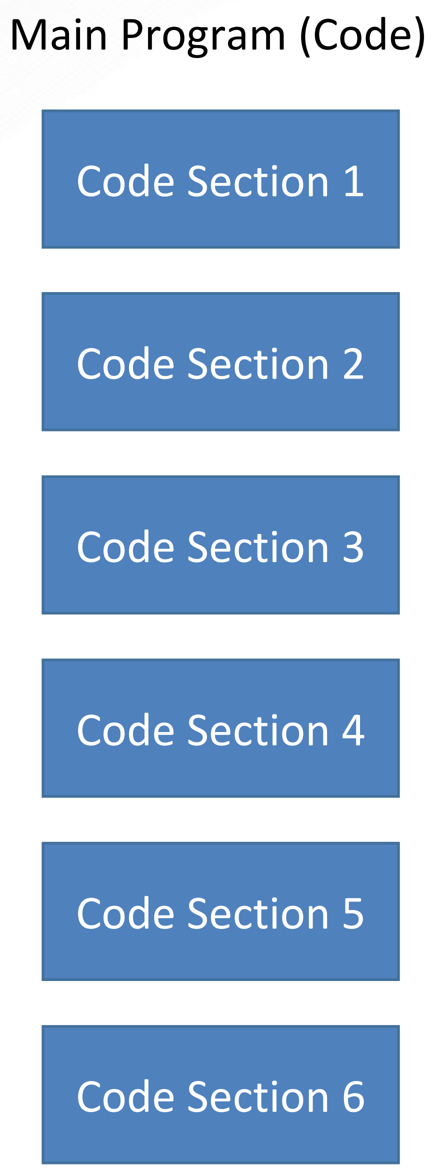
Week 11

Pre-Lecture Slides: Creating Functions

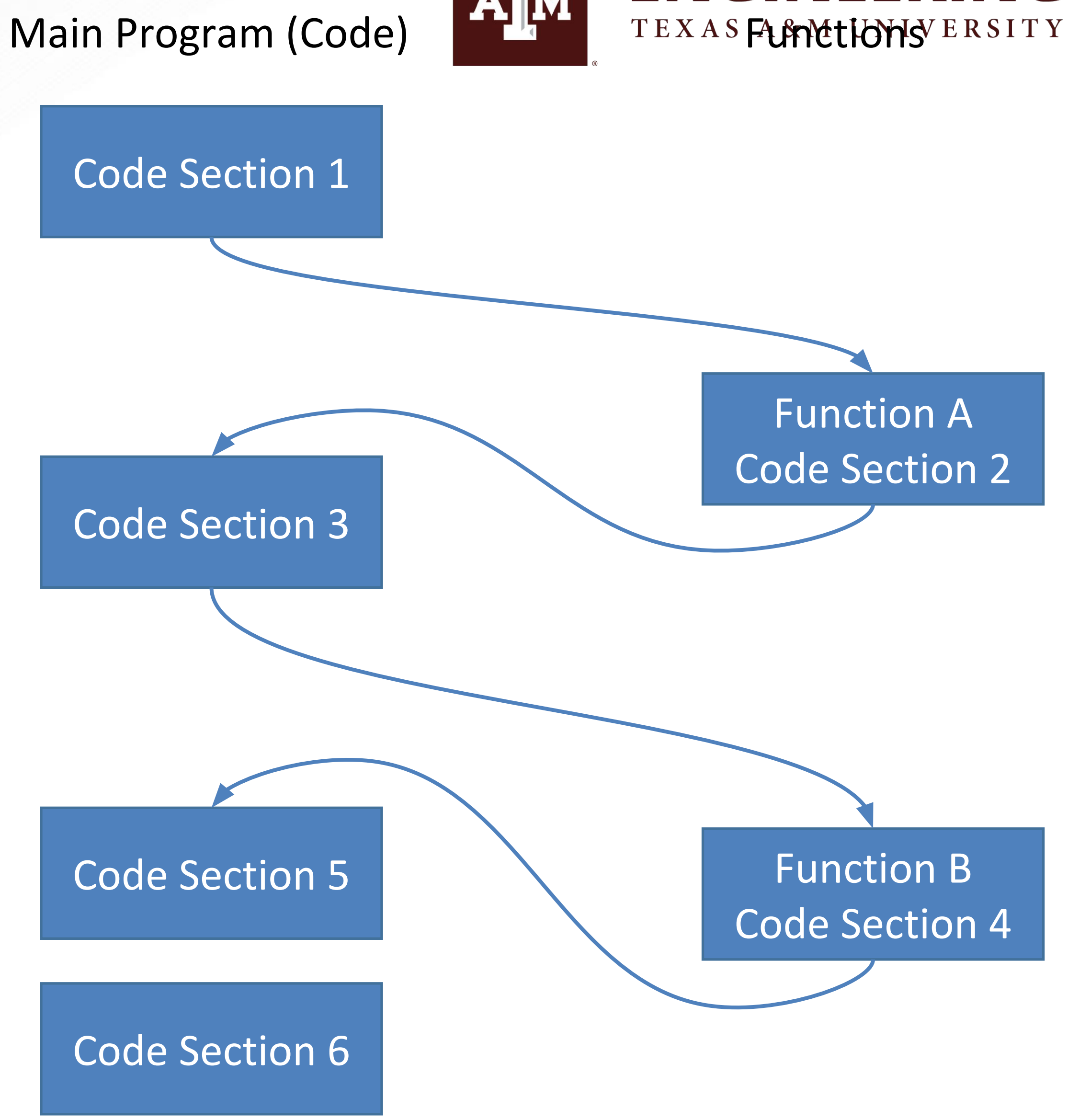
Functions – Main Ideas

Functions are essentially a section of code specified separately from the rest of the code.



Functions – Main Ideas

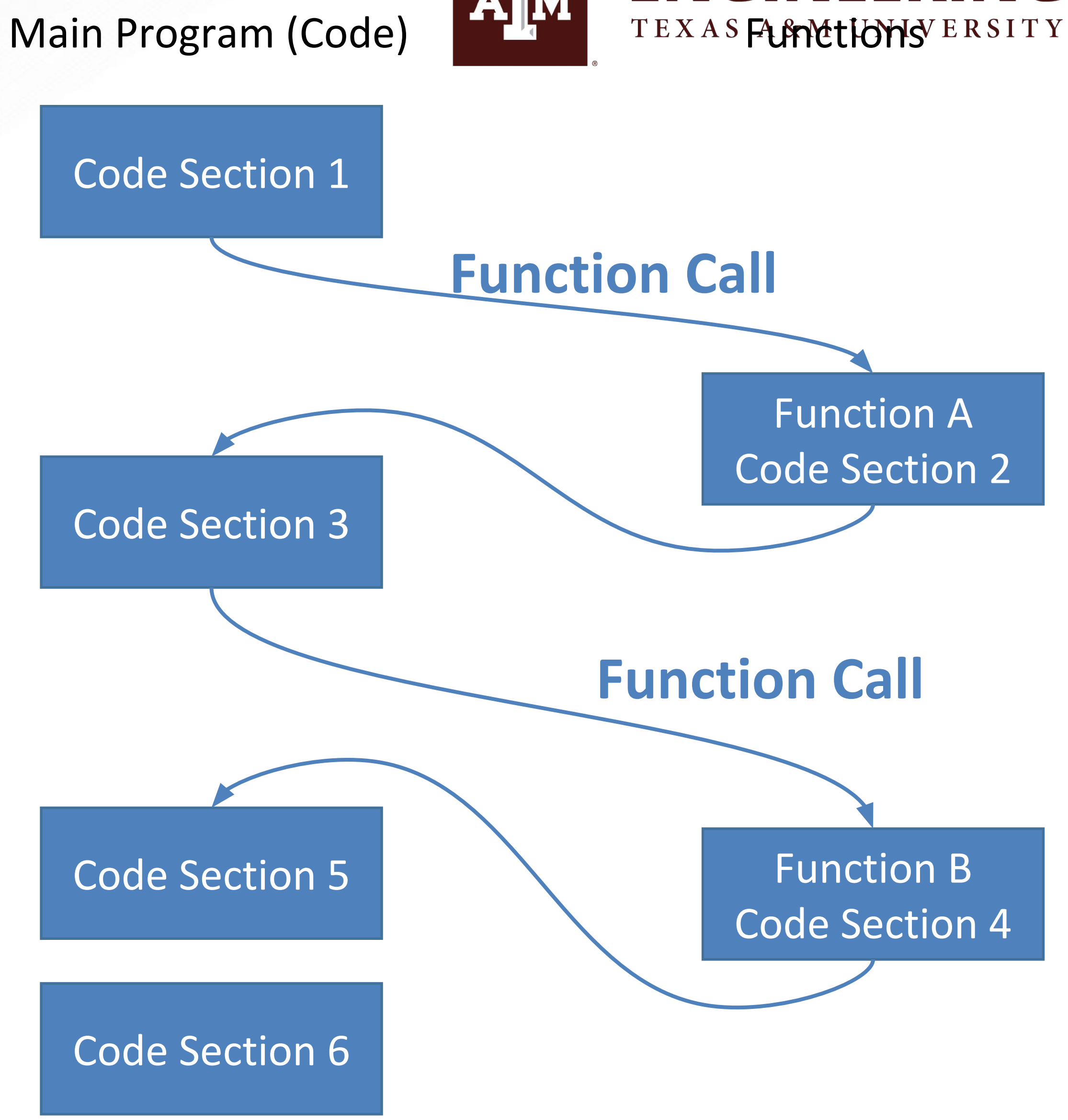
Functions are essentially a section of code specified separately from the rest of the code.



Functions – Main Ideas

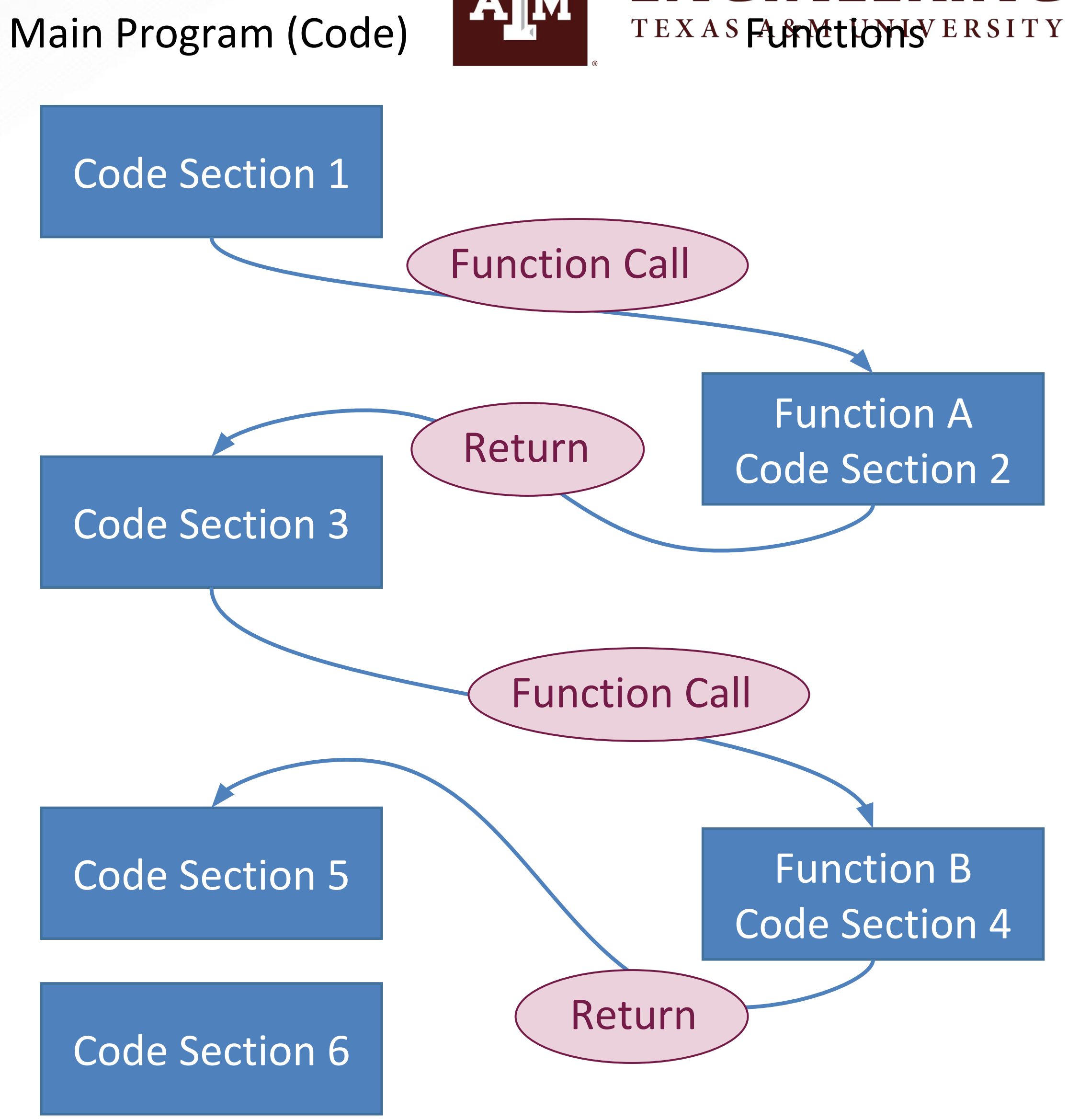
When following code, we will encounter a function **call**

- Also called 'invoking' the function
- Causes the code in the function to be executed next



Functions – Main Ideas

When a function is finished, we **return** to the place it was called from.



Why do we have functions?

Prevent re-writing the same code repeatedly

- We can call the same function several times

Helps conceptually separate parts of the code

- A sections of code that can be thought of as a single action can be separated.
- We don’t see a big block of code that’s separated from other things

Can write a function separately from the rest of the code

Passing Data To/From Functions

Since functions are separate from the rest of the code, they don't necessarily have access to the same data

- We pass data into the function using **arguments**
  - For example, `print("Howdy!")` is passing the string "Howdy" to the function
  - Ideally, all the data the function needs should be passed in through the arguments.
- Data is returned from the function by a **return value**
  - For example, `input()` will return a string – the string entered in the console.
  - We can then assign the returned value to a variable, or use it in an expression

Writing our own functions

The main idea:

- We define the function (first)
- We call the function whenever we want

Function definition

```
def <function name>(<parameters>):
    <stuff to do>
```

A function definition starts with the keyword “def”.

This first line starting with def is often called the function **header**.

Function definition

```
def <function name>(<parameters>):
    <stuff to do>
```

Next is the name of the function. This will be used when the function is called. It should be a unique name.

Function definition

```
def <function name>(<parameters>):
    <stuff to do>
```

Next come parentheses, which possibly contain a list of parameters.

ENGINEERING
TEXAS A&M UNIVERSITY

Function definition

```
def <function name>(<parameters>):
    <stuff to do>
```

Then, there is a colon and the rest of the function definition is indented.

Function definition

```
def <function name>(<parameters>):
    <stuff to do>
```

The **body** of the function is the set of operations/commands that the function should do when called.

Example: writing a function

We want a function to print a warning. We define it as follows:

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")
```

Example: writing a function

Then we can call it:

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")
warn()
```

Console

```
********** WARNING!!! **********
You are about to do something dangerous!
```

Example: writing a function

You must define the function before you try to call it.

```
warn()
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")
```

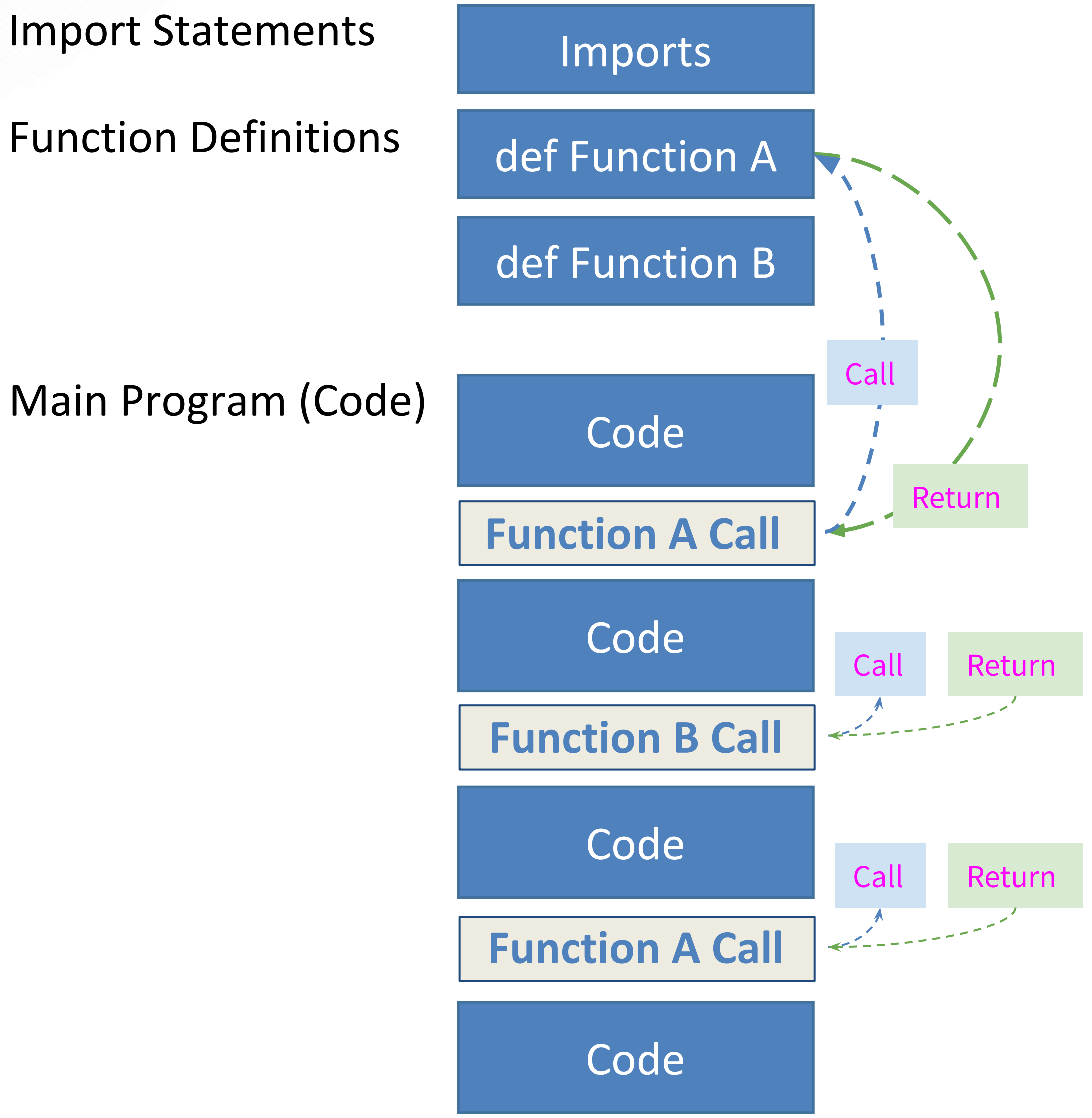
Console

```
NameError: name 'warn' is not defined
```

Functions in a program

List <u>all</u> functions near the beginning of a program, before the “main” code.

1. import statements
2. function definitions
3. main code



ENGINEERING
TEXAS A&M UNIVERSITY

You can call one function from another:

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Console

```
********** WARNING!!! **********
You are about to do something dangerous!
********** WARNING!!! **********
You are about to do something dangerous!
```

The interpreter and functions

When the interpreter encounters a function definition, it “remembers” the name of the function and how many parameters it takes.

- It does not go through the function body

When the function is called, the interpreter goes back to the function body and executes those commands

Example: executing a function

ENGINEERING
TEXAS A&M UNIVERSITY

Next

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

The interpreter first encounters a definition of the function named warn

Console

Example: executing a function

Next

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

It remembers where it saw "warn" and skips the body.

Console

Example: executing a function

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Next

Likewise, it remembers where it saw “doublewarn” and skips the body.
It is next going to encounter the function call

Console

Example: executing a function

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Next

Return HERE

The function call was to doublewarn, so it goes back to the body of that function.
It remembers where it needs to return once it's done.

Console

Example: executing a function

Next

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Return HERE

Return HERE

There was now a call to the function warn, so it goes up to that body.

Console

Example: executing a function

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Next

Return HERE

Return HERE

It executes one line.

Console

```
********** WARNING!!! **********
```

Example: executing a function

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Next

Return HERE

Return HERE

Then the next line.
It also returns to the point it was called from.

Console

```
********** WARNING!!! **********
You are about to do something dangerous!
```

Example: executing a function

Next

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Return HERE

Return HERE

We again encounter a function call.

Console

```
********** WARNING!!! **********
You are about to do something dangerous!
```

Example: executing a function

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Next

Return HERE

Return HERE

Again, the first line executes.

Console

```
********** WARNING!!! **********
You are about to do something dangerous!
********** WARNING!!! **********
```

Example: executing a function

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Next

Return HERE

Return HERE

Then the next.
It now returns to where it’s called from.

Console

```
********** WARNING!!! **********
You are about to do something dangerous!
********** WARNING!!! **********
You are about to do something dangerous!
```

Example: executing a function

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    warn()
    warn()

doublewarn()
```

Next

Return HERE

And since that function is also complete, it returns to where it was called from previously.

Console

```
********** WARNING!!! **********
You are about to do something dangerous!
********** WARNING!!! **********
You are about to do something dangerous!
```

ENGINEERING
TEXAS A&M UNIVERSITY

What would happen here?

```
def warn():
    print("********** WARNING!!! **********")
    print("You are about to do something dangerous!")

def doublewarn():
    callingundefinedfunction()

warn()
```

Notice that doublewarn is calling a nonexistent function.

Console