Module 03

Input / Output
Which of these functions completes the code below to print hooray?

```javascript
if (addOne(2) === 3) {
    print("Hooray! ");
}

// (C) function addOne(x) {
//    print(x + 1);
//}

// (D) function addOne(x) {
//    x = x + 1;
//}

// (A) function addOne(x) {
//    x + 1;
//}

// (B) function addOne(x) {
//    return x + 1;
//}
```
We can write more interesting programs when we can exchange information with the outside world!

Problem with the outside world: there’s a lot of it.
<table>
<thead>
<tr>
<th>Location</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>98.2 GB</td>
</tr>
<tr>
<td>Library</td>
<td>54.7 GB</td>
</tr>
<tr>
<td>Movies</td>
<td>19.0 GB</td>
</tr>
<tr>
<td>Music</td>
<td>17.7 GB</td>
</tr>
<tr>
<td>Pictures</td>
<td>11.3 GB</td>
</tr>
<tr>
<td>Videos</td>
<td>10.4 GB</td>
</tr>
<tr>
<td>Documents</td>
<td>9.8 GB</td>
</tr>
<tr>
<td>Downloads</td>
<td>7.9 GB</td>
</tr>
<tr>
<td>Items</td>
<td>2.9 GB</td>
</tr>
<tr>
<td>Personal</td>
<td>2.4 GB</td>
</tr>
<tr>
<td>User Archive</td>
<td>10.032 MB</td>
</tr>
<tr>
<td>User</td>
<td>9.637 MB</td>
</tr>
<tr>
<td>.tar.gz</td>
<td>4.004 MB</td>
</tr>
<tr>
<td>Backup</td>
<td>1.246 MB</td>
</tr>
<tr>
<td>FederalBackup</td>
<td>0.954 MB</td>
</tr>
<tr>
<td>Transporter</td>
<td>4.51 MB</td>
</tr>
<tr>
<td>Cache</td>
<td>2.35 MB</td>
</tr>
<tr>
<td>Render</td>
<td>1.2 MB</td>
</tr>
<tr>
<td>Config</td>
<td>1.1 MB</td>
</tr>
<tr>
<td>P collegiate</td>
<td>1.1 MB</td>
</tr>
<tr>
<td>Thumbnail</td>
<td>1.1 MB</td>
</tr>
<tr>
<td>Data</td>
<td>1.1 MB</td>
</tr>
<tr>
<td>Total</td>
<td>32.4 MB</td>
</tr>
</tbody>
</table>
Use the **Sketch Folder** as a gateway to the outside world.
Reading and writing text
Marley was dead: to begin with. There is no doubt whatever about that. The register of his burial was signed by the clergyman, the clerk, the undertaker, and the chief mourner. Scrooge signed it: and Scrooge's name was good upon 'Change, for anything.

Mind! I don't mean to say that I know, of my own knowledge, what there is particularly dead about a door-nail. I might have been inclined, myself, to regard a coffin-nail as the deadest piece of ironmongery in the trade. But the wisdom of our ancestors had, apparently, looked further into it than I had. It is not much used now.

Scrooge knew he was dead? Of course he did. How could it be otherwise? Scrooge and he were partners for I don't know how many years. Scrooge was his sole executor, his sole administrator, his sole assign, his sole residuary legatee, his sole friend.

The mention of Marley's funeral brings me back to the point I started from. There is no doubt that Marley was dead. This must be distinctly understood, or nothing wonderful can come of the story I am going to relate. If we were not perfectly convinced that he was dead, there is no possibility of our being able to superinduce those extraordinary circumstances."
From: Rishabh Moudgil <rishabh.moudgil@uwaterloo.ca>
To: Craig Kaplan <csk@uwaterloo.ca>
CC: Kevin Harrigan <kevinh@uwaterloo.ca>, Kristina Bayda <kabayda@uwaterloo.ca>, Travis Bartlett <travis.bartlett@uwaterloo.ca>
Subject: A01 Marking Scheme

Date: Tue, 17 Jan 2017 20:57:36 +0000
Message-ID: <748888CA42FDF349AF07A8978DDED060281C9EC0@connmbx02>
Accept-Language: en-CA, en-US
Content-Language: en-CA
X-MS-Exchange-Organization-AuthAs: Internal
X-MS-Exchange-Organization-AuthMechanism: 04
X-MS-Exchange-Organization-AuthSource: connhub1.connect.uwaterloo.ca
X-MS-Has-Attach:
X-MS-Exchange-Organization-SCL: -1
X-MS-TNEF-Correlator:
Content-Type: multipart/alternative;
     boundary="_000_748888CA42FDF349AF07A8978DDED060281C9EC0connmbx02_"
MIME-Version: 1.0

--_000_748888CA42FDF349AF07A8978DDED060281C9EC0connmbx02_
Content-Type: text/plain; charset="Windows-1252"
Content-Transfer-Encoding: quoted-printable
<table>
<thead>
<tr>
<th>Date</th>
<th>Low temperature</th>
<th>High temperature</th>
<th>Precipitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-Jan-14</td>
<td>-15.6</td>
<td>-8.9</td>
<td>0.1</td>
</tr>
<tr>
<td>02-Jan-14</td>
<td>-17.7</td>
<td>-15.1</td>
<td>0.1</td>
</tr>
<tr>
<td>03-Jan-14</td>
<td>-23.4</td>
<td>-13.1</td>
<td>0</td>
</tr>
<tr>
<td>04-Jan-14</td>
<td>-12.7</td>
<td>-2.5</td>
<td>0</td>
</tr>
<tr>
<td>05-Jan-14</td>
<td>-3.7</td>
<td>-1.2</td>
<td>19.1</td>
</tr>
<tr>
<td>06-Jan-14</td>
<td>-19.6</td>
<td>-2.1</td>
<td>7.7</td>
</tr>
<tr>
<td>07-Jan-14</td>
<td>-26.1</td>
<td>-18.7</td>
<td>1.5</td>
</tr>
<tr>
<td>08-Jan-14</td>
<td>-19.1</td>
<td>-11.1</td>
<td>0</td>
</tr>
<tr>
<td>09-Jan-14</td>
<td>-22.2</td>
<td>-8.3</td>
<td>0</td>
</tr>
<tr>
<td>10-Jan-14</td>
<td>-8.3</td>
<td>2.4</td>
<td>0</td>
</tr>
<tr>
<td>11-Jan-14</td>
<td>0.3</td>
<td>5.4</td>
<td>26.4</td>
</tr>
<tr>
<td>12-Jan-14</td>
<td>-0.8</td>
<td>1.3</td>
<td>0</td>
</tr>
<tr>
<td>13-Jan-14</td>
<td>0.4</td>
<td>5.8</td>
<td>0.2</td>
</tr>
<tr>
<td>14-Jan-14</td>
<td>-2.5</td>
<td>3.3</td>
<td>0</td>
</tr>
<tr>
<td>15-Jan-14</td>
<td>-8.5</td>
<td>-0.4</td>
<td>1.4</td>
</tr>
<tr>
<td>16-Jan-14</td>
<td>-8.7</td>
<td>-4.2</td>
<td>2.7</td>
</tr>
<tr>
<td>17-Jan-14</td>
<td>-8</td>
<td>-0.3</td>
<td>3.9</td>
</tr>
<tr>
<td>18-Jan-14</td>
<td>-10.1</td>
<td>-4.6</td>
<td>1.7</td>
</tr>
</tbody>
</table>
Reading text

Reading text from a file can be quite painful in many programming languages. JavaScript p5 keeps it simple:

```javascript
myArr = loadStrings(filename);
```

Load a text file from the data folder. Break it up into lines and return an array of Strings, one per line.
let lines = [];
function preload() {
    lines = loadStrings( "data/mywords.txt" );
}

function setup() {
    createCanvas( 600, 600 );
    textSize(24);
}

function draw() {
    background(220);
    for (let i = 0; i < lines.length; i++) {
        text( lines[i], 10, 30 + (i * 30));
    }
}
function preload() {
    lines = loadStrings("data/shoppinglist.txt");
}

function setup() {
    createCanvas(600, 600);
    textSize(24);
}

function draw() {
    background(220);
    for (let i = 0; i < lines.length; i++) {
        text(lines[i], 10, 30 + (i * 30));
    }
}
Breaking up lines

A line in a file may contain lots of individual chunks of data separated by whitespace. We’d like to break lines into words, just as we broke files into lines. Two techniques: (see example next slide)

```
.split()
.splitTokens()
```

Turn a line of text into an array of “words” (any non-whitespace characters separated by whitespace).

(Note that join() can reassemble individual strings into a single result.)
function setup() {
    let s = "hello out there";
    print(s); // hello out there

    let myArr1 = [];
    myArr1 = s.split(" ");
    print(myArr1); // myArr1 is of length 3
        // ["hello", "out", "there"]

    let myArr2 = [];
    myArr2 = splitTokens(s, " ");
    print(myArr2); // myArr2 is of length 3
        // ["hello", "out", "there"]
}
Sometimes we need strings converted to numbers

Assume we have a shopping list and we want to calculate the total weight of the items in the list.

<table>
<thead>
<tr>
<th>Item</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>2 kg</td>
</tr>
<tr>
<td>banana</td>
<td>5 kg</td>
</tr>
<tr>
<td>potato</td>
<td>4 kg</td>
</tr>
<tr>
<td>onion</td>
<td>2 kg</td>
</tr>
</tbody>
</table>
let lines = [];
let words = [];
let nextI;
function preload() {
    lines = loadStrings("data/shoppinglist.txt");
}
function setup() {
    createCanvas(600, 600);
    textSize(24);
}
function draw() {
    background(220);
    let totalKG = 0;
    for (let i = 0; i < lines.length; i++) {
        words = splitTokens(lines[i], " ");
        text(lines[i], 10, 30 + (i * 30));
        totalKG = totalKG + int(words[1]);
        nextI = i + 1;
    }
    text("Total KG: " + totalKG, 10, 30+(nextI * 30));
}
SpeedReader Example

• Read in a text file.

• Make one big long list (array) of “words”
  • Words may contain punctuation in this example

• Display one word at a time
let lines = [];
let words = [];
let index = 0;

function preload() {
    lines = loadStrings("data/marley.txt");
}

function setup() {
    createCanvas(400, 200);
    textSize(50);
    textAlign(CENTER);
    fill(255);
    let allLines = join(lines, " ");
    words = splitTokens(allLines);
    frameRate(1);
}

function draw() {
    background(80);
    text(words[index], width/2, height/2);
    index = (index + 1) % words.length;
}
Marley was dead: to begin with. There is no doubt whatever about that. The register of his burial was signed by the clergyman, the clerk, the undertaker, and the chief mourner. Scrooge signed it: and Scrooge's name was good upon 'Change, for anything he chose to put his hand to. Old Marley was as dead as a door-nail. Mind! I don't mean to say that I know, ............
How many times does the word “the” occur?

Get rid of or ignore punctuation

`words = splitTokens(allLines, " .;<>?!@#$%^&*()");`

Capitalization does matter (“the” and “The” both count)

`let wordInUpperCase = words[i].toUpperCase();`
let count = 0;
for (let i = 0; i < words.length; i++) {
    let wordInUpperCase = words[i].toUpperCase();
    if (wordInUpperCase === "THE") {
        count = count + 1;
    }
}
print("The count: ", count);
Writing text to a File

We know we can use `print()` to send any text to the console.

A similar mechanism puts text into the file.

```python
saveStrings(list, textFilename);
```
Writing to a Text File

```javascript
let words = 'apple bear cat dog';
let list = [];
function setup() {
  createCanvas(100, 100);
  background(200);
  text('click here to save', 10, 10, 70, 80);
  let list = split(words, ' ');
}
function mousePressed() {
  if (mouseX > 0 && mouseX < width &&
      mouseY > 0 && mouseY < height) {
    saveStrings(list, 'nouns.txt');
  }
}
```
Reasons to write text

Logging: Create a permanent record of the behaviour of the program to review later.

Persistence: Store information about the program’s state in an external file so that the sketch can restart with that state later.

Workflow: create text output that can be read by another program for further processing.
Reading and writing images
Loading an Image

Let img;

function preload()
{
    img = loadImage( "data/boromir.jpg" );
}

let img;

function preload() {
    img = loadImage( "data/boromir.jpg" );
}

function setup() {
    createCanvas( 800, 400 );
}
function draw() {
    background(255);
    imageMode(CORNER);
    noTint();
    image(img, 0, 0);
    image(img, width - img.width, height - img.height);

    tint(255, 120, 120);
    imageMode(CENTER);
    image(img, width / 2, height / 2, 250, 250);
}
Image Loading Idiom

let img;

function preload() {
    img = loadImage("some_image.jpg");
}

function setup() {
    ...
}

function draw() {
    image(img, 0, 0, width, height);
    ...
}
You can also copy a region out of a source image, and scale it to any rectangle in the sketch window.

```c
    copy(img, sx, sy, sw, sh, dx, dy, dw, dh);
```
You can also copy a *region* out of a source image, and scale it to any rectangle in the sketch window.

```c
    copy(img, sx, sy, sw, sh, dx, dy, dw, dh);
```

The source image to copy pixels from
You can also copy a region out of a source image, and scale it to any rectangle in the sketch window.

```
copy(img, sx, sy, sw, sh, dx, dy, dw, dh);
```

A rectangle of pixels in the source image. Just like the arguments in a call to rect()
You can also copy a *region* out of a source image, and scale it to any rectangle in the sketch window.

```c
    copy(img, sx, sy, sw, sh, dx, dy, dw, dh);
```

A rectangle of pixels in the sketch window. Again, just like a call to `rect()`
copy(img, sx, sy, sw, sh, dx, dy, dw, dh);
Writing images

Take a screenshot.

```
save("filename.png");
```

Save the contents of the sketch window to an image with the given file name.
function keyPressed() {
    if (key === 's') {
        save("screen.png");
    }
}
Reading illustrations
**Raster image**: represented using a grid of pixels.

**Vector illustration**: represented using geometric paths.
**Raster image**: represented using a grid of pixels.
JPG, PNG, GIF, BMP, TIFF, …

**Vector illustration**: represented using geometric paths.
PDF, EPS, AI, SVG, …
Load a vector Image

let tiger;

function preload() {
    tiger = loadImage("data/tiger.svg");
}

function setup() {
    createCanvas(600, 600);
}

function draw() {
    background(220);
    image(tiger, 0, 0);
}
Using Sprite Files
Sprite: Compass
Sprite: Playing Cards

A 2 3 4 5 6 7 8 9 10 J Q K ? ♦️

A 2 3 4 5 6 7 8 9 10 J Q K ♡️

A 2 3 4 5 6 7 8 9 10 J Q K ♠️

A 2 3 4 5 6 7 8 9 10 J Q K ♣️