



Digital Systems

10.1 Functions

FOX
MIS

Functions

Digital Product Management

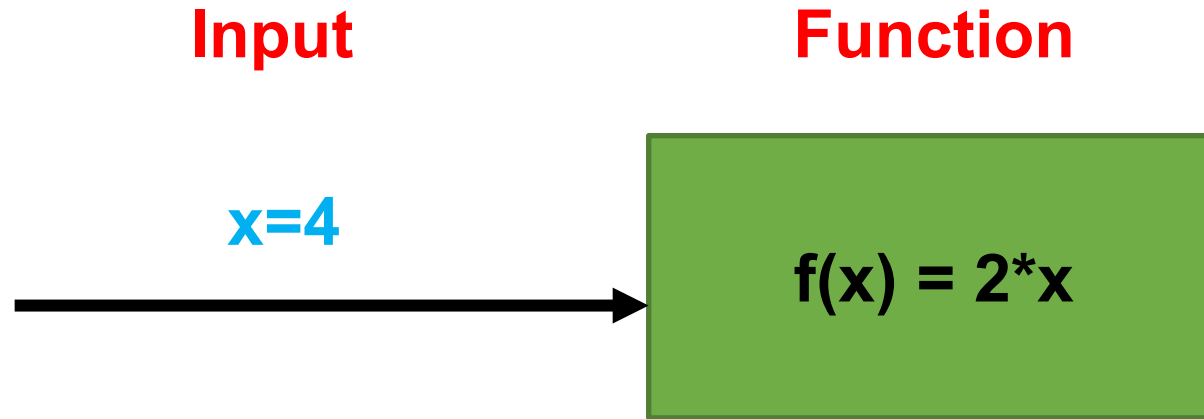
FOX
MIS

What is a function in Mathematics?

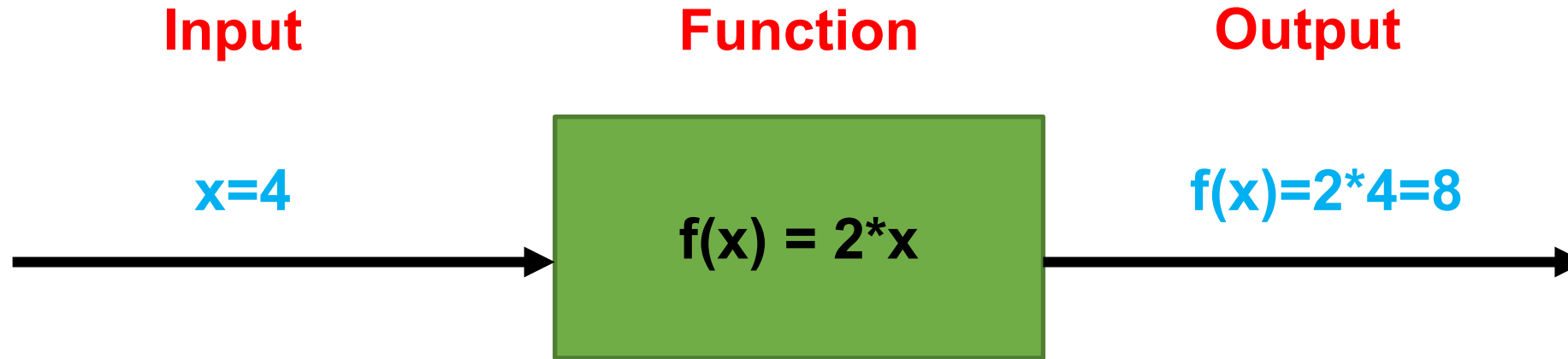
Function

$$f(x) = 2^x$$

What is a function in Mathematics?

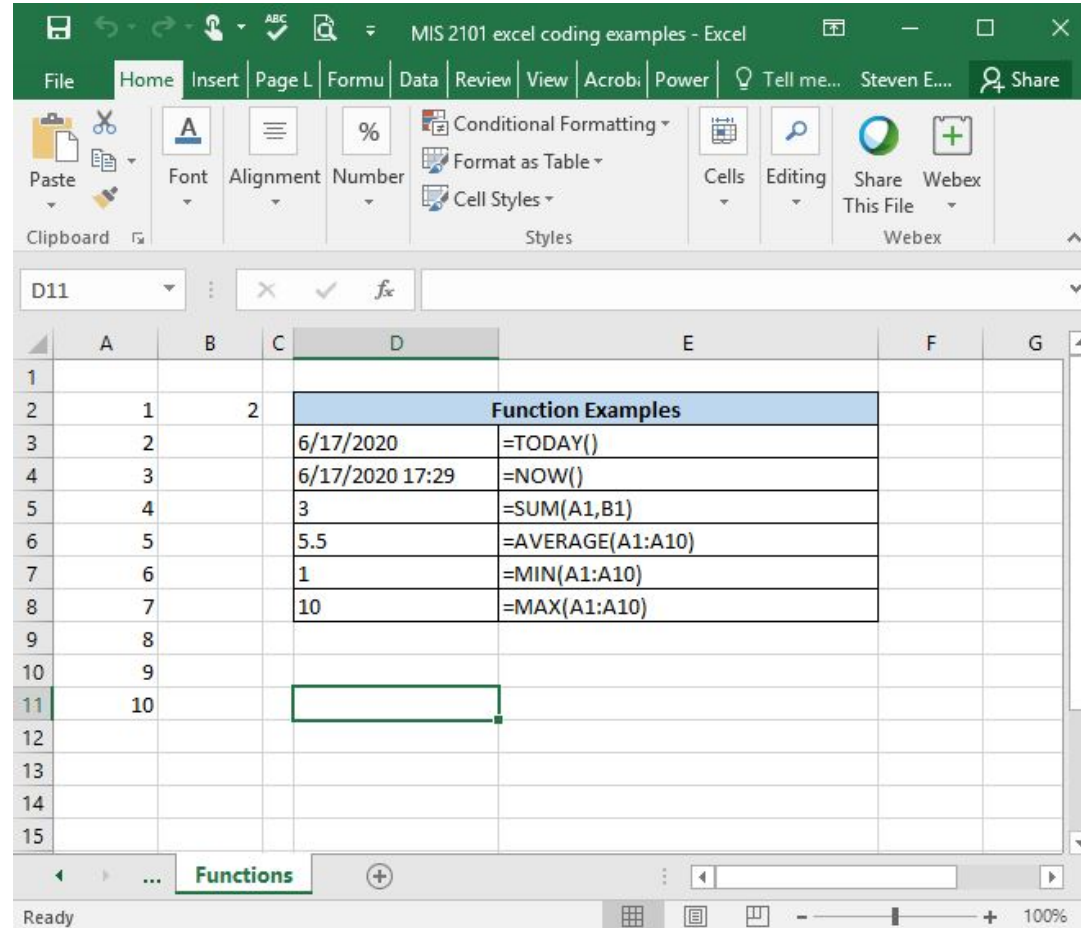


What is a function in Mathematics?



What is a function in Excel?

- **=TODAY()**
- **=NOW()**
- **=SUM(A1,B1)**
- **=AVERAGE(A1:A10)**
- **=MIN(A1:A10)**
- **=MAX(A1:A10)**



What is a function in Excel? (cont.)

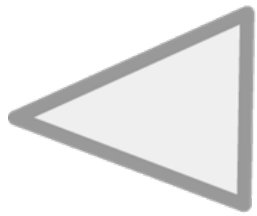
- **All functions**
 - **Have a name**
 - **Are passed zero or more pieces of information**
 - **Return a value**

Functions allow our code to
be more maintainable and
reusable!

What is a function in JavaScript?

- **A way to organize your code to make it easier to create, maintain and reuse**
- **All functions:**
 - Have a name
 - Are passed zero or more pieces of information
 - Return a value (usually)
- **Main program just does basic input/output.**
 - All of the real work is packaged up and performed in functions

A realistic-looking spaceship!



distance traveled?

Source: JavaScript Absolute Beginner's Guide by Kirupa Chinnathambi

distance = speed x time

Source: JavaScript Absolute Beginner's Guide by Kirupa Chinnathambi

Meanwhile in JS Land

That diagram can be turned into the following:

```
let speed = 10;  
let time = 5;  
alert(speed * time);
```

Let's say we have to calculate the distance multiple times.

Our code might look as follows.

```
let speed = 10;  
let time = 5;  
alert(speed * time);
```

```
let speed1 = 85;  
let time1 = 1.5;  
alert(speed1 * time1);
```

```
let speed2 = 12;  
let time2 = 9;  
alert(speed2 * time2);
```

```
let speed3 = 42;  
let time3 = 21;  
alert(speed3 * time3);
```

You should avoid unnecessarily repeating code. It makes your life more complicated.

This is where functions
come in...

Meet the Function

Using functions, the code we saw earlier can look like this:

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```


What exactly is a function?

At a very basic level, a function is nothing more than a wrapper for some code. It does two things well:

1. Groups statements together
2. Makes your code reusable

You will rarely write or use code that doesn't involve functions!

```
let speed = 10;
let time = 5;
alert(speed * time);

let speed1 = 85;
let time1 = 1.5;
alert(speed1 * time1);

let speed2 = 12;
let time2 = 9;
alert(speed2 * time2);

let speed3 = 42;
let time3 = 21;
alert(speed3 * time3);
```

VS.

```
function showDistance(speed,
time) {
    alert(speed * time);
}

showDistance(10, 5);
showDistance(85, 1.5);
showDistance(12, 9);
showDistance(42, 21);
```

Trace a Function in JavaScript

```
function showDistance(speed, time) {  
  alert(speed * time);  
}
```

```
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Invoke showDistance(10, 5)

Trace a Function in JavaScript

Pass the value of 10 to speed
Pass value of 5 to time

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Trace a Function in JavaScript

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Calculate $\text{speed} * \text{time} = 10 * 5 = 50$
Display 50

Trace a Function in JavaScript

```
function showDistance(speed, time) {  
  alert(speed * time);  
}
```

```
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Invoke showDistance(85, 1.5)

Trace a Function in JavaScript

Pass the value of 85 to speed
Pass value of 1.5 to time

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Trace a Function in JavaScript

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Calculate $\text{speed} * \text{time} = 85 * 1.5 = 127.5$
Display 127.5

Trace a Function in JavaScript

```
function showDistance(speed, time) {  
  alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Invoke showDistance(12, 9)

Trace a Function in JavaScript

Pass the value of 12 to speed
Pass value of 9 to time

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Trace a Function in JavaScript

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Calculate $\text{speed} * \text{time} = 12 * 9 = 108$
Display 108

Trace a Function in JavaScript

```
function showDistance(speed, time) {  
  alert(speed * time);  
}
```

```
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Invoke showDistance(42, 21)

Trace a Function in JavaScript

Pass the value of 42 to speed
Pass value of 21 to time

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Trace a Function in JavaScript

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

Calculate speed*time=42*21=882
Display 882

A Simple Function

```
function sayHello() {  
    alert("hello!");  
}
```

You have the **function** keyword, followed by your function name, some weird parentheses and brackets, and the code your function will run when called.

Calling a Function

```
function sayHello() {  
    alert("hello!");  
}  
  
sayHello();
```

The function call is typically the name the function you want to **call** (aka **invoke**) followed again by the parentheses.

What exactly a function does can be customized. It doesn't have to be boring and predictable like what have seen so far. One way is by providing what are known as **arguments** where your function call contains some data that you pass into the function.

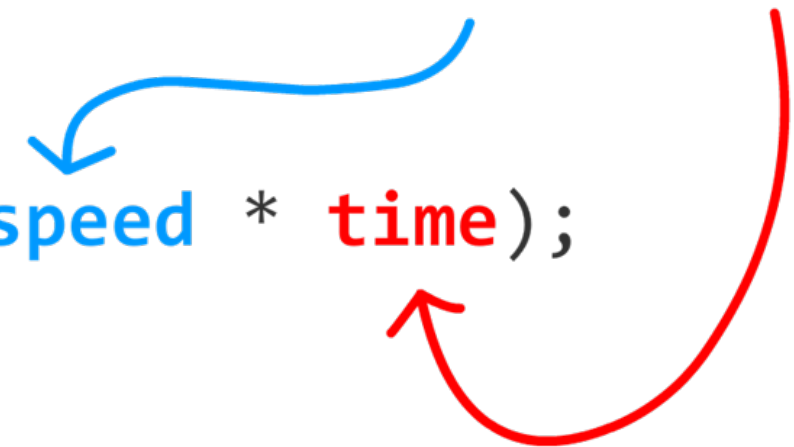
Passing arguments

The **showDistance** function takes two arguments: **speed**, **time**:

```
function showDistance(speed, time) {  
    alert(speed * time);  
}  
  
showDistance(10, 5);  
showDistance(85, 1.5);  
showDistance(12, 9);  
showDistance(42, 21);
```

the function

```
function showDistance(speed, time) {  
    alert(speed * time);  
}
```

A diagram illustrating the flow of data in a JavaScript function. A blue arrow points from the 'speed' parameter in the function signature to the 'speed' variable in the function body. A red arrow points from the 'time' parameter in the function signature to the 'time' variable in the function body.

Source: JavaScript Absolute Beginner's Guide by Kirupa Chinnathambi

the function call

```
showDistance(10, 5);
```

the function

```
function showDistance(speed, time) {  
    alert(speed * time);  
}
```

Source: JavaScript Absolute Beginner's Guide by Kirupa Chinnathambi

Returning Data

```
function getDistance(speed, time) {  
    let distance = speed * time;  
    return distance;  
}  
  
let myDistance = getDistance(10, 5);  
alert(myDistance);
```

The **return** keyword allows you to send data back to whatever called your function in the first place.

Once your **function** hits the **return** keyword, it stops everything it is doing at that point, returns whatever value you specified to the caller, and exits the function only. It does not exit the program!!!!

No code in your function after **return** will run.

```
function getDistance(speed, time) {  
  let distance = speed * time;  
  return distance;  
  
  if (speed < 0) {  
    distance *= -1;  
  }  
}
```

Time for “Challenges”!

Challenges

- **TotalDistance**
- **SalesTax**
- **MPG**
- **TipTaxTotal**
- **C2F**
- **IngredientAdjuster**

Homework Introduction

(peek under the hood)

- **Review Riley's Ranking Calculator:**
 - Let's look at the functions

Diamond Peer Teacher Ariella Izbinsky

[Total Distance Walkthrough](#)

FOX
MIS

Diamond Peer Teacher Ariella Izbinsky



[Ingredient Adjuster Walkthrough](#)

FOX
MIS

Diamond Peer Teacher Anna Boykis

[Males and Females Percentage Walkthrough](#)

FOX
MIS

Diamond Peer Teacher Lauren Quinn

[Planting Grapevines Walkthrough](#)

Diamond Peer Teacher Quinten Powers

[Compound Interest Walkthrough](#)

More to Come

Prepare with Readings & Videos before our next class!!!