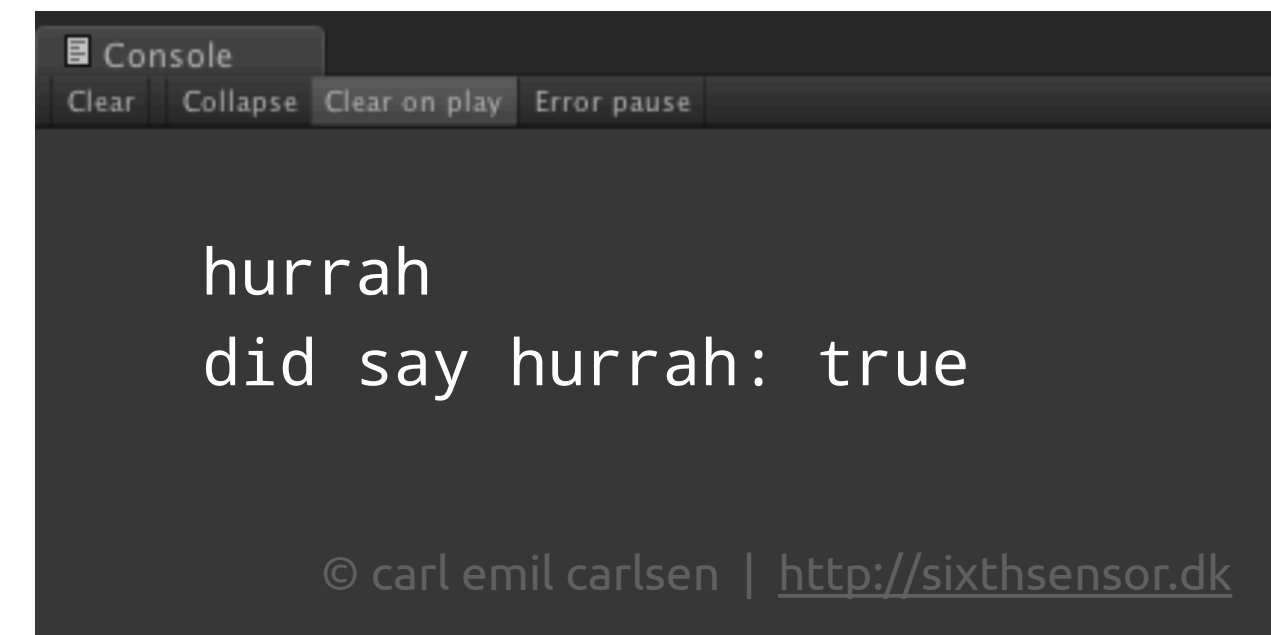


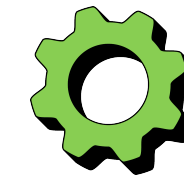
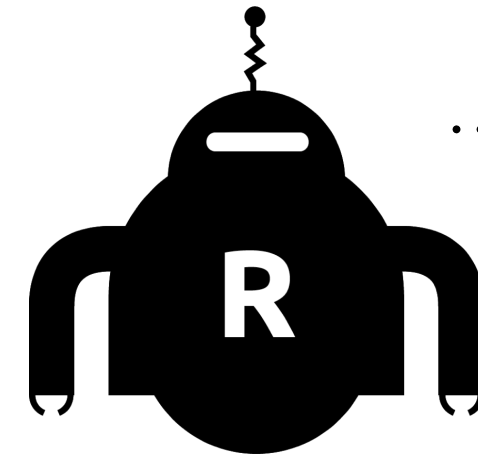
```
void Start(){  
    bool didSay = true;  
    Debug.Log( "did say hurrah: true" );  
}
```



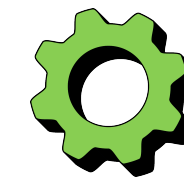
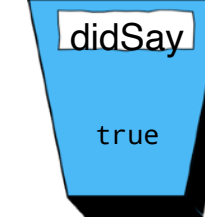
```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value





```
void Start(){  
    bool didSay = true;  
    Debug.Log( "did say hurrah: " + didSay );  
}
```



```
bool MaybeSay( string prose, float chance ){  
    bool doSay = Random.value < chance;  
    if(doSay){  
        Debug.Log(prose);  
    }  
    return doSay;  
}
```

defining a function with
arguments and return value

