CS 100

Methods

Why use methods?

- We want to write a program that prints out different messages on the screen,
- and we would like to display the following pattern in-between messages to separate them

```
****************

****************
```

Solution approach

- Use two fprintf

```
fprintf('***************\n');
fprintf('***************\n');
```

The structure of the code..

```
%% produce some output
...
fprintf('***************\n');
fprintf('***************\n');
%% produce some other output
...
fprintf('***************\n');
fprintf('***************\n');
%% produce even more output
...
fprintf('***************\n');
fprintf('***************\n');
%% produce the final output
...
```

Anything wrong with this?

The problem

- The solution is fine
  - It does produce the desired result
  - No issues with respect to correcness and functionality

- But, there is an issue from a different perspective
  - How hard it would be to change the program in the future
  - How much work is it to write the same statements over and over
  - ...

What if..

- Later on the client wants us to change
  - The number of rows of stars
  - The number of stars per row
  - Use another character than a star
  - Print the date and time with each seperator
  - ...

- How much work is involved?

If we want to change anything

- Have to edit every *<u>copy</u>* of the code in the program
- It is easy to overlook some copies
- It can be hard to find them all
  - They might not be written identically
- A piece of code that looks like serving the same purpose might be doing something else

The <u>Big Idea</u> behind methods

- Identify a sub-problem that has to be solved
- Write code for solving that sub-problem, only once
- Give that code a name: that makes it a method
- Whenever the sub-problem needs to be solved, use the method name to say:
    - Go to that code now to take care of this sub-problem, and do not come back until you are done!

Example:

- Identify a sub-problem that has to be solved
- Take the repeated lines of code

```
fprintf('***************\n');

fprintf('***************\n');
```

- Wrap it as a method by giving it a name, e.g., *printSeparator*

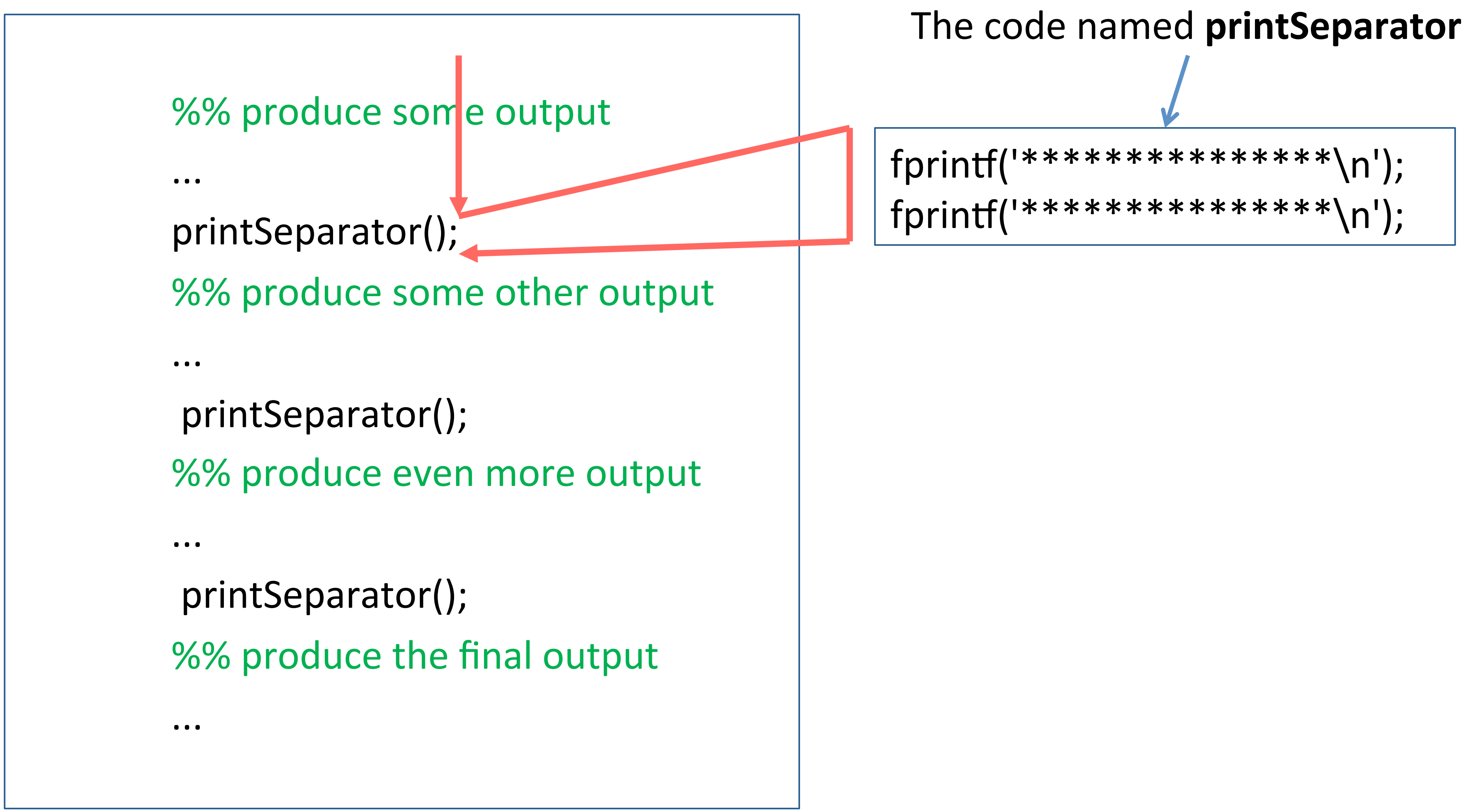
Example:

```
%% produce some output
...
printSeparator()
%% produce some other output
...
 printSeparator()
%% produce even more output
...
 printSeparator()
%% produce the final output
...
```

The code named **printSeparator**

```
fprintf('***************\n');
fprintf('***************\n');
```

Example:

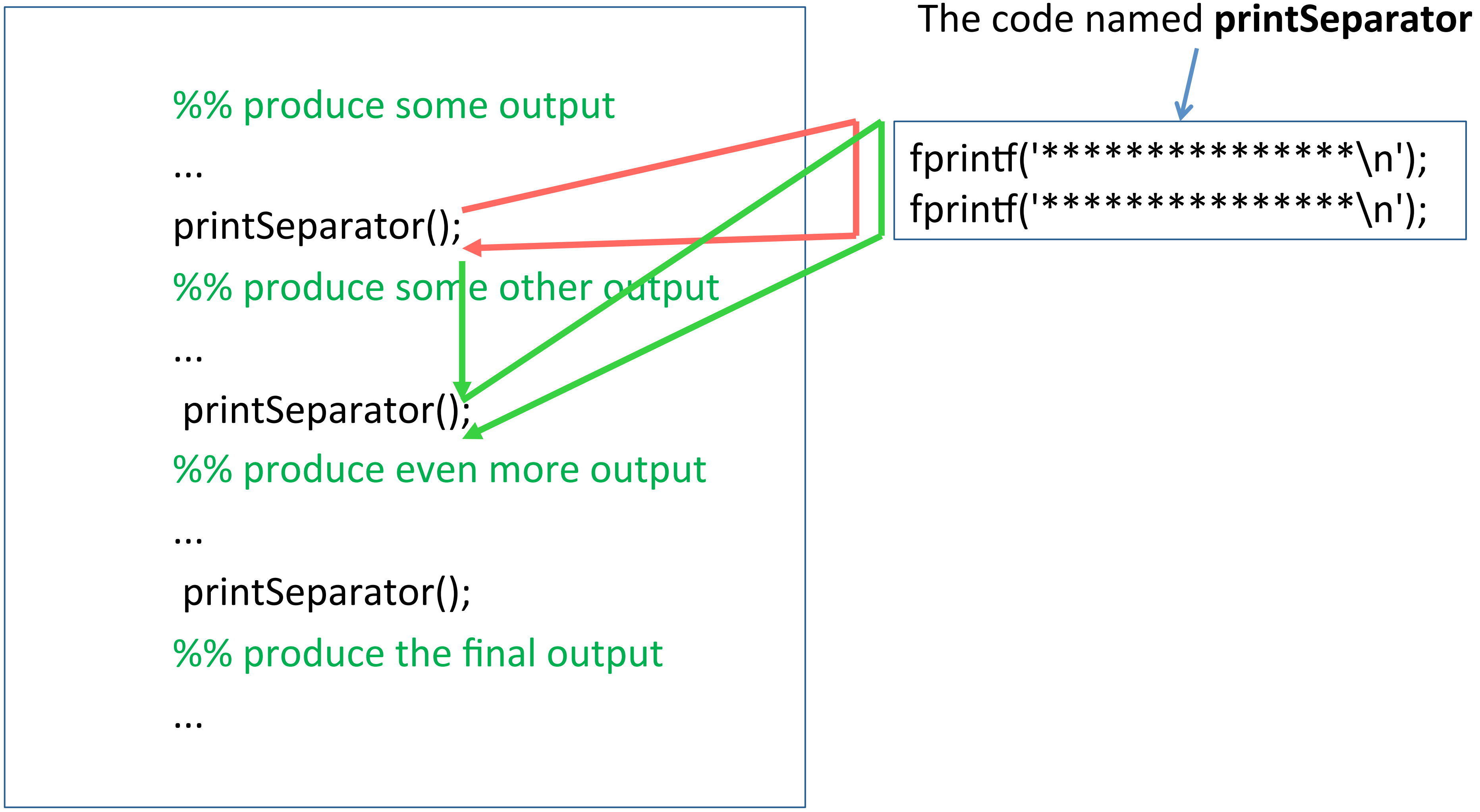


```
%% produce some output
...
printSeparator();
%% produce some other output
...
 printSeparator();
%% produce even more output
...
 printSeparator();
%% produce the final output
...
```

The code named **printSeparator**

```
fprintf('***************\n');
fprintf('***************\n');
```

Example:

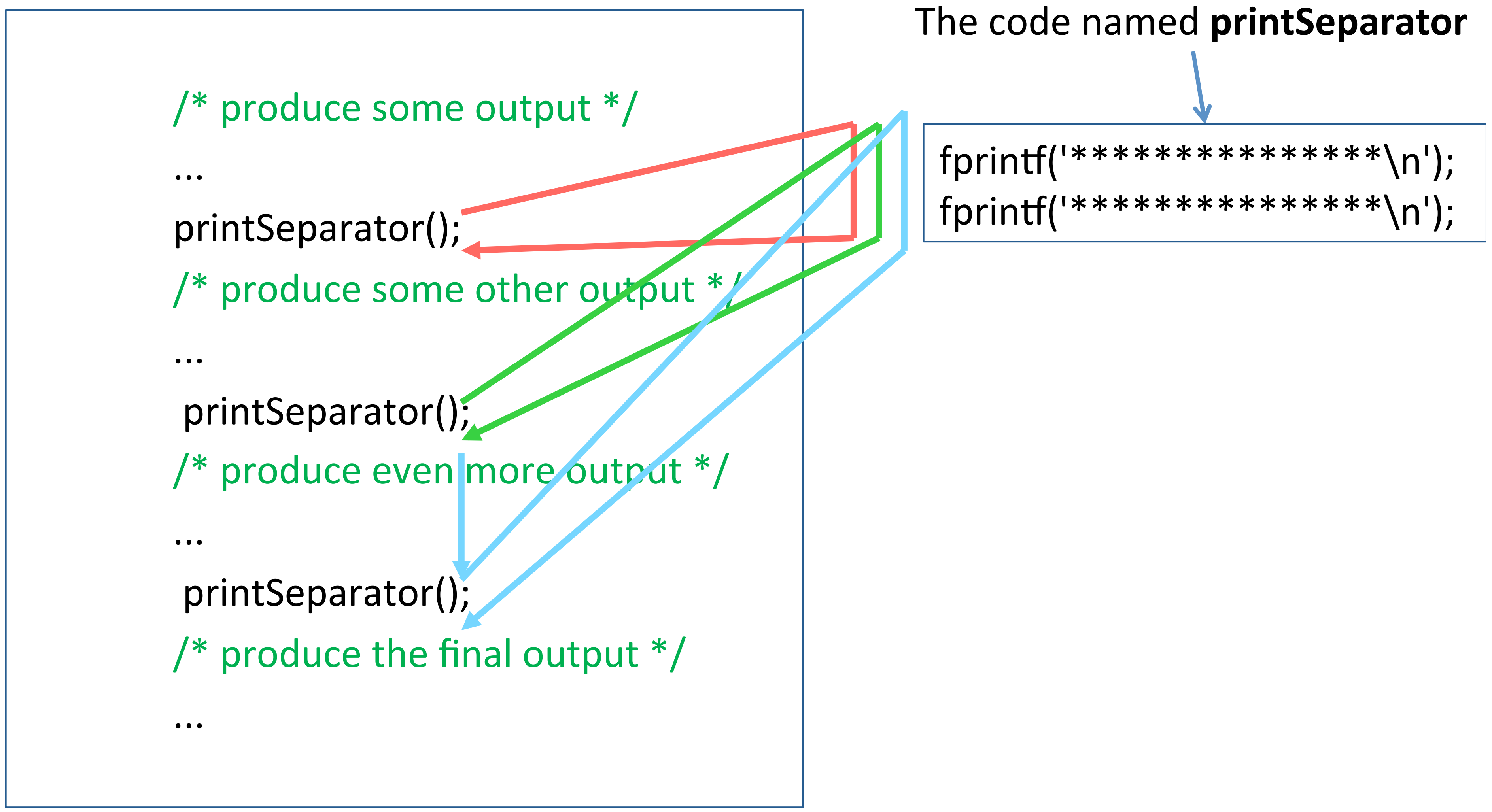


The code named **printSeparator**

```
fprintf('***************\n');
fprintf('***************\n');
```

```
%% produce some output
...
printSeparator();
%% produce some other output
...
 printSeparator();
%% produce even more output
...
 printSeparator();
%% produce the final output
...
```

Example:



```
/* produce some output */
...
printSeparator();
/* produce some other output */
...
 printSeparator();
/* produce even more output */
...
 printSeparator();
/* produce the final output */
...
```

The code named **printSeparator**

```
fprintf('***************\n');
fprintf('***************\n');
```

Example:

```
/* produce some output */
...
printSeparator();
/* produce some other output */
...
 printSeparator();
/* produce even more output */
...
 printSeparator();
/* produce the final output */
...
```

The code named **printSeparator**

```
fprintf('***************\n');
fprintf('***************\n');
```

Question:

If we need to change the separator in this new verison of the program,

What do we have to do?

How many places in the program have to be changed?

The Big Picture so far..

- Methods
- Method control flow
- The motivation for methods

- Next..
  - How to define and use methods in Matlab
  - Details of different type of methods and their usage

How to define methods

- The following is a typical method declaration

```
%% print a separator line on output
function [] = printSeparator()
     fprintf('***************\n');
     fprintf('***************\n');
end
```

How to define methods

- The following is a typical method declaration

heading comment

```
%% print a separator line on output
function [] = printSeparator()
     fprintf('***************\n');
     fprintf('***************\n');
end
```

method name

a method can have any number of and any kind of statements

Using a method..

```
printSeparator()
```

OR

```
printSeparator
```

New concepts

- Two new concepts that will be discussed later on..
  - **Return** values
  - **Parameters**
- The simple method in our example do not have any return value or a parameter

```
function [] = printSeparator()
          fprintf('***************\n');
          fprintf('***************\n');
end
```

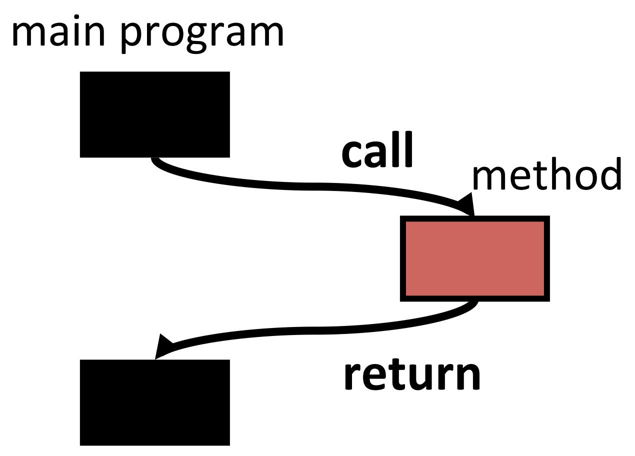
Some Java methods

- We have already seen and used several methods:

```
fprintf
find
sum
mod
diag
...
```

New Concepts

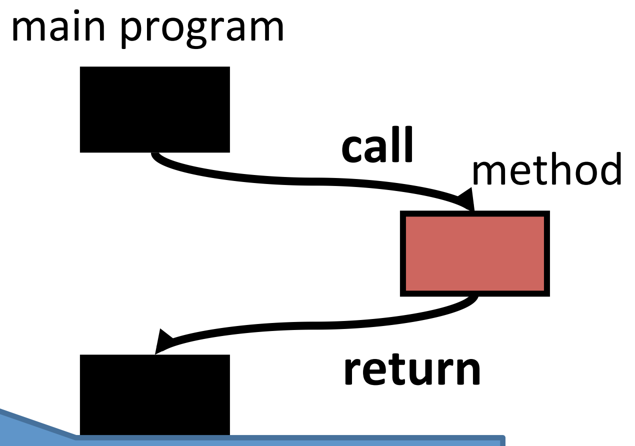
- Two new concepts:
  - **Return** values
  - **Parameters**
- The simple method in our example did not have any return value or a parameter



```
function [] = printSeparator()
        fprintf('***************\n');
        fprintf('**************\n');
end
```

New Concepts

- Two new concepts:
  - **Return** values
  - **Parameters**
- The simple method in our example did not have any return value or a parameter



First we will focus on this new concept

```
function [] = printSeparator()
        fprintf('***************\n');
        fprintf('***************\n');
end
```

Refresher: Printing a Separator

- The original program within the run method called this method to print separator lines

```
%% print a separator line on output
function [] = printSeparator()
     fprintf('***************\n');
     fprintf('***************\n');
end
```

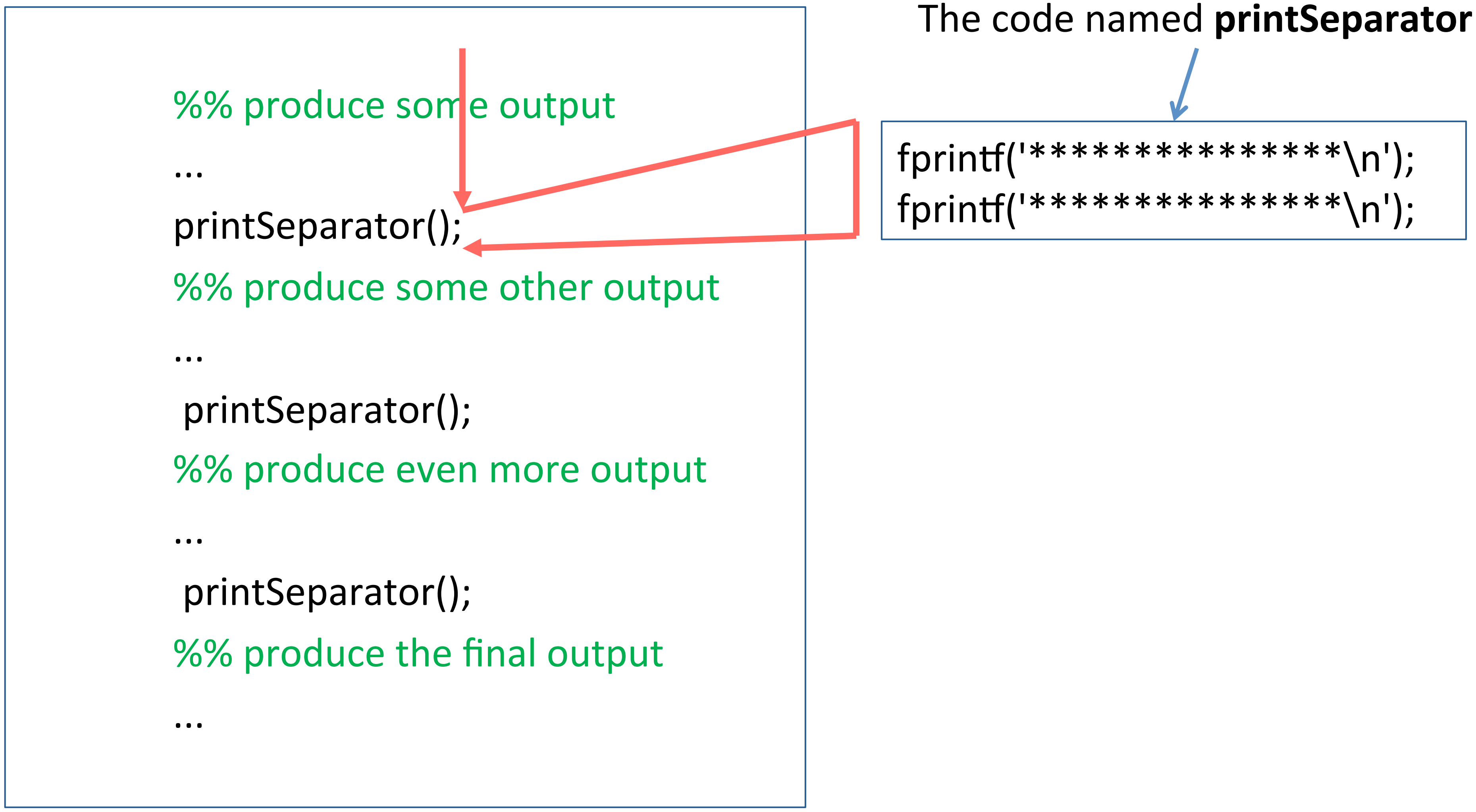
Refresher: Example

```
%% produce some output
...
printSeparator()
%% produce some other output
...
 printSeparator()
%% produce even more output
...
 printSeparator()
%% produce the final output
...
```

The code named **printSeparator**

```
fprintf('***************\n');
fprintf('***************\n');
```

Refresher: Example

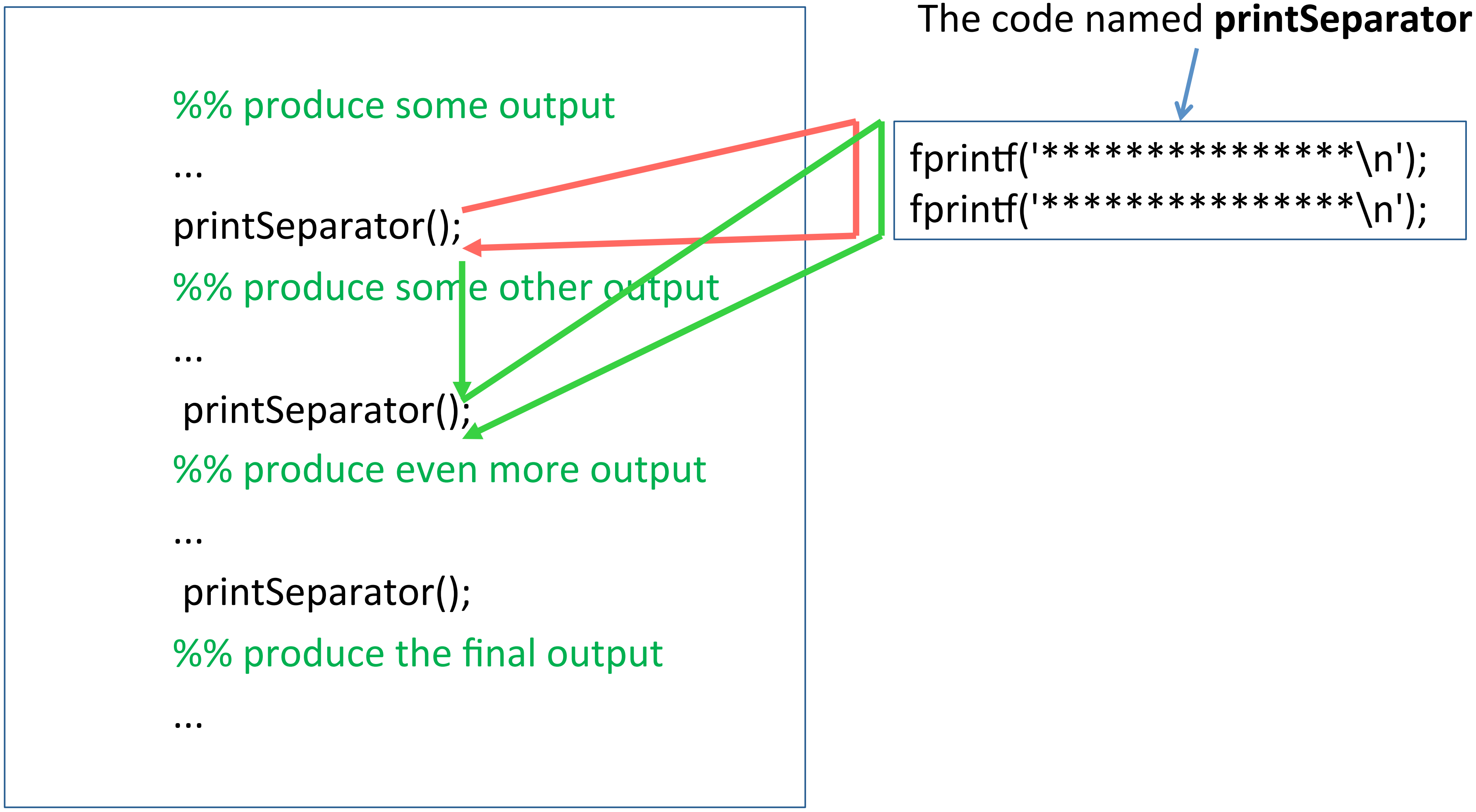


```
%% produce some output
...
printSeparator();
%% produce some other output
...
 printSeparator();
%% produce even more output
...
 printSeparator();
%% produce the final output
...
```

The code named **printSeparator**

```
fprintf('***************\n');
fprintf('***************\n');
```

Refresher: Example

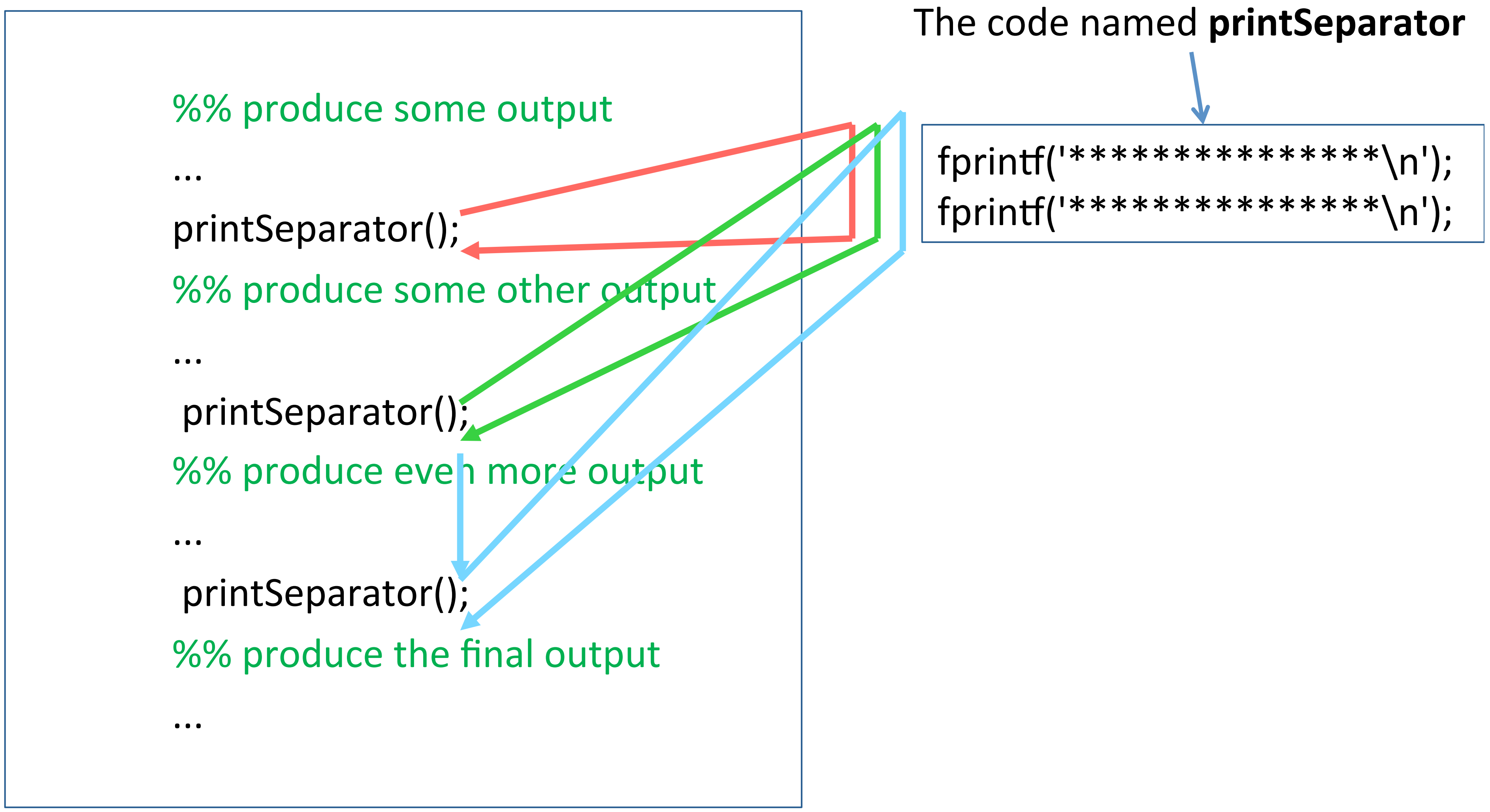


```
%% produce some output
...
printSeparator();
%% produce some other output
...
 printSeparator();
%% produce even more output
...
 printSeparator();
%% produce the final output
...
```

The code named **printSeparator**

```
fprintf('***************\n');
fprintf('***************\n');
```

Refresher: Example



```
%% produce some output
...
printSeparator();
%% produce some other output
...
 printSeparator();
%% produce even more output
...
 printSeparator();
%% produce the final output
...
```

The code named **printSeparator**

```
fprintf('***************\n');
fprintf('***************\n');
```

A new problem..

- The client wants another change
  - The program should print out 5 rows of stars when it starts and when it ends
  - But it should print out 2 rows of stars in-between messages

One possible solution

- Define two different methods for two different type of separators

```
%% print two lines of stars
function [] = printSeparator2Lines()
        fprintf('***************\n');
        fprintf('***************\n');
end


%% print 5 lines of stars
function [] = printSeparator5Lines()
        for i =1:5
                fprintf('***************\n');
        end
end
```

The modified Example

```
printSeparator5Lines()
%% produce some output
...
printSeparator2Lines()
%% produce some other output
...
 printSeparator2Lines()
%% produce even more output
...
 printSeparator2Lines()
%% produce the final output
...
printSeparator5Lines()
```

Can we reuse the same method for two different type of separators

- How can we generalize the required function
  - Print two rows of stars
  - Print N rows of stars
- N is the number of rows we want to print
- N is the information that method needs to know

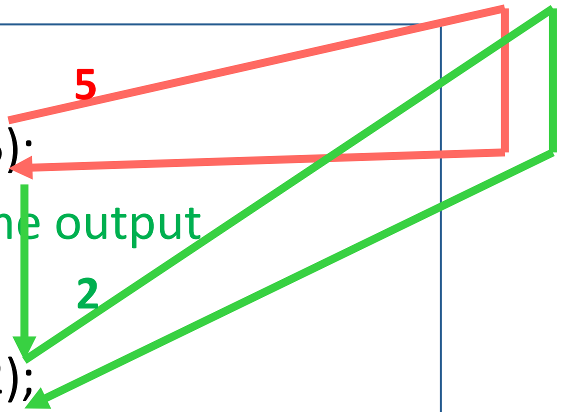
The modified Example

5

```
printSeparator(5);
%% produce some output
...
printSeparator(2);
%% produce some other output
...
 printSeparator(2);
%% produce even more output
...
 printSeparator(2);
%% produce the final output
...
printSeparator(5);
```

code for printSeparator

The modified Example



```
printSeparator(5);
%% produce some output
...
printSeparator(2);
%% produce some other output
...
 printSeparator(2);
%% produce even more output
...
 printSeparator(2);
%% produce the final output
...
printSeparator(5);
```

code for printSeparator

Structure of the modified method

```
%% print a separator line on output
function [] = printSeparator()
     ???
end
```

- n is called the **argument** (or parameter) of the method
- n can be used inside the method like a variable

Code for the modified method

```
%% print a separator line on output
function [] = printSeparator(n)
     for i =1:n
          fprintf('***************\n');
     end
end
```

Need for multiple arguments

- What if we want to set both the **number of lines** and the **number of stars per line**?

```
%% print a separator line on output
function [] = printSeparator(n)
      for i =1:n
            fprintf('***************\n');
      end
end
```

Multiple arguments

- A method can have more than one argument
- Arguments are matched based on **order**

```
printSeparator( 10, 3 )
```

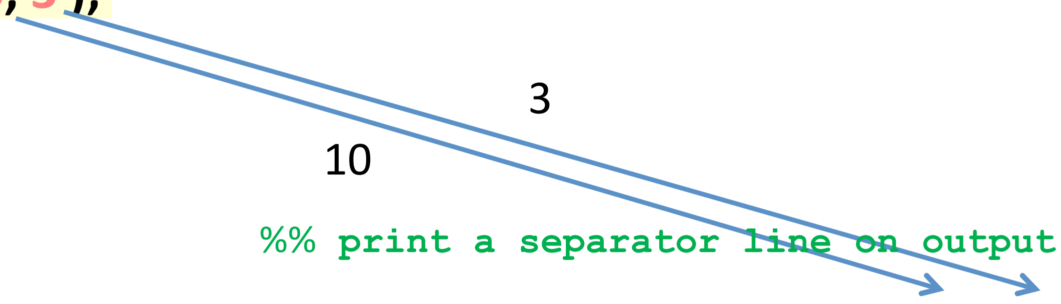
```
%% print a separator line on output

function [] = printSeparator(s, n)
      for i =1:n
            for j = 1:s
                  fprintf('*');
            end
            fprintf('\n');
      end
end
```

Multiple arguments

- A method can have more than one argument
- Arguments are matched based on **order**

```
printSeparator( 10, 3 );
```



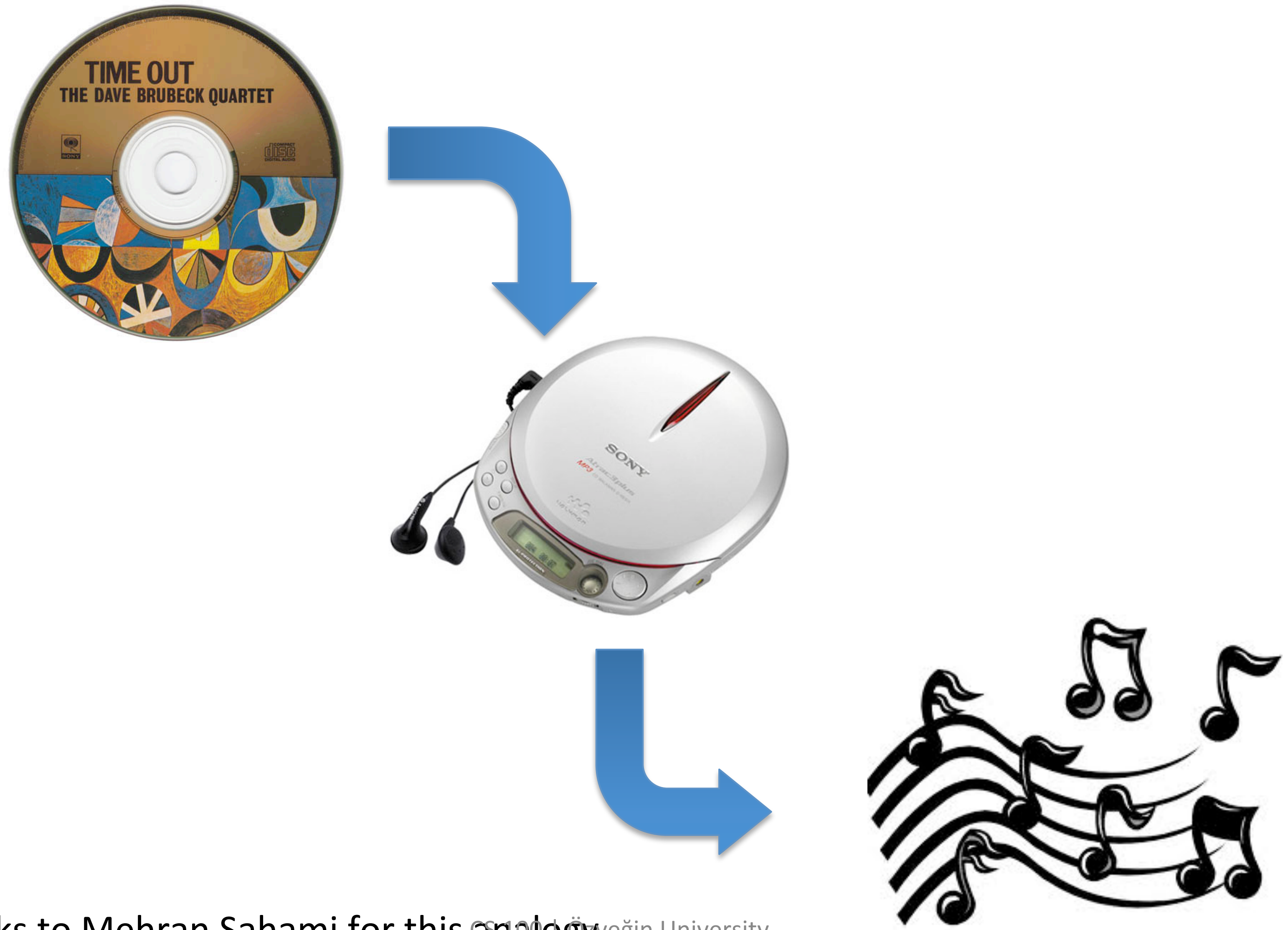
```
%% print a separator line on output

function [] = printSeparator(s, n)
        for i =1:n
                for j = 1:s
                        fprintf('*');
                end
                fprintf('\n');
        end
end
```

Method call mechanism

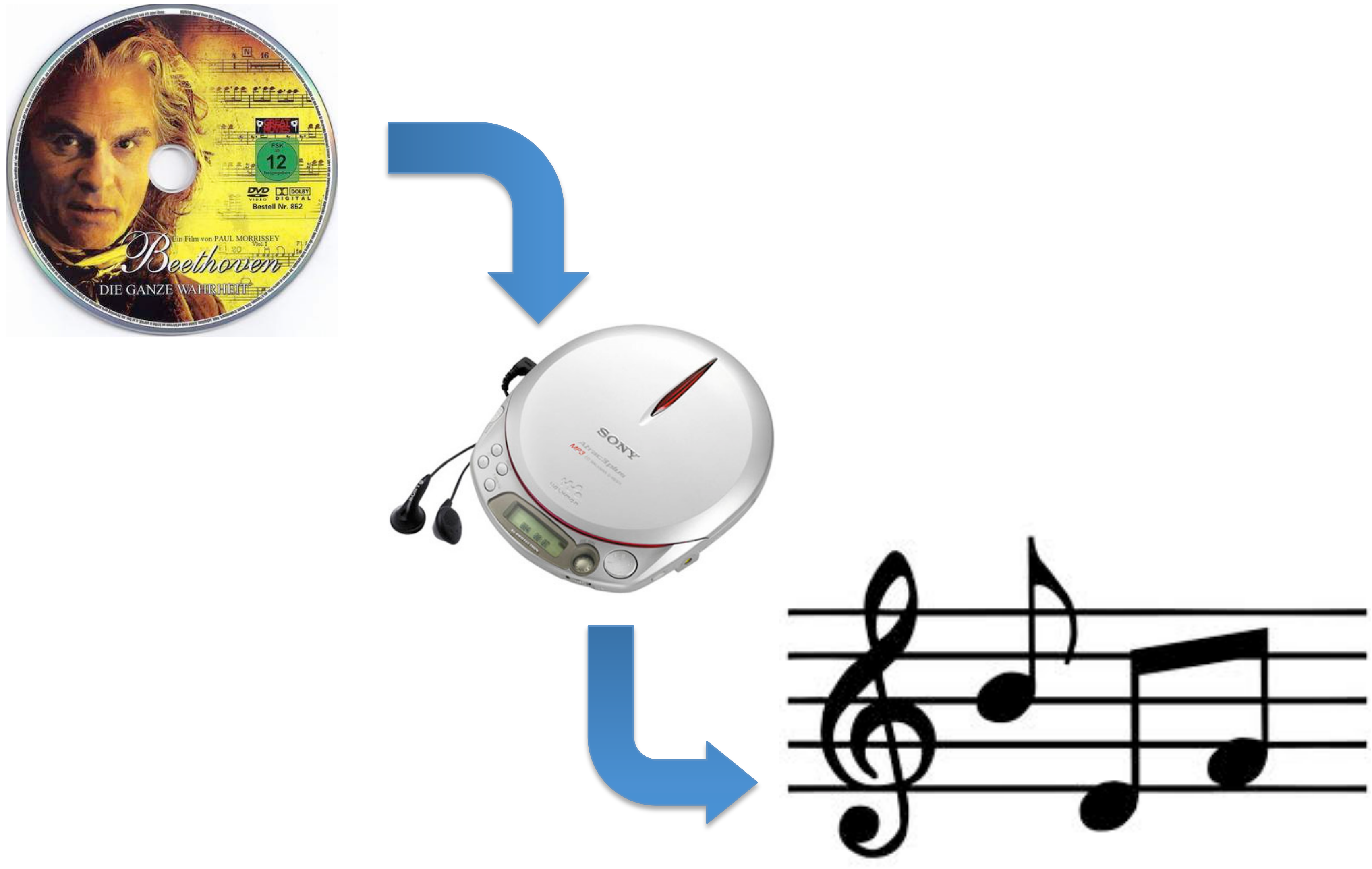
- Each method can be considered as a frame that contains
  - the code of the method
  - memory cells for all the parameters
  - memory cells for all the variables declared inside the method

CD Player is a parameterized method

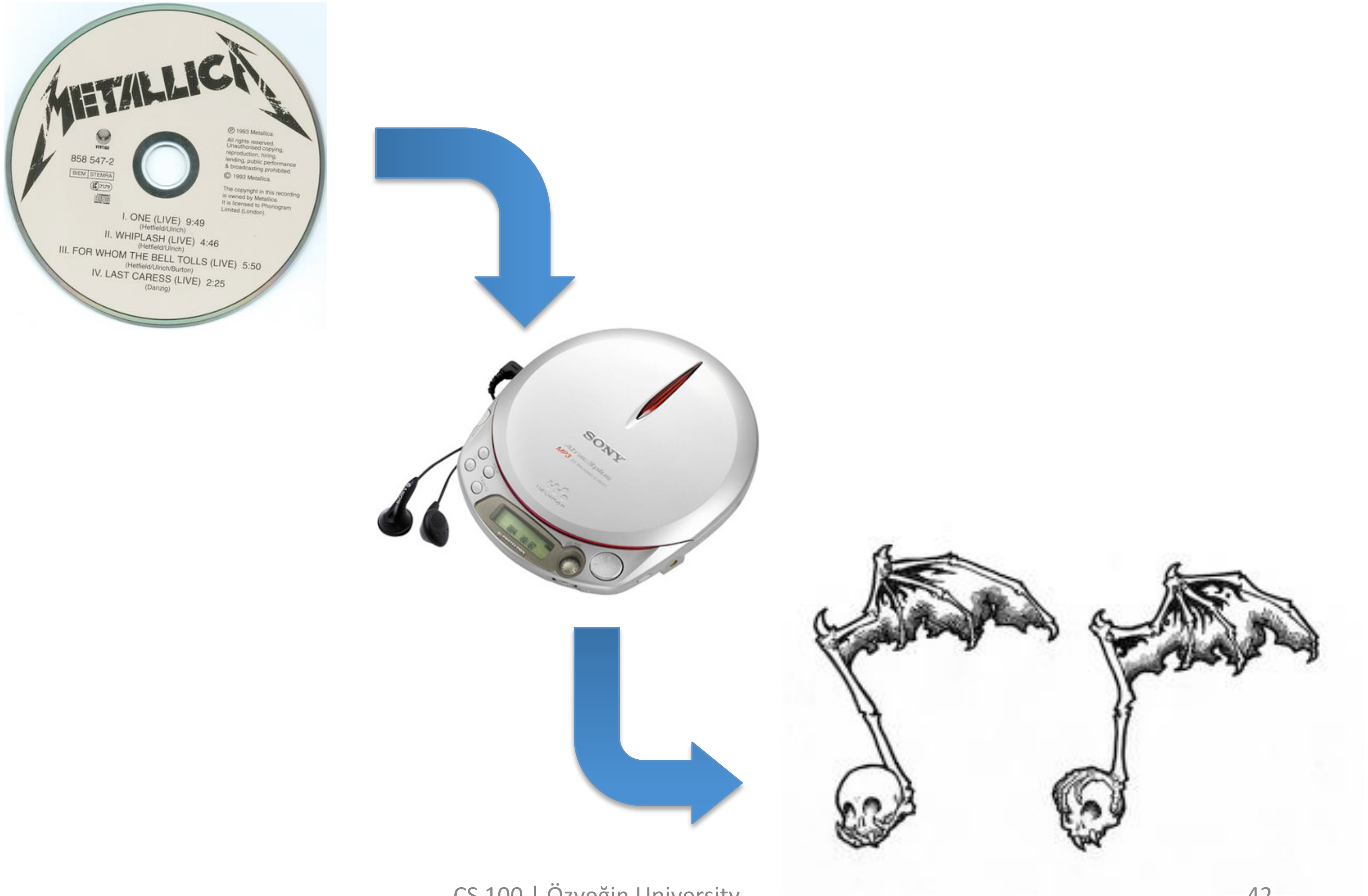


Thanks to Mehran Sahami for this analogy.

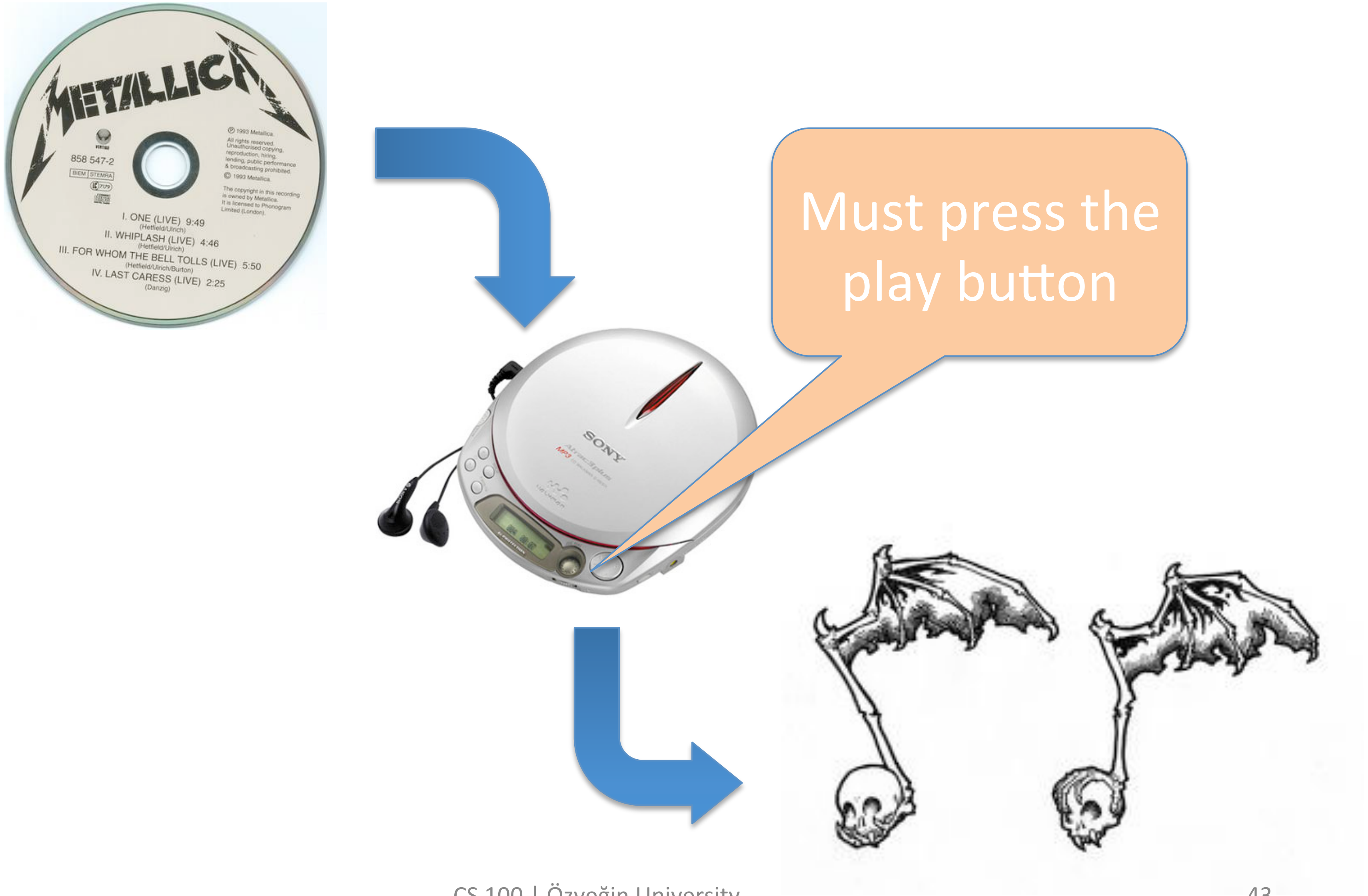
CD Player is a parameterized method



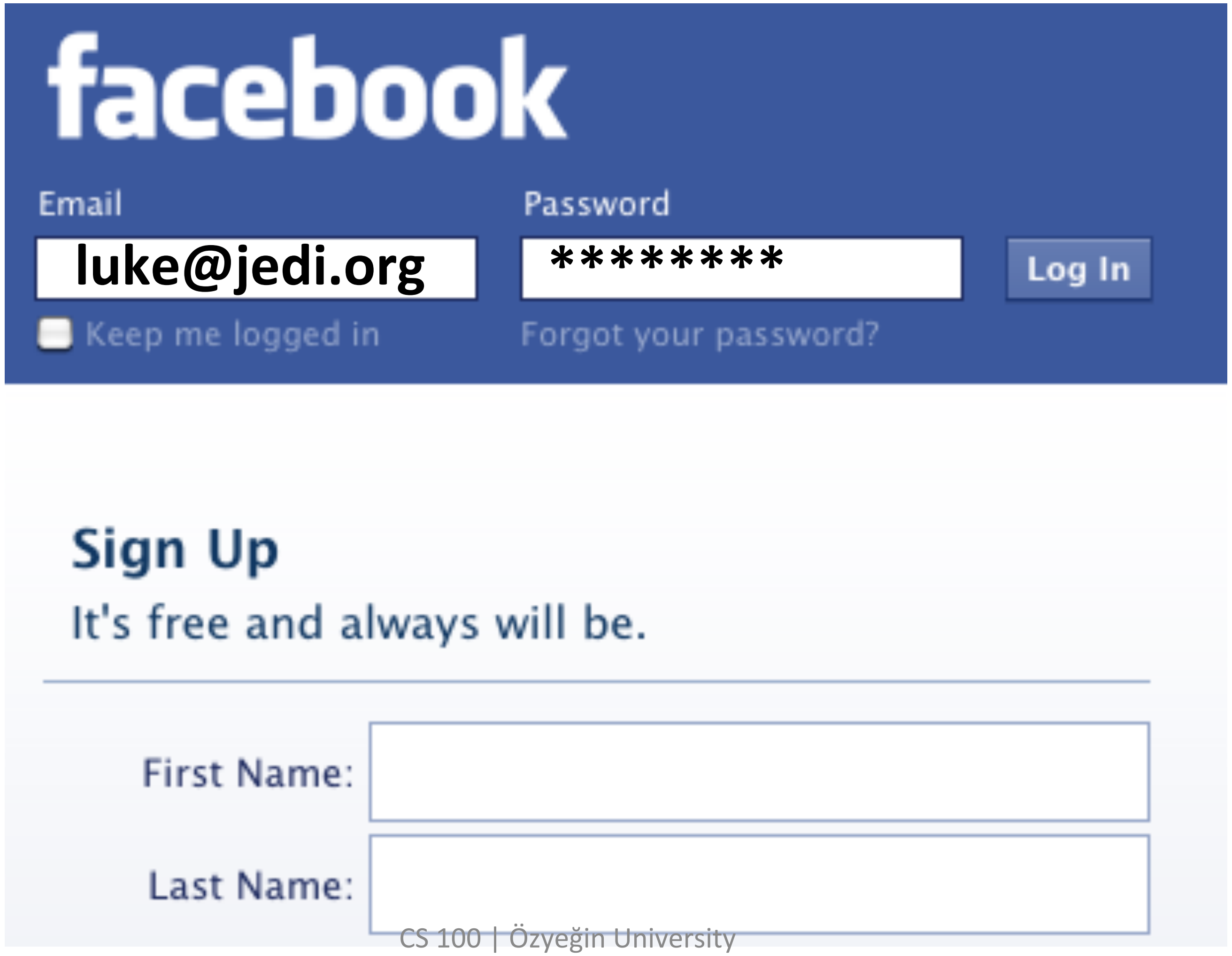
CD Player is a parameterized method

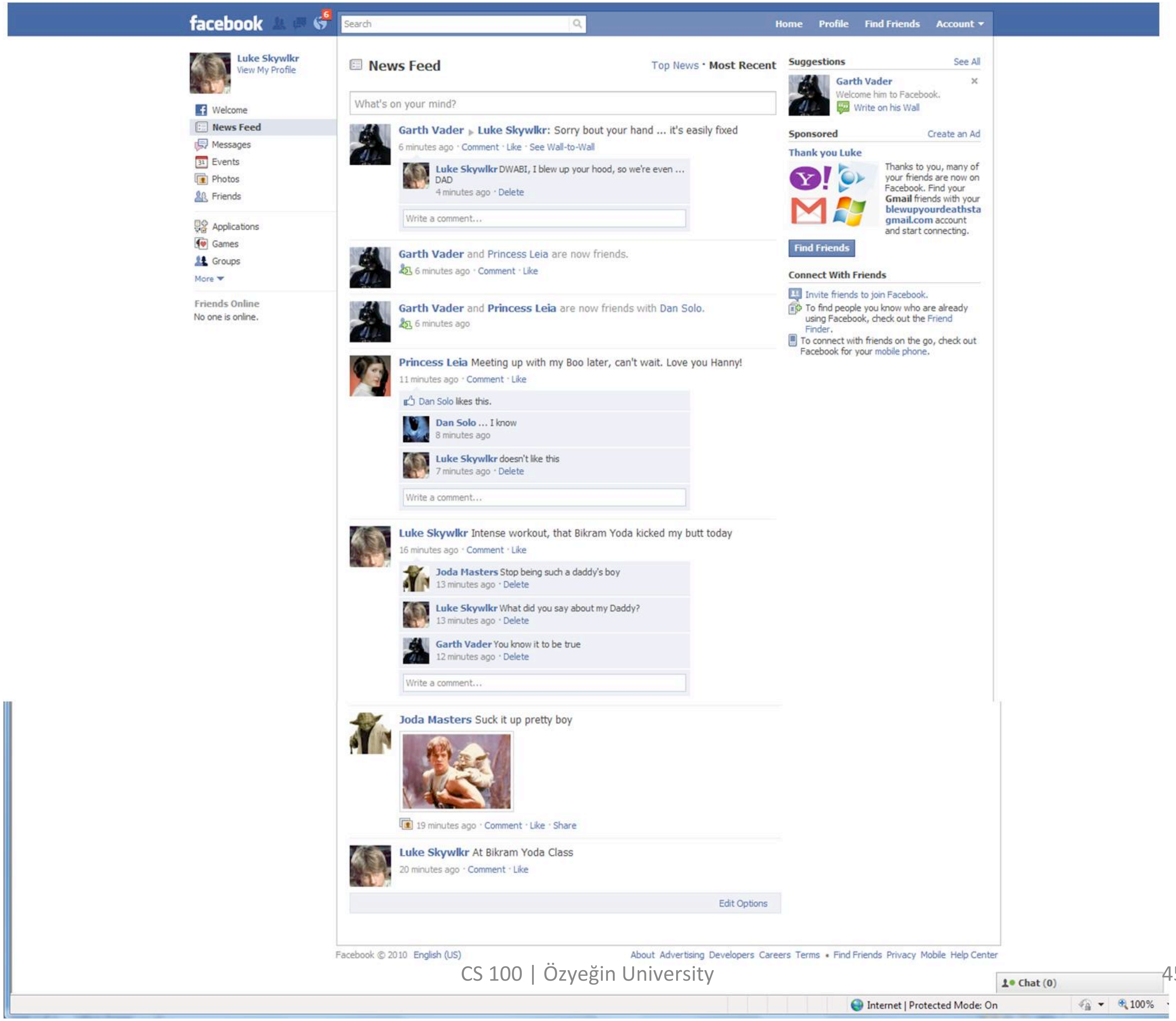

METALLICA
I. ONE (LIVE) 9:49
II. WHIPLASH (LIVE) 4:46
III. FOR WHOM THE BELL TOLLS (LIVE) 5:50
IV. LAST CARESS (LIVE) 2:25
SONY

Invoking/calling the method


METALLICA
I. ONE (LIVE) 9:49
II. WHIPLASH (LIVE) 4:46
III. FOR WHOM THE BELL TOLLS (LIVE) 5:50
IV. LAST CARESS (LIVE) 2:25
SONY
Must press the play button

Facebook is a parameterized method





News Feed — Top News · **Most Recent**

What's on your mind?

Garth Vader ▸ **Luke Skywlkr**: Sorry bout your hand ... it's easily fixed
6 minutes ago · Comment · Like · See Wall-to-Wall

Luke Skywlkr DWABI, I blew up your hood, so we're even ... DAD
4 minutes ago · Delete

Write a comment...

Garth Vader and Princess Leia are now friends.
6 minutes ago · Comment · Like

Garth Vader and **Princess Leia** are now friends with Dan Solo.
6 minutes ago

Princess Leia Meeting up with my Boo later, can't wait. Love you Hanny!
11 minutes ago · Comment · Like

Dan Solo likes this.

Dan Solo ... I know
8 minutes ago

Luke Skywlkr doesn't like this
7 minutes ago · Delete

Write a comment...

Luke Skywlkr Intense workout, that Bikram Yoda kicked my butt today
16 minutes ago · Comment · Like

Joda Masters Stop being such a daddy's boy
13 minutes ago · Delete

Luke Skywlkr What did you say about my Daddy?
13 minutes ago · Delete

Garth Vader You know it to be true
12 minutes ago · Delete

Write a comment...

Joda Masters Suck it up pretty boy

19 minutes ago · Comment · Like · Share

Luke Skywlkr At Bikram Yoda Class
20 minutes ago · Comment · Like

Edit Options

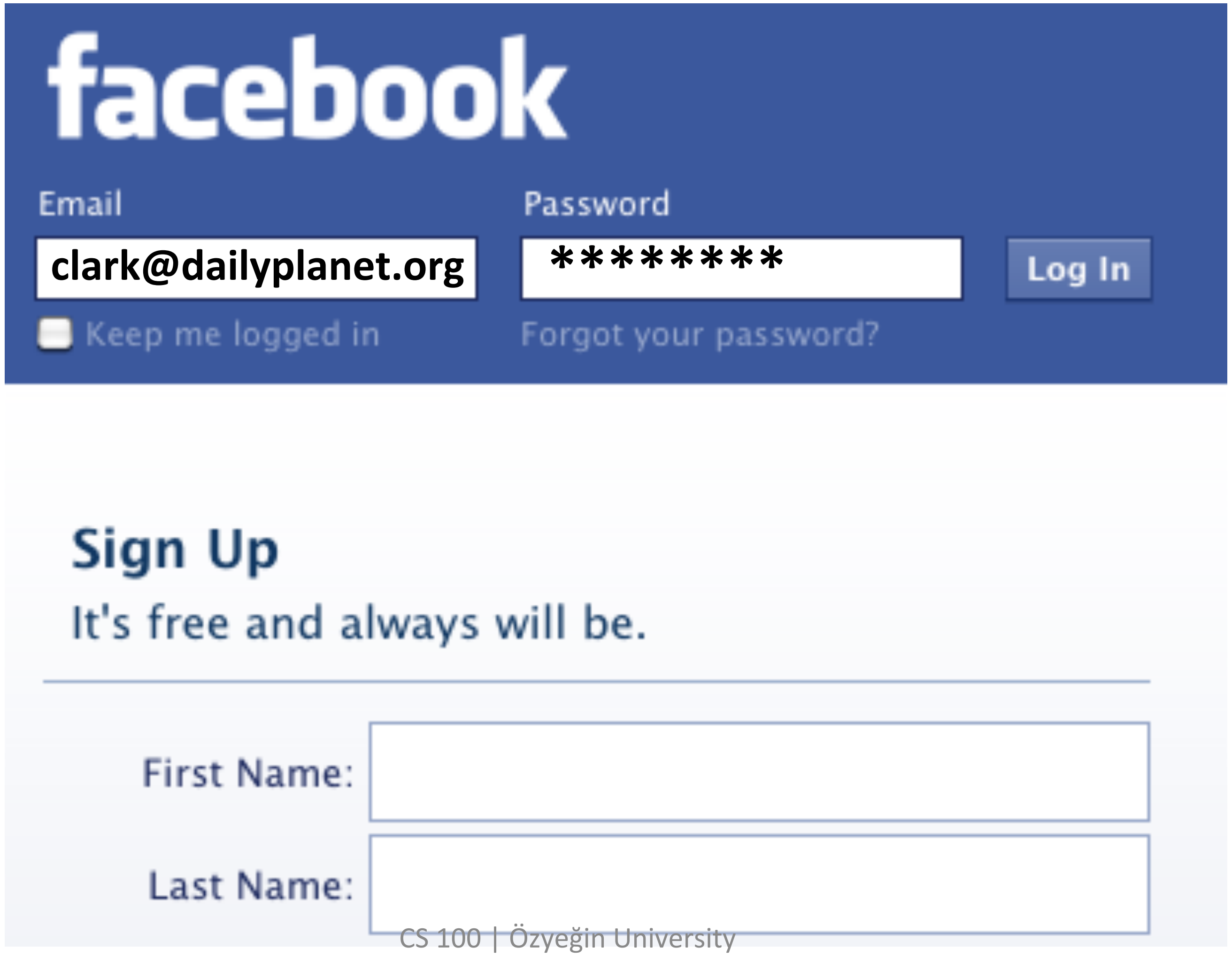
Suggestions See All
Garth Vader
Welcome him to Facebook.
Write on his Wall

Sponsored Create an Ad
Thank you Luke
Thanks to you, many of your friends are now on Facebook. Find your **Gmail** friends with your **blewupyourdeathstagmail.com** account and start connecting.
Find Friends

Connect With Friends
Invite friends to join Facebook.
To find people you know who are already using Facebook, check out the Friend Finder.
To connect with friends on the go, check out Facebook for your mobile phone.

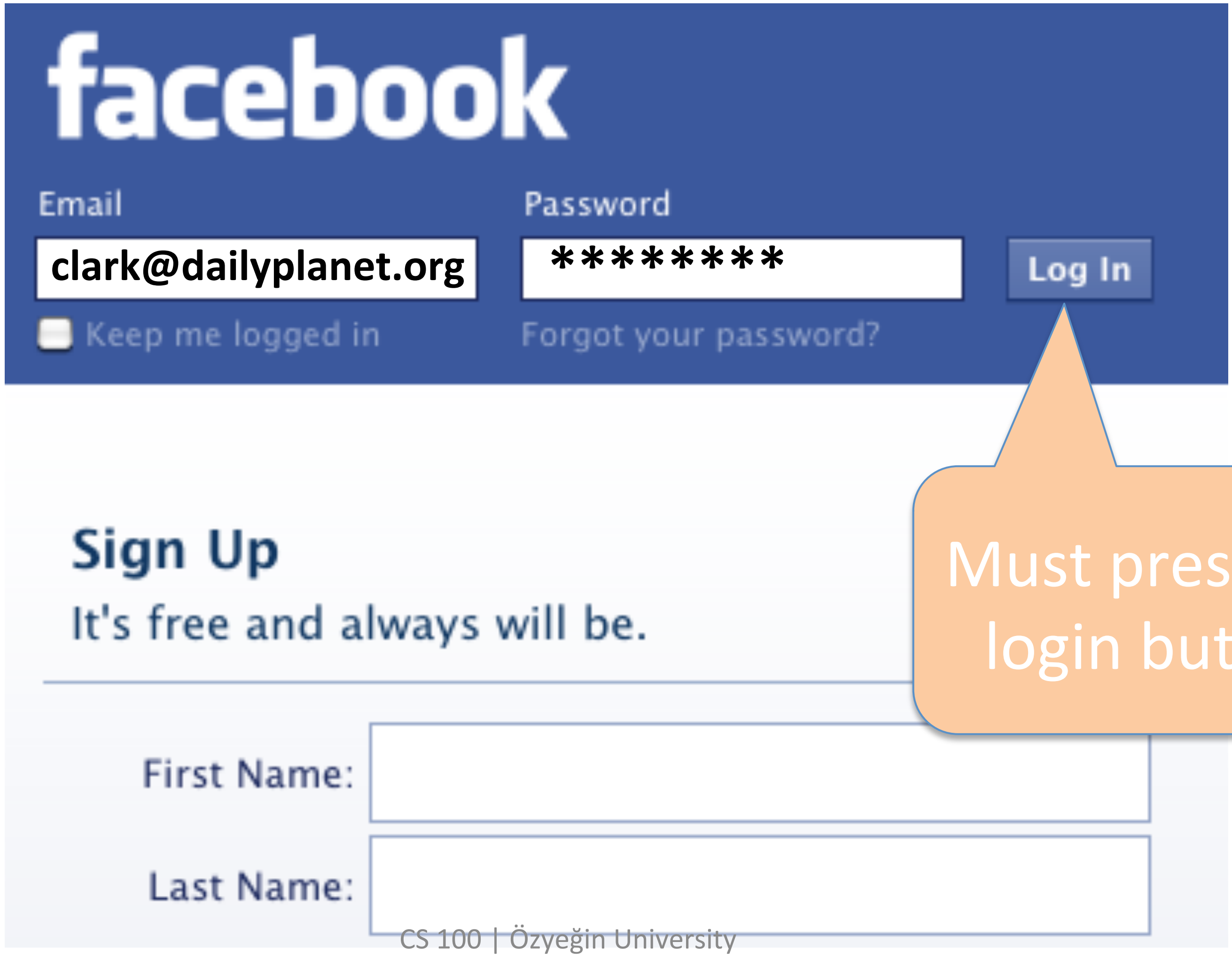
Facebook © 2010 English (US)
About Advertising Developers Careers Terms • Find Friends Privacy Mobile Help Center

Facebook is a parameterized method



The Joy of Tech™
by Nitrozac & Snaggy
facebook
Clark Kent
News Feed
Messages
Events
Friends
Game Requests
Application Requests
More
Chat with Friends
News Feed
Martha Kent
Tagged: Clark's baby pics!
Message: Clark honey, when you get back from Saturn, can you pick me up some milk?
5,789,524 people like this.
Batman Ha ha! Busted!
joyoftech.com
Superman's social network nightmare.

Invoking/calling the method



Must press the login button

fprintf is a parameterized method

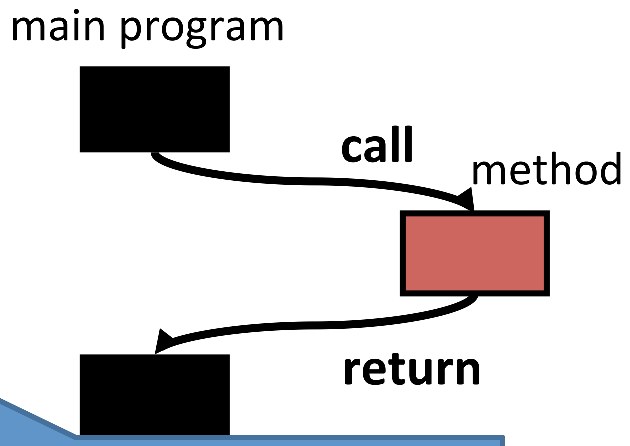
```
fprintf(’Hello, world\n');
```

fprintf is a parameterized method

```
fprintf('May the force be with you\n');
```

New Concepts

- Two new concepts:
  - **Return** values
  - **Parameters**
- The simple method in ur example did not have any return va or a parameter



Now we will focus on this new concept

```
function [] = printSeparator()
        fprintf('***************\n');
        fprintf('**************\n');
end
```

A new example problem..

- Write a method which, given the radius, computes and returns the area of a circle with that radius

- The new problem here is that
  - The method should **return** a value

Return values..

- Arguments are used for sending data to the method
- Return values are used for the opposite: for sending data back

Example: The *area* method

- Write a method which, given the radius, computes and returns the area of a circle with that radius
- New feature:
  - The return statement sends a value back

```
%% calculates the area of a circle with radius r
function area = areaOfCircle(r)
      area = 3.14 * r * r;
end
```

Control and Data Flow

```
r = input('Enter the radius of the circle');
a = areaOfCircle(r);
fprintf('The area of the circle is %d\n' , a);
```

```
%% calculates the area of a circle with radius r
function area = areaOfCircle(r)
      area = 3.14 * r * r;
end
```

Returning a value to the caller

- The **return** statement…

Which one's heavier?

Returning a value to the caller

- The **return** statement…

Calls Asterix.

passes him the arguments

Returning a value to the caller

- The **return** statement…

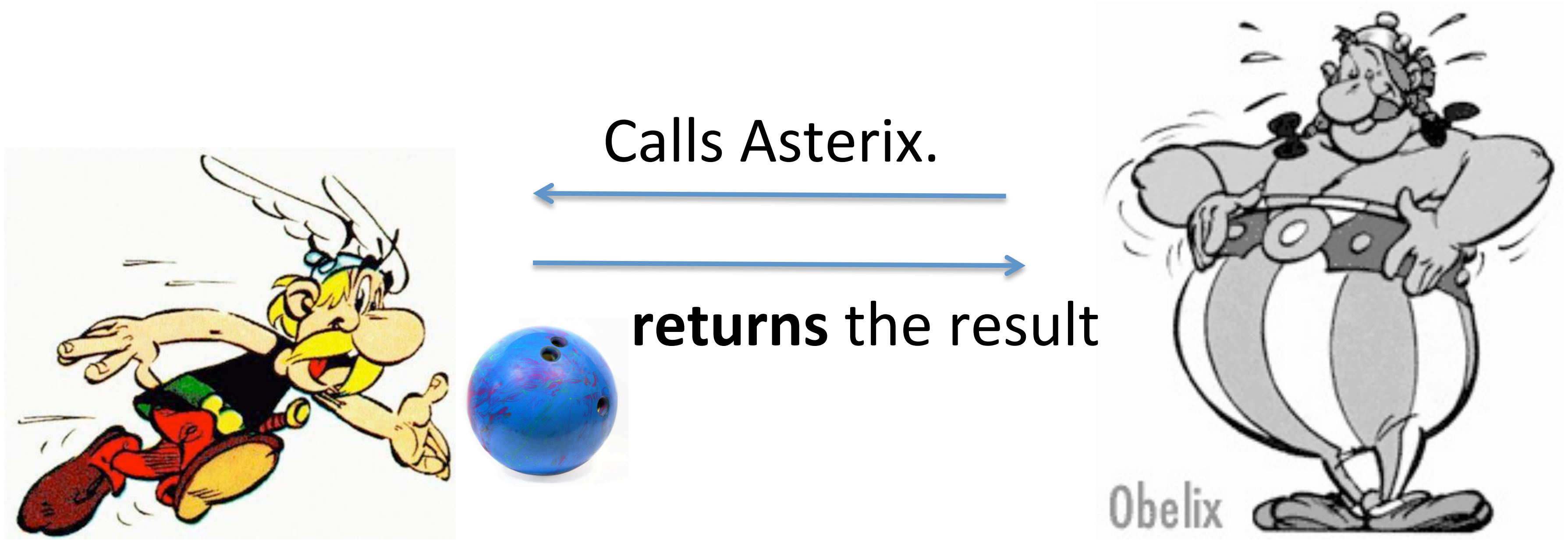
waiting for Asterix

Performs measurement

Calls Asterix.

Returning a value to the caller

- The **return** statement…



Returning a value to the caller

- The **return** statement…

continues his life…

Different parameters

- May do the same thing by passing Asterix different arguments.

Calls Asterix.

passes him the arguments

Returning a value to the caller

- The **return** statement…

Calls Asterix.

returns the result

Returning a value to the caller

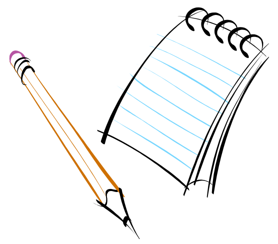
- A method can return a value to its caller.

```
n1 = input('Enter first number: ');
n2 = input('Enter second number: ');
n3 = input('Enter third number: ');

maxNumber = max(max(n1,n2), n3);
fprintf('Max is %d\n', maxNumber);
```

```
function mval = max(x, y)
        if(x > y)
            mval = x;
        else
            mval = y;
        end
end
```

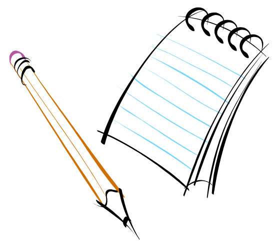
Exercise: Methods



- Write a method that calculates the factorial of a number, n
- n should be provided as an **integer** argument
- The method will be used by the run method as follows

```
number = input('Enter a number: ');
result = myfactorial(number);
fprintf('%d! = %d\n' , number, result);
```

Exercise: Methods



```
%% calculates the factorial of a number
function fact =  myfactorial(n)
       fact = 1;
       for i = 2:n
              fact = fact * i;
       end
end
```

Local variables

- Variables that are defined inside a method
- Cannot be used by other methods
  - The value of a local variable can be sent to another method only as an argument of that method
- Created just before the method executes, destroyed when the method returns
  - Local to the method where it is defined
  - Note: parameters are also local

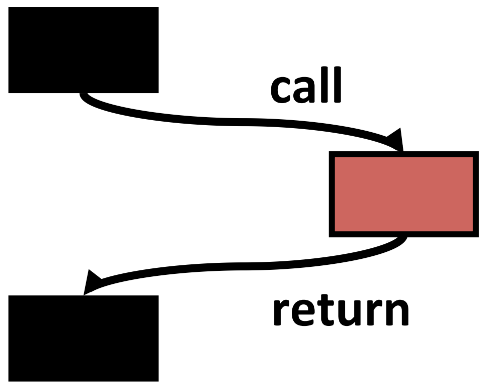
Method File Names

The name of the files that the method is saved should be named after the method’s name.

Review Method Control Flow

- Methods allow you to **visit** a block of code and then **come back**
  - This code block can be elsewhere in your program (in another class)

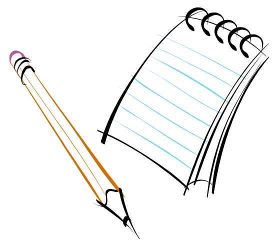
- We have described the basic flow before..



When a method is called..

- Memory space is **allocated** for the method arguments and local variables
- Argument values are **copied**
- Control **transfers** to the method
- The method **executes**
- Control and the **return** value is transferred back to the point of call

Exercise: raise to power



- Write a method that calculates n^k

Methods: Summary

- Methods may take **several** parameters, or none
- Methods may return **several** values, or none
- Methods are valuable
  - A tool for program structuring
  - Provide abstract services: the caller cares **what** the methods do, but not **how**
  - Makes programs easier to write and understand

Methods

- Each method can be considered as a frame that contains
  - the code of the method
  - memory cells for all the parameters
  - memory cells for all the variables declared inside the method

Methods calling other methods

```
function [] = printSeparator()
       fprintf('****************\n');
       fprintf('***************\n');
end
```