For Loops in Arduino

ME 120

Mechanical and Materials Engineering
Portland State University
http://me120.mme.pdx.edu/

Motivation

- "for" loops are essential code constructs
  - Allow for repetition of code blocks
  - Specify the number of repetitions in advance
  - Loop counters can be used as indices in arrays

- Common applications
  - Sweep a servo back and forth
  - Compute average of an analog reading
  - Step through all elements of an array

Typographical convention used in these slides:

We write "for" and "while" and "loop" in quotation marks when we are referring to the "for" loop structure, the "while" loop structure and the required "loop" function in Arduino sketches. Excessive use of quotes is generally considered bad form. We risk offending in order to distinguish "for" as a code construct and the ordinary word (for) as regular preposition.

Reference

- Official Arduino web site
  - http://arduino.cc/en/Reference/for
  - http://arduino.cc/en/Tutorial/ForLoop

- ME 120 reading from the MME 120 wiki
  - “for loops in Arduino”
  - “What’s this void loop thing?”

http://me120.mme.pdx.edu/doku.php?id=topics:arduino_programming

But first ...
let’s review the structure of Arduino code

Three parts of a basic Arduino sketch

```
//  File:  blink.ino
//
//  Turns on an LED on for one second,
//  then off for one second, repeatedly.

int LED_pin = 11;
```

Header:
- Overview comments
- Global variables

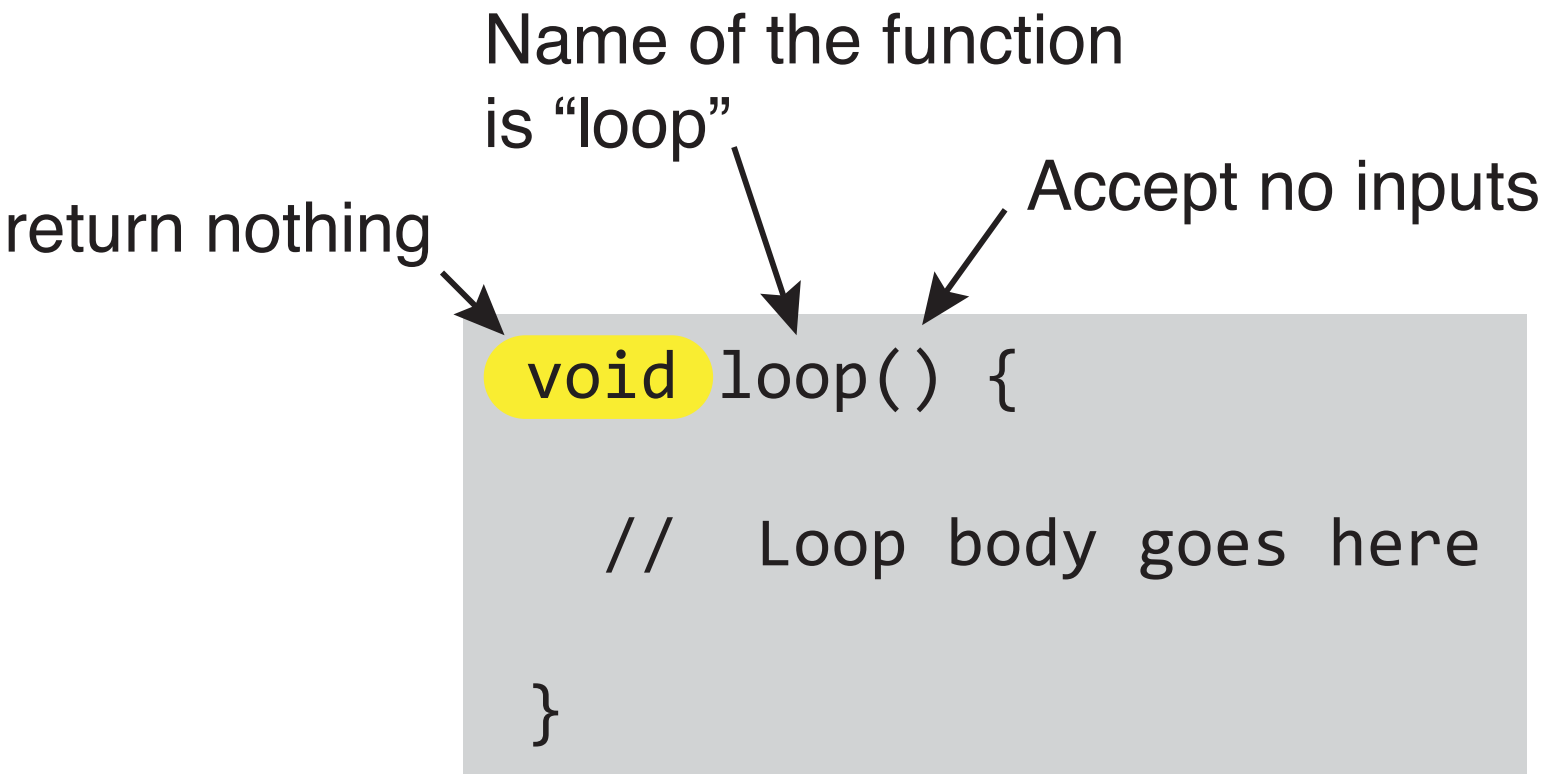
```
void setup() {
  pinMode(LED_pin, OUTPUT);
}
```

Setup:
- Execute only once
- Tasks for start-up

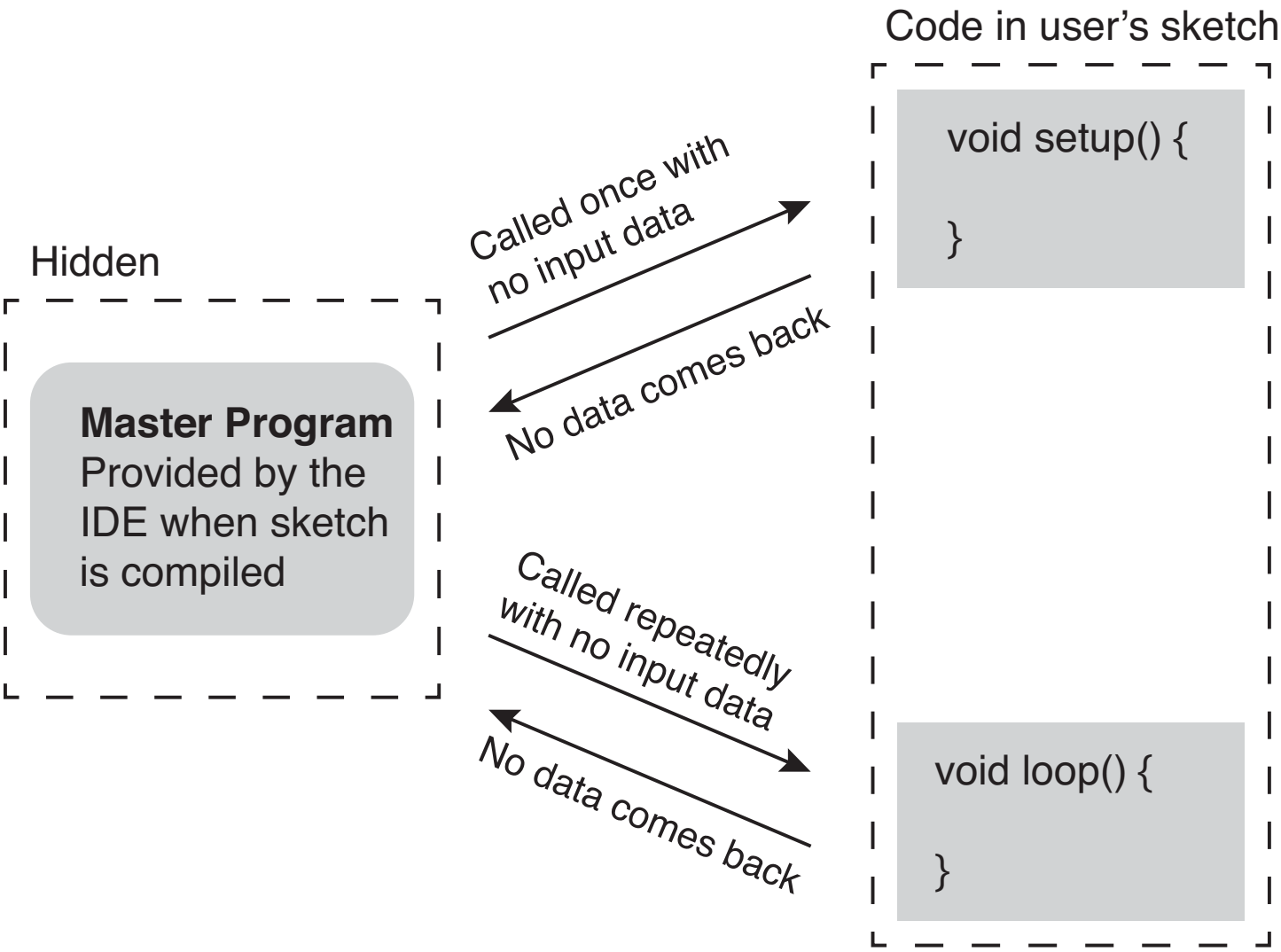
```
void loop() {
  digitalWrite(LED_pin, HIGH);
  delay(1000);
  digitalWrite(LED_pin, LOW);
  delay(1000);
}
```

Loop:
- Execute repeatedly
- Primary tasks for the sketch

“loop” accepts no inputs and returns no values



The hidden master calls “loop”



What does this mean?

The “loop” function is the main part of the Arduino sketch that is repeated indefinitely

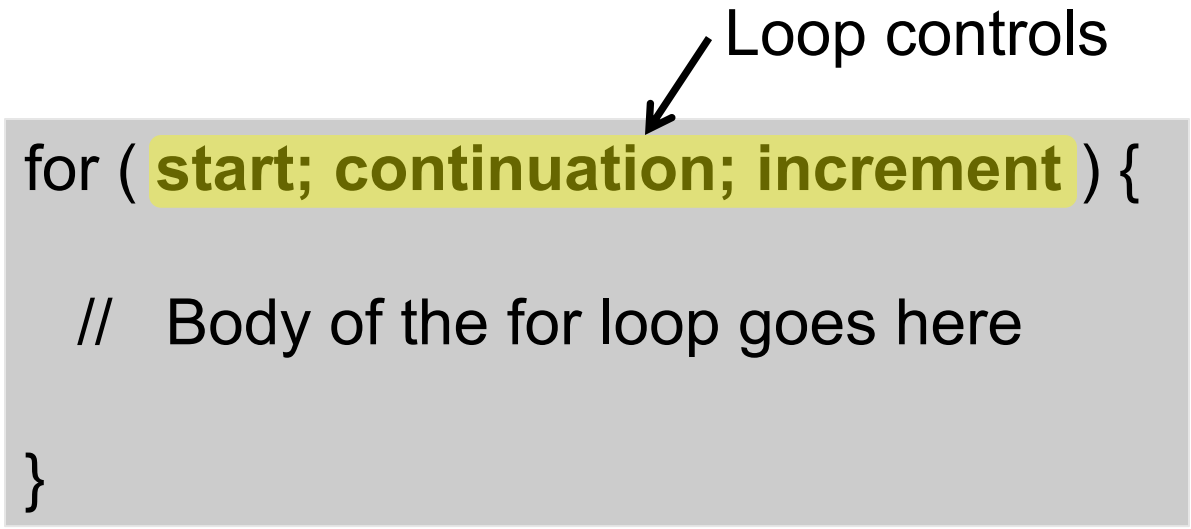
- “loop” is a function that is necessary for all Arduino sketches.
- “loop” may contain “for” loops and “while” loops

If nothing in your sketch needs to be repeated, you can leave the body of “loop” empty.

- For example, we use an empty “loop” function when we are experimenting with code snippets. In that case, all the code would then be entered in the setup function.
- In most Arduino programs, the “loop” function is not empty. It’s where most of the work of the sketch is performed.

“for” loops

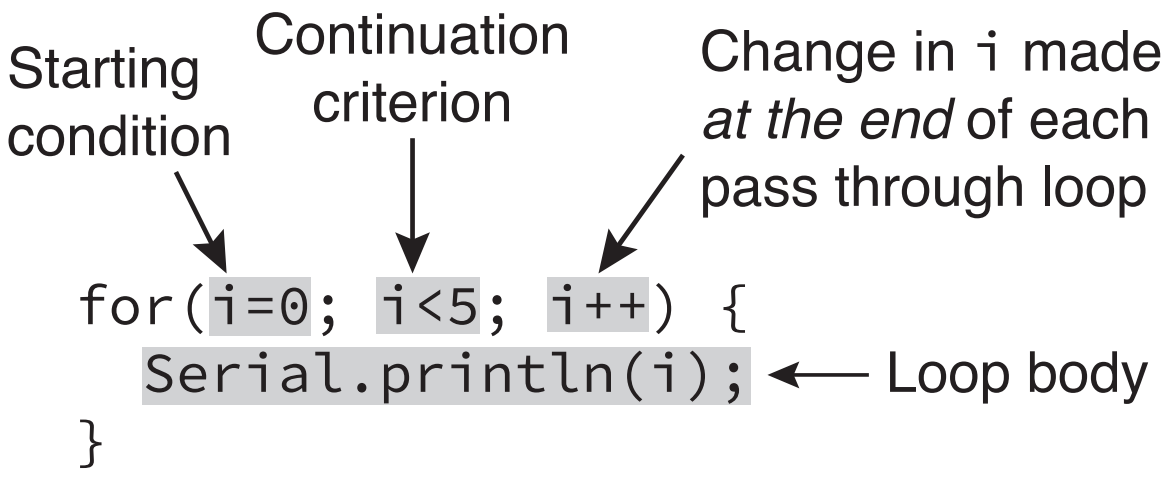
Basic “for” loop structure



Three parts to the loop controls:

- **Starting condition**: set the initial value of the loop counter
- **Continuation criterion**: a logical test that must be true for the loop to execute
- **Increment/decrement**: the rule for changing the loop counter

Example



- **`i=0`** is the starting condition. It establishes the value of **`i`** on the first (and only the first) trip through the loop
- **`i<5`** must be true for the loop body to be executed
- **`i++`** means that the value of **`i`** is incremented by one *after* the loop body is executed.

Example: add up 4 integers

Compute: $s = \sum_{i=1}^{4} i$

```
int i, sum;

sum = 0;
for ( i=1; i<=4; i++) {
    sum = sum + i;
}
```

Example: add up 4 integers

Compute: $s = \sum_{i=1}^{4} i$

```
int i, sum;

sum = 0;
for ( i=1; i<=4; i++) {
    sum = sum + i;
}
```

Walk through the code

`sum`	`i`	`i<=4`	Comment
0	N.A.	N.A.	`i` not defined before the loop starts
0	1	True	Condition at start of first pass
1	1		Condition at end of first pass
1	2	True	… start of second pass
3	2		… end of second pass
3	3	True	… start of third pass
6	3		… end of third pass
6	4	True	… start of fourth pass
10	4		… end of fourth pass
10	5	False	`i` fails stopping test, jump to first statement after the loop.

Common increment expressions

Expression	Meaning
`i++`	Increment by 1 at end of the loop
`i--`	Decrement by 1 at end of the loop
`i+=1`	Same as `i++`
`i-=1`	Same as `i--`
`i+=2`	Increment by 2 at end of the loop
`i-=2`	Decrement by 2 at end of the loop
`++i`	Increment by 1 at beginning of the loop
`--i`	Decrement by 1 at beginning of the loop

Loop 1: What does this code do?

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int i = 0;
  i = i + 1;
  Serial.println(i);
}
```

Loop 2: What does this code do?

```
int i = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  i = i + 1;
  Serial.println(i);
}
```

Loop 3: What does this code do?

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int i;
  for ( i=0; i<5; i++ ) {
    Serial.println(i);
  }
}
```

Loop 4: What does this code do?

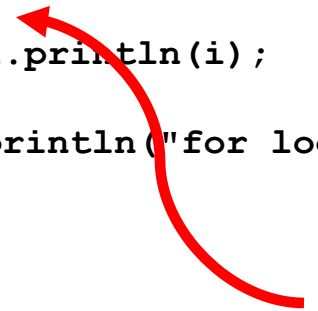
```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int i;
  for ( i=0; i<5; i+=2 ) {
    Serial.println(i);
  }
}
```

Loop 5: What does this code do?

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int i;
  for ( i=0; i<10; i++) {
    i = 5;
    Serial.println(i);
  }
  Serial.println("for loop over\n");
}
```



Don’t do this!
Never change the loop counter in the body of the loop!

Loop 6: What does this code do?

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int i;
  for ( i=0; i<10; i++) {
    Serial.println(i);
    delay(100);
  }
  Serial.println("for loop over\n");
}
```