Day 4

Functions() {

}
Agenda

1. What is a function?
2. Why do we use functions?
3. Structure of a function
4. The map( ) function
Functions

Like sentences.
To repeat with different words.
What is a function?

A function is a set of organised commands used to perform a specific action.

Examples:
- Draw rectangle
- Get the time
- Plan a vacation

Task {
  Step to complete the task
  Step to complete the task
  Step to complete the task
  Step to complete the task
  Step to complete the task
  Step to complete the task
  }

Why do we use functions?

- Reusability
- Organization
- Abstraction
Reusability

- Define repeated tasks
- Write less code

```java
Main Program {
    Take out mobile phone;
    Turn it on;
    Look at the time;
    Go to class;
    Give presentation;
    Take out mobile phone;
    Turn it on;
    Look at the time;
    Have lunch;
    Take out mobile phone;
    Turn it on;
    Look at the time;
    Go to buy a coffee;
    Meet for group project;
}
```
Reusability

- Create a function **outside of main code**
- “Call” the function as many times as you’d like

```cpp
Main Program {
    Get the time;
    Go to class;
    Give presentation;
    Get the time;
    Have lunch;
    Get the time;
    Go to buy a coffee;
    Meet for group project;
}

Get the time {
    Take out mobile phone;
    Turn it on;
    Look at the time;
}
```
Organization

- Break many lines of code into smaller, digestible “building blocks”
- Structure in a way that is easy to read, review, and debug for yourself and for others
Main Program {

Decide;
Research;
Decide;
Research;
Book;
Prepare;

Decide {
    Choose destination;
    Set dates;
    Establish budget;
}

Research {
    Read travel guides;
    Ask for recommendations;
    Compare prices;
    Create itinerary;
}

Book {
    Buy flights;
    Book accommodations;
    Make reservations;
}

Prepare {
    Apply for visa;
    Renew passport;
    Get vaccinations;
    Buy travel insurance;
}
Abstraction

- Let’s you carry out a task **without knowing the details of the implementation**
  e.g. Draw a rectangle without needing to know the steps to create each line

<table>
<thead>
<tr>
<th>Shape</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>createShape()</td>
<td>Setting</td>
</tr>
<tr>
<td>loadShape()</td>
<td>background()</td>
</tr>
<tr>
<td>PShape</td>
<td>clear()</td>
</tr>
<tr>
<td></td>
<td>colorMode()</td>
</tr>
<tr>
<td>2D Primitives</td>
<td>fill()</td>
</tr>
<tr>
<td>arc()</td>
<td>noFill()</td>
</tr>
<tr>
<td>ellipse()</td>
<td>noStroke()</td>
</tr>
<tr>
<td>line()</td>
<td>stroke()</td>
</tr>
<tr>
<td>point()</td>
<td></td>
</tr>
<tr>
<td>quad()</td>
<td></td>
</tr>
<tr>
<td>rect()</td>
<td></td>
</tr>
<tr>
<td>triangle()</td>
<td></td>
</tr>
</tbody>
</table>

Some of Processing’s built-in functions
[processing.org/reference](processing.org/reference)
Structure

- **Parameters**: the “materials” we provide for the function
- **Body**: what we want the function to do
- **Return value**: what we want the function to give back to us

```plaintext
return-type function-name (function-parameters) {
    // body
    Local variables;
    Variable definitions;
    return-value;
}

// example
int sum (int a, int b) {
    int result;
    result = a + b;
    return result;
}
```
- Used to "call" the function elsewhere in the program
- Best practice: **be specific**; designate a name that **describes** the action
  e.g. `sayMyName()` vs. `name()`
- The **format of data** that the function will return

- The **return value** must be consistent with function’s **return type**
e.g. `int` returns an integer, `bool` returns true or false

```cpp
int functionName() {  
    // do this  
    // do that  
    // return value of type int
}

bool functionName() {  
    // do this  
    // do that  
    // return value of type bool
}

String functionName() {  
    // do this  
    // do that  
    // return value of type String
}
```
- If **no value** needs to be returned, use **void**

  e.g. `setUp()`, `draw()`, and `mousePressed()`
```java
String whatMyName() {
    return "Rihanna";
}

// call the function
String myName = whatMyName();
println(myName);

// or
println(whatMyName());

void sayMyName() {
    println("Destiny’s Child");
}

// call the function
sayMyName();
sayMyName();
```
function parameters

- Are **values, and their types**, passed into a function (the "**materials**")
- Gives the function **flexibility**
- A function can take **multiple** parameters, but
- **Not all** functions require parameters
e.g. `setup()`, `draw()`

```c
// built-in function
void rect(float posX, float posY, float width, float height) {
    // definition
}

// call the function
rect(10, 10, 20, 40);
rect(5.5, 7.5, 3.25, 5.25);
```
- **Global variables**: defined in the main program. They can be used by any function.

- **Local variables**: defined within a function, including parameters. They can’t be used outside that function.
- A function can also be called within another function
What’s logic of this sequence of numbers?

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

What’s logic of this sequence of numbers?

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Fibonacci Numbers

\[ X(n) = X(n-2) + X(n-1) \]

\[ X(0) = 0, \ X(1) = 1 \]
What’s logic of this sequence of numbers?

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Fibonacci Numbers

\[ X(n) = X(n-2) + X(n-1) \]
\[ X(0) = 0, \; X(1) = 1 \]
Break
map( ) function

```c
map(value, start1, stop1, start2, stop2);
float x = map(mouseX, 0, 800, 50, 150);
```
Let’s code.
Task

if mouse clicked draw a circle at mouse position;
as mouse moves towards the right edge of canvas, the radius grows
Scale the radius of the circle to be between 30 and 150, based on the mouse position
Task

draw your flow chart first
check with your partner
Homework
Continue working on text adventure

- Finish 3 stages
- Add a function
- Add pictures
- Make Slides
lamp doesn’t work

- Lamp plugged in?
  - No: plug in lamp
  - Yes: bulb burned out?
    - Yes: replace bulb
    - No: buy new lamp
Flow chart

- Start
- End
- Process
- Input/Output
- Conditional judgement
  - Yes
  - No
It’s your turn