Functions

Defining Functions

Function Parameters

Chapter 9, Examples 9-1, 9-2, 9-8

Chapter 5, Example 5-14
Function itself is not new to us

- We’ve been using *(calling)* functions...
  - to create canvas (i.e., createCanvas)
  - to draw shapes (e.g., rectangle, ellipse, ...)
  - to set attributes (e.g., colors, strokeWeight,...)
  - to react when event occurs (e.g., mousePressed, keyPressed, ...)
  - ...

CS 105 - Functions
They are functions others have created

- Others created them in advance as they represent features and actions that would be needed to create dynamic, interactive sketches.
But ...

- we need to create our own **custom functions** at times
Because

- we want ...
  - to improve readability of code
  - to reuse lines of code (to avoid repetition)
Because

- we want ...
  - to improve readability of code
  - to reuse lines of code (to avoid repetition)
How do we improve readability of code?

- What does the following lines of code represent?

```cpp
line(0, 0, 5, 0);
line(5, 0, 5, 5);
line(5, 5, 0, 0);
```

- It is not immediately clear what the drawing would look like (we have to draw them out to be clear)

=> readability of code is **bad**
Improving readability of code - comment

- We can use comment to explain what the chunk of code represents or does

```cpp
// draw triangle
line(0, 0, 5, 0);
line(5, 0, 5, 5);
line(5, 5, 0, 0);
```
Improving readability of code - function

- We can also create function and use name (e.g., drawTriangle) to represent what it does

```
drawTriangle(0, 0, 5, 0, 5, 5);
line(0, 0, 5, 0);
line(5, 0, 5, 5);
line(5, 5, 0, 0);
```
Improving readability of code - function

(1) create custom function `drawTriangle()`

```javascript
function drawTriangle(x1, y1, x2, y2, x3, y3) {
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
}
```

(2) use `drawTriangle()`

```javascript
drawTriangle(0, 0, 5, 0, 5, 5);
    line(0, 0, 5, 0);
    line(5, 0, 5, 5);
    line(5, 5, 0, 0);
```
Improving readability of code - function

(1) create custom function `drawTriangle()`

```javascript
function drawTriangle(x1, y1, x2, y2, x3, y3) {
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
}
```

(2) use `drawTriangle()`

```javascript
drawTriangle(0, 0, 5, 0, 5, 5);
```
Improving readability of code - comment or function?

- **function** offers more than **comment** (e.g., reusability, modularity)

- Recommended practice:

```python
if (code block is reused OR code block is long)
    function
else
    either comment or function is fine
```
```javascript
function draw() {
    // call the tree function
    tree();
}

// draw tree
// with bottom of trunk at 40, 90
function tree() {
    ...
}
```

Starter: [https://editor.p5js.org/sanghosuh/sketches/IPCaTCzg](https://editor.p5js.org/sanghosuh/sketches/IPCaTCzg)

[https://editor.p5js.org/sanghosuh/sketches/IKcseWrg](https://editor.p5js.org/sanghosuh/sketches/IKcseWrg)
Because

- we want ...
  - to improve readability of code
  - to reuse lines of code (to avoid repetition)
Because

- we want ...
  - to improve readability of code
  - to reuse lines of code (to avoid repetition)
When do we want to reuse lines of code?

- e.g., when drawing many of the same objects

```javascript
// sails
triangle(60, 15, 60, 70, 30, 70);
triangle(65, 15, 65, 70, 85, 70);

// hull
quad(20, 73, 30, 85, 86, 85, 93, 73);
```

Function is **not necessary**

```javascript
// sails
triangle(150, 15, 150, 70, 120, 70);
triangle(155, 15, 155, 70, 175, 70);

// hull
quad(110, 73, 120, 85, 176, 85, 183, 73);
```

```javascript
// sails
triangle(250, 15, 250, 70, 220, 70);
triangle(255, 15, 255, 70, 275, 70);

// hull
quad(210, 73, 220, 85, 276, 85, 283, 73);
```

Function is **necessary**
// draw 1st tree
// with bottom of trunk at 40, 90
function tree1() {
    ...
}

// draw 2nd tree
// with bottom of trunk at 100, 90
function tree2() {
    ...
}

// draw 3rd tree
// with bottom of trunk at 160, 90
function tree3() {
    ...
}

function draw() {
    ...
    tree1();
    tree2();
    tree3();
}
// draw 1st tree
// with bottom of trunk at 40, 90
function tree1() {
  ...
}

// draw 2nd tree
// with bottom of trunk at 100, 90
function tree2() {
  ...
}

// draw 3rd tree
// with bottom of trunk at 160, 90
function tree3() {
  ...
}

function draw() {
  ...
  tree1();
  tree2();
  tree3();
}

Inefficient because...
- we need to create 3 functions
- each function draws a tree
  only at specified position
- for another tree at a different position,
  we have to create additional function
- not a scalable solution
// draw tree
// with bottom of trunk at x, y
function tree(x, y) {
  ...
}

one function that
draws tree at any position
landscape2 - multiple trees

...  
// use the tree function to  
// create a forest  
tree(40, 90);  
tree(100, 90);  
tree(160, 90);  
}  

// draw tree  
// with bottom of trunk at x, y  
function tree(x, y) {  
  ...  
}  

Starter: https://editor.p5js.org/sanghosuh/sketches/IKcseWrg  
https://editor.p5js.org/cs105/sketches/KRLY8p0mS
(Recap) Anatomy of Function Definition *without* Parameters

```plaintext
function tree() {
  noStroke();
  // trunk
  fill("#5A3709"); // brown
  quad(35, 40, 45, 40, 50, 90, 30, 90); // leaves
  fill("#5CBF17"); // green
  ellipse(40, 30, 55, 55);
}
```

- means "define a new function"
- function name
- empty parentheses means "no function parameters"
(Recap) Anatomy of Function Definition with Parameters

```javascript
function tree(x, y) {
    noStroke();
    // trunk
    fill("#5A3709");  // brown
    quad(x - 5, y - 50, x + 5, y - 50, x + 10, y, x - 10, y);
    // leaves
    fill("#5CBF17");  // green
    ellipse(x, y - 60, 55, 55);
}
```

function parameters are used as variables inside code block

means “define a new function”

function name

function parameters
And **where** do we define custom functions?

- Define your functions "outside" setup and draw
  - the same place global variables are declared

```javascript
function setup() {
    ...
}

function draw() {
    ...
}
```

But let's define functions below `setup()` and `draw()` for consistency (readability)
(Example) **where** do we define custom functions?

```javascript
function setup() {
    frameRate(1);
    print("setup");
}

function draw() {
    print("draw start");
    myFunction();
    print("draw end");
}

function myFunction() {
    print(" myFunction");
}
```
NOTE: function definition order doesn’t matter

```javascript
function printB() {
    print("B");
}

function setup() {
    printA();
    print("Setup");
    printB();
}

function printA() {
    print("A");
}
```

But let’s define functions below `setup()` and `draw()` for consistency (readability)

https://editor.p5js.org/cs105/sketches/40THSvHIm
function setup() {
    frameRate(1);
    print("setup");
}

function draw() {
    print("draw start");
    myFunction();
    print("draw end");
}

function myFunction() {
    print(" myFunction");
}
function setup() {
  frameRate(1);
  print("setup");
}

function draw() {
  print("draw start");
  myFunction();
  print("draw end");
}

function myFunction() {
  print("myFunction");
}
function setup() {
    frameRate(1);
    print("setup");
}

function draw() {
    print("draw start");
    myFunction();
    print("draw end");
}

function myFunction() {
    print("myFunction");
}
function setup() {
    frameRate(1);
    print("setup");
}

function draw() {
    print("draw start");
    myFunction();
    print("draw end");
}

function myFunction() {
    print(" myFunction");
}
function setup() {
  frameRate(1);
  print("setup");
}

function draw() {
  print("draw start");
  myFunction();
  print("draw end");
}

function myFunction() {
  print(" myFunction");
}
function setup() {
    frameRate(1);
    print("setup");
}

function draw() {
    print("draw start");
    myFunction();
    print("draw end");
}

function myFunction() {
    print("myFunction");
}
We refer to variables in function definition as **parameters**.
We refer to values passed to parameters as **arguments**.

```javascript
function tree(x, y) {
    ...
}
```

```javascript
tree(40, 150);
```

Defining function

Calling function
vehicle

// draw a vehicle
// x, y position of centre bottom
// hue colour hue to fill the car body
// (0 to 360 degrees)

function vehicle(x, y, hue) {
  ...
}

https://editor.p5js.org/cs105/sketches/ylnubokJs
What does the following code print?

```javascript
function setup() {
  let a = 10;
  print(a); // ?
  f(a);
  print(a); // ?
}

function f(b) {
  b = b + 1;
}
```

```javascript
function setup() {
  let a = 10;
  print(a); // ?
  f(a);
  print(a); // ?
}

function f(b) {
  b = b + 1;
}
```

? ?
Function attributes are “passed-by-value”

- “passed-by-value” means the function parameter is a copy of the argument variable used in the function call.
- Changing the value of a function parameter inside the function code block will not change the value of the variable used as an argument.

```javascript
function setup() {
    let a = 10;
    print(a); // 10
    f(a);
    print(a); // 10
}

function f(b) {
    b = b + 1;
}
```
Functions and Scope

- Custom functions can access global variables, (just like setup and draw)
- Custom functions can have local variables, (just like setup and draw)

```javascript
let a = 20;

function setup() {
  addOne();
  print(a); // 21
}

function addOne() {
  a = a + 1;
}
```

```javascript
function setup() {
  printTen();
}

function printTen() {
  let l = 10;
  print(l); // 10
}
```