# Functions with Outputs

## Returning Values

**Exercise 1**
What value is returned by the following function calls?

(a)

```
def cost_with_taxes( cost ):
    final_price = cost * 1.1
    return final_price

cost_with_taxes( 25.00 )
```

**Answer:** The function returns `27.500000000000004` (or something very similar to this number) which is approximately `25.00 * 1.1`. The number returned looks a bit odd/extremely precise due to the fact that the computer is representing infinite data with finite precision- but that's a topic for another computer science course.

(b)

```
def rect_perimeter( w, l ):
    return 2*w + 2*l

rect_perimeter( 5, 3 )
```

**Answer:** 16 is returned by the function call.

(c)

```
def print_random_msg():
    language = "Python"
    print(language + " is named after a comedy series!")

print_random_msg()
```

**Answer:** This function does not return any output as noted by the lack of `return` statement. Thus, because it has no output, it doesn't return any data, and therefore has no return value.

(d)

```
def repeat_msg( msg, n_repeats ):
    print(msg * n_repeats)
    return

repeat_msg("ha",5)
```

**Answer:** The `return` statement is empty, that is, the function does not return any data. Thus, this function produces no return value.

**Exercise 2**
Write the following functions:

(a) `rect_area_from_dims( l, w )` : returns the area of a rectangle

`l`: the length of the rectangle

`w`: the width of the rectangle

Solution

```
# CMPT 140 - Functions with Outputs
# Topic(s): Function Composition
# Part (a)


def rect_area_from_dims( l, w ):
    """
    compute and return area of a rectangle
    l: rectangle's length
    w: rectangle's width
    return: area of the rectangle
    """
    return l * w
```

This code can be found in
`cmpt140-ch09-py/cmpt140_ch09_func_composition_a/cmpt140_ch09_func_composition_a.pyde`

(b) `rect_area_from_coords( tl_x, tl_y, br_x, br_y )` : returns the area of a rectangle.

`tl_x, tl_y`: (x,y) coordinates of the top-left corner

`br_x, br_y`: (x,y) coordinates of the bottom-right corner

Solution

```
# CMPT 140 - Functions with Outputs
# Topic(s): Function Composition
# Part (b)


def rect_area_from_coords( tl_x,tl_y,br_x,br_y ):
    """
    compute and return area of rectangle
    tl_x: rectangle's top-left x-coordinate
    tl_y: rectangle's top-left y-coordinate
    br_x: rectangle's bottom-right x-coordinate
    br_y: rectangle's bottom-right y-coordinate
    return: area of the rectangle
    """
    return (br_x - tl_x) * (br_y - tl_y)
```

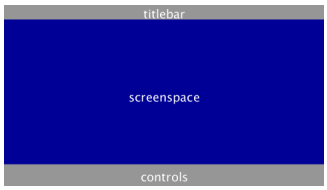
This code can be found in
`cmpt140-ch09-py/cmpt140_ch09_func_composition_b/cmpt140_ch09_func_composition_b.pyde`

(c) `screenspace( w, h, title_h, controls_h )` : returns the area of the usable (blue) part of the screen

`w, h` : the total width and height of the entire screen

`title_h` : height of the title bar

`control_h` : height of the control bar




Solution

```
# CMPT 140 - Functions with Outputs
# Topic(s): Function Composition
# Part (c)


def screenspace( w,h,title_h,controls_h ):
    """
    rectangular area of unoccupied screenspace
    i.e. area not occupied by titlebar or controls
    w: screen width
    l: screen height
    title_h: height of titlebar
    controls_h: height of controls
    """

    # compute total area
    total_area = w * h

    # compute occupied area
    titlebar_area = title_h * w
    controls_area = controls_h * w

    # compute and return unoccupied area
    return total_area - (titlebar_area + controls_area)
```

This code can be found in
`cmpt140-ch09-py/cmpt140_ch09_func_composition_c/cmpt140_ch09_func_composition_c.pyde`

## Nested Function Calls

**Exercise 3**
What value is returned by the following function calls?

(a)

```
def greatest_difference(a,b,c):
    return max(a,b,c) - min(a,b,c)

greatest_difference(3,-5,0)
```

**Answer:** 8 is returned by the function call. `max(3,-5,0)` returns 3 since it is the largest of the numbers while `min(3,-5,0)` returns -5 since it is the smallest of the numbers. The difference between 3 - -5 is 8.

(b)

```
def clamp_to_canvas(x):
    return max(0, min(x, 100))

clamp_to_canvas(150)
```

**Answer:** 100 is returned by the function call. To compute the value for `max()` in `max(0,min(150,100))`, the value of its argument `min(150,100)` needs to be computed first. `min(150,100)` returns 100 as that is the smaller of the two arguments. The value becomes the argument to `max(0,min(150,100))`, so now the function call looks like `max(0,100)`. 100 is the larger of the two arguments and becomes the final return value.

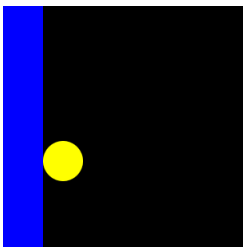
(c)

```
def num_chars_over( msg ):
    max_chars = 20
    num_chars_exceeded = max( 0, len(msg) - max_chars )
    return num_chars_exceeded

num_chars_over("Want to hang out this weekend?")
```

**Answer:** 10 is returned by the function call. `max_chars` is simply assigned to 20. `num_chars_exceeded` is then assigned to `max(0,len("Want to hang out this weekend?")-20)`, which becomes `max(0,30-20)` as `len("Want to hang out this weekend?")` returns the length of the string as 30. `max(0,30-20)` becomes `max(0,10)`. `num_chars_exceeded` refers to the larger of the two numbers 10 and becomes the overall function's return value.

**Exercise 4**

We want to draw a circle that follows the mouse. But the circle isn't allowed in the blue area:



Assume you already have this code to get started:

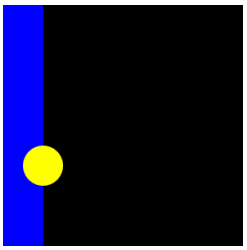
```
d = 50  # size of the mouse-following circle
border = 50  # furthest left circle centre can be drawn

def setup():
    size(300,300)
    noStroke()

def circle_radius(diameter):
    return diameter/2
```

```
def draw():
    background(0,0,0)
    # blue borders (circle can not be drawn in here)
    fill(0,0,255)
    global border
    rect(0,0,border,height)
    # draw the circle
    fill(255,255,0)
    global d
    ellipse(circle_x(mouseX,border,diameter),mouseY,d,d)
```

Your job: define the function `circle_x()`

computes and **returns** the x-coordinate to use for the circle's center based on the mouse position

**Hint 1:** You may need to use `max()` or `min()`
**Hint 2:** Beware of the following case. It's not allowed!



Solution

```
# CMPT 140 - Functions with Outputs
# Topic(s): Nested Function Call Composition

diameter = 50  # radius of circle to follow mouse with
left_border = 50  # furthest left circle centre can be drawn

def setup():
    size(300,300)
    noStroke()

def circle_radius(diameter):
    """ computes the radius of the circle given diameter
    diameter: the circle's diameter
    return: the circle's radius """
    return diameter/2

def circle_x(x,diameter,left_border):
    """ centre x-coordinate of the circle within drawable area
    diameter: circle's diameter in px
    left_border: the left boundary of the drawable area in px
    return: circle's centre x-coordinate """
    return max(x, left_border + circle_radius(diameter))

def draw():
    background(0,0,0)

    # blue borders (circle can not be drawn in here)
    fill(0,0,255)
    global left_border
    rect(0,0,left_border,height)

    # draw the circle
    fill(255,255,0)
    global diameter
    ellipse(circle_x(mouseX,diameter,left_border),mouseY,diameter,diameter)
```

This code can be found in `cmpt140-ch09-py/cmpt140_ch09_nested_composition/cmpt140_ch09_nested_composition.pyde`

**Design Demo: Global Variables**
We want to make a simple timer.

Count time in hours, minutes and seconds

Starting from 0

Let's see how to properly use and update variables to keep track of the time.

Solution

```
# CMPT 140 - Functions with Outputs
# Topic(s): Design with Global Variables

# Variables to keep track of the time
h = 0  # current hour
m = 0  # current minute
s = 0  # current second

def setup():
    size(300, 300)
    textSize(30)
    frameRate(1)

def update_time(hr, min, sec):
    """ updates the time by one second.
    returns the hour, minute and second (as integers) of the updated time

    Parameters:
    hr: the current hour
    min: the current minute
    sec: the current second.
    """
    # Note: we're not modifying global variables here.
    # The current values for hr,min,sec were passed in as *parameters*
    sec = sec + 1
    if sec == 60:
        sec = 0
        min = min + 1

    if min == 60:
        min = 0
        hr = hr + 1

    # This is our way of returning multiple values
    return hr, min, sec

def draw():
    global h, m, s
    background(0)
    # This is a top-level function.
    # Update the global variables h, m and s by calling
    # update_time() to get appropriate new values
    h, m, s = update_time(h, m, s)
    time = str(h) + ":" + str(m) + ":" + str(s)
    text(time, 110, 140)
```

This code can be found in `cmpt140-ch09-py/cmpt140_ch09_global_design/cmpt140_ch09_global_design.pyde`