Nesting Programming Constructs

Nesting Loops and Conditionals

Exercise 1
What does this function do?

(a)

```
def ex1a(pen_enabled, colour_enabled):

    if pen_enabled:
         if colour_enabled:
              fill(0,255,0)
              ellipse(50,50,50,50)
```

Answer: If the program is actively drawing to the canvas (`pen_enabled == True`) and it is drawing shapes in colour (`colour_enabled`), then it will draw a green circle of diameter 50 to coordinate (50,50). If either of the cases are `False`, nothing is drawn.

(b)

```
def ex1b(n_rows, n_columns, diameter)
    for row in range(n_rows):
        for col in range(n_columns):
            # current circle's centre coordinates
            circle_x = row*diameter+diameter/2
            circle_y = col*diameter+diameter/2
            # draw circle
            ellipse(circle_x,circle_y,diameter,diameter)
```

Answer: Draws `rows` by `columns` grid of circles in the top-left corner of the canvas, each `diameter` pixels in diameter. The intermediate variables `circle_x` and `circle_y` conveniently calculate and store the current circle's centre (x, y) coordinate.

(c)

```
def ex1c(pen_enabled, square_size):

    if pen_enabled:
        square_x = 0
        while square_x < screen_width:
            rect(square_x,0,square_size,square_size)
            square_x = square_x + square_size
```

Answer: If the program is actively drawing to the canvas (`pen_enabled == True`), then it draws a series of squares, starting at the leftmost edge, along the top of the canvas until no more visible squares can be drawn (a square reaches the right edge of the canvas or spills over to the outside of the window).

(d)

```
def ex1d(n_squares, square_size, gap_square):

    for square in range(n_squares):
        if square != gap_square:
            square_x = square*square_size
            rect(square_x,0,square_size,square_size)
```

Answer: Draws `n_squares` number of squares, starting at the leftmost edge, along the top of the canvas. However, it will skip drawing the `gap_square`th square in the series.

Exercise 2
Assume the following is pre-defined for you:

```
day                    # integer: representing day of the year
is_sky_clear           # boolean: whether any constellations
                       # can be seen in sky
n_constellations       # integer: total # of constellations
is_visible(i,j)        # function: returns boolean indicating
                       # constellation i's visibility at j
                       # hours past midnight
draw_constellation(i,j)  #function:  draws constellation i as
                           # seen at j hours past midnight
```

Write Processing code that will:

(a) If it is day 182 and the sky is clear, draw the sixth constellation at zero hours past midnight, i.e. `draw_constellation(6,0)`

(b) For each hour past midnight until 6AM (hours zero to six inclusive), draw all constellations at that hour

(c) If it is day 182, draw the second constellation in the databank at every hour past midnight until 6AM inclusive

(d) Draw all **visible** constellations captured at two hours past midnight

Solution

Below is the relevant portion of the code solution required to solve the exercise; the file contains the full code solution which includes the rest of the setup.

```
# set up the canvas for each part to have black night sky
size(250,250)
background(0,0,0)

# Part a)
if day == 182 and is_sky_clear == True:
    draw_constellation(6,0)

# Part b)
for hour in range(7):
    for constellation in range(1,n_constellations+1):
        draw_constellation(constellation,hour)

# Part c)
if day == 182:
    for hour in range(7):
        draw_constellation(2,hour)

# Part d)
for constellation in range(1,n_constellations+1):
    if is_visible(constellation,2):
        draw_constellation(constellation,2)
```

This code can be found in `cmpt140-ch13-py/cmpt140_ch13_nesting/cmpt140_ch13_nesting.pyde`

Multilayer Nesting

Exercise 3
What does this function do?

(a)

```
def ex3a(portal_activated, portal_visible, draw_effects):

    if portal_activated:
       if portal_visible:
           if draw_effects:
               fill(225,255,255)
           else:
               fill(0,0,0)
           ellipse(50,50,25,50)
```

Answer: If a portal is being actively drawn and is visible to the audience, then an ellipse representing the portal will be drawn at (50,50). If the drawing of special effects are enabled, then the portal will be coloured a light blue hue. Otherwise, the portal will be black in colour.

(b)

```
def ex3b(n_rows, n_columns, n_inner):

    for row in range(n_rows):
```

```
        for column in range(n_columns):
            for d in range(n_inner):
                x = column*30+15
                y = row*30+15
                diameter = 30-d*8
                ellipse(x,y,diameter,diameter)
```

Answer: Draws a `n_rows` by `n_columns` grid of circles in the two outermost loops. The innermost loop draws `n_inner` circles within each grid position such that each succeeding circle drawn on top of one another decreases in size.

(c)

```
def ex3c(n_rows, n_cols):

    for row in range(n_rows):
        for col in range(n_cols):
            if booth_status(row,col) == "Occupied":
                fill(255,0,0)
            elif booth_status(row,col) == "Open":
                fill(0,255,0)
            elif booth_status(row,col) == "Unavailable":
                fill(200,200,200)
            x = 20*col+10
            y = 20*row+10
            rect(x,y,15,15)
```

Answer: Draws a grid of expo (or exhibition) booths as squares and colours them based on their booking status. If a booth is booked ("Occupied"), it is coloured red. If a booth is available for booking ("Open"), it is coloured green. If a booth can not be booked (e.g. a pole is in the way), it is deemed "Unavailable" and coloured a light gray.

Exercise 4

Define a function called `layout_targets(difficulty)`

the parameter `difficulty` is a string with two possible values: `"moderate"` and `"challenging"`

the layout of the targets should depend on the difficulty as per the drawing below

you have access to 2 pre-defined functions: `draw_blue_target(row, col)` and `draw_red_target(row, col)`

row and column indices for those functions start at 0, not 1!

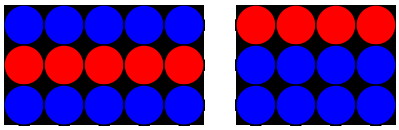

Figure 1: "moderate" layout (left) and "challenging" layout (right)

Solution

Below is the relevant portion of the code solution required to solve the exercise; the file contains the full code solution which includes the rest of the setup.

```
def draw():
    background(0,0,0)
    global difficulty
    # "moderate" difficulty has 3 by 5 grid of targets
    # such that (s.t.) even rows have blue targets, odd rows have red targets
    if difficulty == "moderate":
       for y in range(3):
           for x in range(5):
               if (y % 2 == 0):
                   draw_blue_target(x,y)
               else:
                   draw_red_target(x,y)
    # "challenging" difficulty has 3 by 4 grid of targets
    # s.t. first row has red targets, all other rows have blue targets
    elif difficulty == "challenging":
        for y in range(3):
            for x in range(4):
                if (y == 0):
                    draw_red_target(x,y)
                else:
                    draw_blue_target(x,y)
```

The full code solution can be found in
`cmpt140-ch13-py/cmpt140_ch13_multilayer_nesting/cmpt140_ch13_multilayer_nesting.pyde`