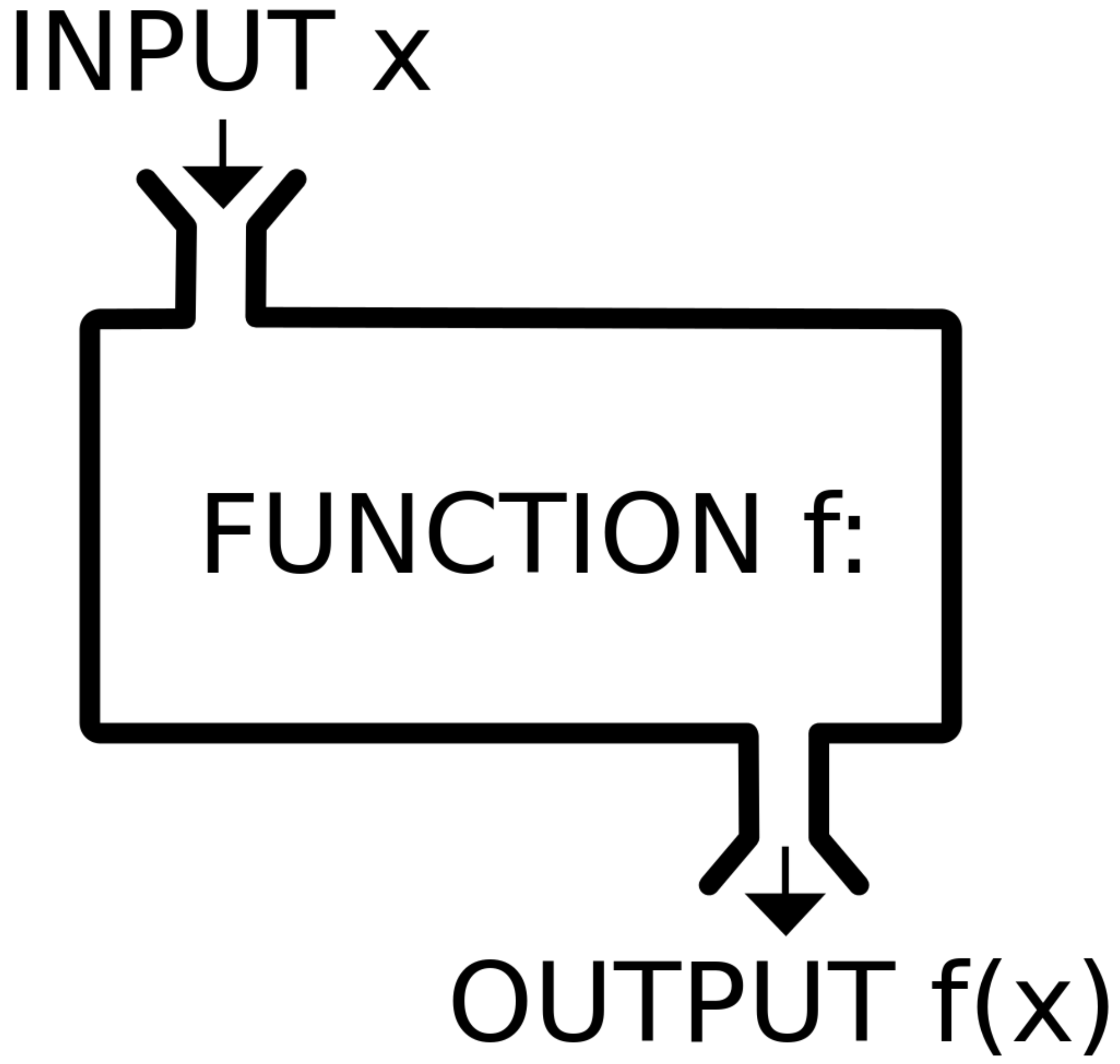
S1 Z2

Functions


INPUT x
FUNCTION f:
OUTPUT f(x)

Functions

- Basics
- Parameters
- Return

Functions

```
function sayHello() {

}
```

Functions

```
function sayHello() {

   console.log('Hello there’);

}
```

Functions

```
function sayHello() {
   console.log('Hello there');
}
sayHello(); //Hello there
```

Functions

```
function showValue(x){
   console.log('Value is: '+x);
}

showValue(2); //  Value is: 2
showValue('Karol’); //  Value is: Karol
```

Functions

```
function showSum(x, y) {
  let sum = x + y;
  console.log("Sum equels :" + sum);
  console.log("Is of type :" + typeof sum);
}
showSum(2);
showSum(2, 3);
showSum(2, 3, 4);
showSum("karol", 2);
showSum(2, "karol");
showSum("karol", "rogowski");
```

Functions

```
let var1 = 2;
let var2 = 3;

function showSum2(x,y){
   let sum = x + y;
   console.log('Sum equels :' + sum); //Sum equels :5
   console.log('Is of type :'+typeof(sum)); // Is of type :number
   y = y+x;
   console.log(y); // 5
}

showSum2(var1, var2);
console.log(var2); // 3
```

Functions

```
function getSum(x,y){
    let result = x + y;
    return result;
}

let var1 = getSum(2,3);
console.log('Sum equels :' + var1); //Sum equels :5
console.log('Is of type :'+typeof(var1)); // Is of type :number

let var2 = getSum(2,'Karol');
console.log('Sum equels :' + var2); //Sum equels :2Karol
console.log('Is of type :'+typeof(var2)); //Is of type :string

let var3 = getSum('Karol','Rogowski');
console.log('Sum equels :' + var3); //Sum equels :KarolRogowski
console.log('Is of type :'+typeof(var3)); //Is of type :string
```

Functions

```
function exampleFunction(){
   console.log("exampleFunction executed");
   let x = 10;
}

exampleFunction();
console.log(x); //x is not defined
```

FIELD & STREAM

Scope

- **Scope -** refers to the visibility of **variables**. In other words, which parts of your program can see or use it. Normally, every **variable** has a global **scope**. Once defined, every part of your program can access a **variable**. It is very useful to be able to limit a **variable's scope** to a single function.

Functions

```
let x = 5;
function exampleFunction() {
   console.log("exampleFunction executed"); //exampleFunction executed
   console.log(x); //5
}

exampleFunction();
console.log(x); //5
```

Functions

```
let x =5;

function exampleFunction(){
   console.log("exampleFunction executed");
   //exampleFunction executed
   let x = 10;
   console.log(x); //10
}

exampleFunction();
console.log(x); //5
```

Functions

```
let x =5;

function exampleFunction(){
   console.log("exampleFunction executed");
   //exampleFunction executed
   x = 10;
   console.log(x); //10
}

exampleFunction();
console.log(x); //10
```

Functions

```
let x =5;

function exampleFunction(){
    let x =1;
    console.log("exampleFunction executed");
    //exampleFunction executed
    x = 10;
    console.log(x); //10
}

exampleFunction();
console.log(x); //5
```

Objects

Objects

- Basics
- Objects + Functions
- Grouped Objects
- Out of the box

Objects

```
let book = {
   title: 'LOTR',
   pages: 2745,
   hardcover: true
}
```

Objects

```
let book = {
   title: 'LOTR',
   pages: 2745,
   hardCover: true
};

console.log(book.title); // LOTR
console.log(book.pages); // 2745
console.log(book.hardCover); // true
```

Objects

```
let book = {
  title: "LOTR",
  pages: 2745,
  hardCover: true,
  characters: ["Merry", "Pippin"]
};

console.log(book.title); // LOTR
console.log(book.pages); //2745
console.log(book.hardCover); // true
book.characters.push("Sam");
console.log(book.characters); //(3) ['Merry', 'Pippin', 'Sam']
```

Objects

```
let book1 = {
  title: "Karol",
};
let book2 = book1;
let book3 = {
  title: "Karol",
};
book1.title = "Adam";
book2.title = "Rogowski";
book1.rrr = 23;
console.log(book1 == book2); // true
console.log(book1 == book3); // false
```

Objects

```
let book = {
   title: 'LOTR',
   pages: 2745,
   hardCover: true
};

function showBookInfo(bookObject){
   console.log(bookObject.title); // LOTR
   console.log(bookObject.pages); // 2745
   console.log(bookObject.hardCover); // true
}

showBookInfo(book);
```

Objects

```
let book = {
   title: 'LOTR',
   pages: 2745,
   hardCover: true
};

function changeCover(book){
   book.hardCover = !book.hardCover;
   console.log('Cover changed’); //Cover changed
}

changeCover(book);
showBookInfo(book); // LOTR, 2745, false
```

Objects

```
let books = [
    {
        title: 'LOTR',
        pages: 2745,
        hardCover: true
    },
    {
        title: 'Witcher',
        pages: 1266,
        hardCover: false
    },
    {
        title: 'Sherlock Holmes',
        pages: 1950,
        hardCover: true
    }
];
```

Objects

```
for(let i = 0; i < books.length; i++){
   showBookInfo(books[i]);
}

books.forEach(function(book) {
   showBookInfo(book);
});
```

Out of the box

- Math
- Date
- String
- Number
- Error
- Function

Math

```
Math.max();
Math.min();
Math.floor();
Math.pow();
Math.PI;
Math.SQRT2;
Math.LOG10E;
Math.E;
```

Date

```
let dateObj = new Date();
console.log(dateObj);
console.log(dateObj.toDateString()); //Tue Oct 20 2020
console.log(dateObj.toLocaleDateString());
dateObj.setMonth(11);
console.log(dateObj);
dateObj.setMonth(12);
console.log(dateObj);
```

String

```
let str = "Karol";
console.log(str.substr(2, 2)); //ro
console.log(str.repeat(3)); // KarolKarolKarol
console.log(str.includes("o")); // true
console.log(str.anchor("test")); //<a name="test">Karol</a>
console.log(str.length); // 5
```

Number

```
let k = 2.12345;
console.log(k.toExponential(3)); //2.123e+0
console.log(k.toFixed(3)); //2.123
console.log(k.toPrecision(3)); //2.12

k = 4;
console.log(Number.isInteger(k)); //true
console.log(Number.isNaN(k)); // false
console.log(Number.isSafeInteger(k)); //true
```