Reading: Rampup Programming Exercise - Variables, Methods and Functions

Rampups are extra programming exercises for you to practice new concepts. They should be especially helpful to those of you who have never heard of these concepts.

Before doing this assignment, you should try the short examples in the Reading: Try It! Using Variables, Methods and Functions.

This exercise should be done in the dukelearntoprogram.com javascript environment here.

Exercise 1 - Make a Phrase From Three Words

Write a function named **`phrase3words`** that puts three words together into a phrase that is of type string with blanks between the words. The function **`phrase3words`** has three parameters named **`value1`**, **`value2`** and **`value3.`** This function concatenates the words together into one string that has value1 first, followed by a blank, followed by value2, followed by a blank, followed by value3.

The code below has been started for you. It defines the function **`phrase3words`** and calls the function three times, printing the answer each time. You need to replace **[code here]** with the missing code.

CODE:

```
function phrase3words(value1, value2, value3) {
      var answer = [code here] ;
      return answer;
}

var result1 = phrase3words("smile","at","everyone");
print(result1);
var result2 = phrase3words("everyone","wave", "back");
print(result2);
var result3 = phrase3words("coding","is", "fun");
print(result3);
```

OUTPUT:

```
smile at everyone
everyone wave back
coding is fun
```

Exercise 2 - Format a name

Write a function named **`reformatName`** that puts a name together in a specific format. The function **`reformatName`** has two parameters named **`first`** and **`last,`** representing the first and last names. This function puts the names together into one string that has the last name first, followed by a comma and blank, and then followed by the first name.

For example, the call reformatName("Susan", "Rodger") returns the string "Rodger, Susan", and the call reformatName("Robert", "Duvall") returns the string "Duvall, Robert".
The function has been started below for you, with the body of the code missing.

CODE:

```
function reformatName(first, last) {
  // missing code
}

var result = reformatName("Susan", "Rodger");
print(result);
result = reformatName("Robert", "Duvall");
print(result);
```

OUTPUT:

```
Rodger, Susan
Duvall, Robert
```

Exercise 3 - Number of pixels in an image

Write a function named **`numberPixels`** that calculates the total number of pixels in an image. The function **`numberPixels`** has one parameter named **`namefile,`** which is a string that is the name of an image file. This function calculates and returns the total number of pixels in the specified image filename.

For example, the call numberPixels("chapel.png") returns 71148, and the call numberPixels("dinos.png") returns 2073600. Both of these are images on the dukelearntoprogram.com website. For each image on that website it also displays the size of that image. That website shows that chapel.png has 231 pixels in width and 308 pixels in height, and 231*308 is 71148 total pixels. That website also shows that dinos.png has 1920 pixels in with and 1080 pixels in height, and 1920*1080 is 2073600 total pixels.

The function has been started below for you. Note that you first need to create a SimpleImage from the filename. Then you can apply SimpleImage methods to the variable **`someImg`** such as **`getHeight()`** to get the number of pixels in the height of the image.

CODE:

```
function numberPixels(namefile) {
      var someImg = new SimpleImage(namefile);
      var height = someImg.getHeight();
      // missing code
}

var result = numberPixels("chapel.png");
print(result);
result = numberPixels("dinos.png");
print(result);
```

OUTPUT:

```
71148
2073600
```

Exercise 4 - Perimeter of an image

Write a function named **`perimeter`** that calculates the number of pixels in the perimeter of an image. The perimeter is the number of pixels on the edge of the image, from all four sides. The function **`perimeter`** has one parameter named **`imageName,`** which is a string that is the name of an image file. This function calculates and returns the perimeter in the specified image filename.

For example, the image "rodger.png" has 315 pixels in width and 424 pixels in height. That means it has 315 pixels on the bottom, 315 on the top, 424 on the right side and 424 on the left

side. The perimeter of this image is 315 + 315 + 424 + 424 = 1478. The call perimeter("rodger.png") returns 1478.
The function has been started below for you.

CODE:

```
function perimeter(imageName) {
      // missing code
}

print(perimeter("rodger.png"));
```

OUTPUT:

```
1478
```

Exercise 5 - Print the RGB values of a pixel

Write a function named **`printPixel`** that prints the red, blue and green values of a pixel, in that order, one on each line, and identifies each one. The function **`printPixel`** has three parameters, **`namefile,`** which is a string that is the name of an image file, and **`xpos`** and **`ypos`** that are numbers representing the x and y coordinates of the pixel location. Consider using the SimpleImage methods getRed, getGreen, and getBlue. Since this function is printing values, it does not need a return statement.

Note that in the image drewgreen.png, Drew is standing in the middle and the background is bright green. So the first pixel printed at x and y location (10,10) is near the edge and is bright green. For its red, green and blue values, it has all green (255), no blue (0) and only a tiny bit of red (1). The second pixel printed is in the middle of the image and is some part of Drew.

The function has been started below for you.

CODE:

```
function printPixel(nameImage, xpos, ypos) {
      // missing code
}

printPixel("drewgreen.png",10, 10);
printPixel("drewgreen.png",250, 500);
```

OUTPUT:

```
red is 1
green is 255
blue is 0
red is 102
green is 90
blue is 80
```

Exercise 6 - Sum of the RGB values for a Pixel

Write a function named **`sumPixel`** that calculates and returns the sum of the red, blue and green values of a pixel. The function **`sumPixel`** has three parameters, **`namefile,`** which is a string that is the name of an image file, and **`xpos`** and **`ypos`** that are numbers representing the x and y coordinates of the pixel location. Since this function is returning a value, it should NOT have a print statement in the function, and it should have a return statement.

Consider the image drewgreen.png. The pixel at location (250,500) has red component 102, green component 90 and blue component 80. The call `sumPixel("drewgreen.png", 250, 500)` should return 102+90+80 = 272. The pixel at location (10,10) has red component 1, green component 255 and blue component 0. The call `sumPixel("drewgreen.png", 10, 10)` should return 1 + 255 + 0 = 256.

The function has been started below for you.

CODE:

```
function sumPixel(nameOfImage, xpos, ypos) {
   // missing code
}

var answer = sumPixel("drewgreen.png", 250, 500);
print(answer);
answer = sumPixel("drewgreen.png",10, 10);
print(answer);
```

OUTPUT:
272
256