Conditional Branching

Relational and Logical Operators

If-statements

If-statements: Asking questions about data

If-statements

What kind of questions might we want to ask?

Relational Operators

> < <= >= ==

Exercise 1
Given the following variables:

```
x = 10
y = 2
z = 5
title = "How To Fly"
author = "Penguin"
publisher = "iLike2Publish"
```

Evaluate the following expressions:

(a) `y*z == x`

Answer: True

(b) `x > len(author)`

Answer: True

(c) `title < author`

Answer: True

(d) `author > publisher`

Answer: False (lowercase letters are sorted after uppercase letters)

Exercise 2
Given the following pre-defined variables:

```
fish_a              # string: fish A's name
fish_b              # string: fish B's name
max_name_length     # integer: maximum length for names
```

Write Python expressions to determine:

(a) Whether `fish_a` and `fish_b` refer to the same values

(b) Whether `fish_a` occurs after `fish_b` in lexicographical ordering

(c) Whether the length of `fish_b` exceeds `max_name_length`

(d) Whether the shorter (length-wise) of `fish_a` and `fish_b` is less than or equal to `max_name_length`

Solution

```
# CMPT 140 - Conditional Branching
# Topic(s): Relational Operators


# pre-defined variables
fish_a             # string of fish A's name
fish_b             # string of fish B's name
max_name_length    # int of max chars fish name can be

# Part a)
fish_a == fish_b

# Part b)
fish_a > fish_b

# Part c)
len(fish_b) > max_name_length

# Part d)
min(fish_a,fish_b) <= max_name_length
```

This code can be found in `cmpt140-ch11-py/cmpt140_ch11_relational_ops/cmpt140_ch11_relational_ops.pyde`

Boolean Operators

and or not

Exercise 3
Given the following variables:

```
a = True
b = False
c = True
```

What do the following Boolean expressions evaluate to (`True` or `False`)?

(a) `not a`

Answer: False

(b) `a and b`

Answer: `False`

(c) `b or a`

Answer: `True`

(d) `b and c or a`

Answer: `True`

(e) `not c or b`

Answer: `False`

(f) `not b and c`

Answer: `True`

(g) `b or a and not b and c`

Answer: `True`

Exercise 4

Given the following variables describing fish:

```
has_dorsal_fin      # whether fish has a dorsal fin
has_scales          # whether fish has scales
has_stripes         # whether fish has stripes
has_spots           # whether fish has spots
```

Use logical operators to write Python expressions to determine if a fish:

(a) Does not have scales

(b) Has both stripes and spots

(c) Has at least one of stripes or spots

(d) Is missing at least one of a dorsal fin or scales

(e) Has stripes or spots, but not both

Solution

```
# CMPT 140 - Conditional Branching
# Topic(s): Logical Operators


# pre-defined variables
has_dorsal_fin    # whether fish has a dorsal fin
has_scales        # whether fish has scales
has_stripes       # whether fish has stripes
has_spots         # whether fish has spots
has_hidden_gills  # whether gills are not readily visible

# Part a)
not has_scales

# Part b)
has_stripes and has_spots

# Part c)
has_stripes or has_spots

# Part d)
not (has_dorsal_fin and has_scales)

# Part e)
(has_stripes or has_spots) and not (has_stripes and has_spots)
```

This code can be found in `cmpt140-ch11-py/cmpt140_ch11_logical_ops/cmpt140_ch11_logical_ops.pyde`

Exercise 5
Given the following variables:

```
x = 10
y = 2
z = 5
```

Evalute the following expressions:

(a) `not x < y`

Answer: True

(b) `x > y and y > z`

Answer: False

(c) `x == z or x > z`

Answer: True

(d) `not (z < x and y < x)`

Answer: False

(e) `x+z > x+y or z < x and y < x`

Answer: True

```
(f) y < x-z or not x**y > y**x
```

Answer: True

```
(g) (x/y == z or x+z > z) and not (y*x < y+x)
```

Answer: True

Branching and Conditional Statements

Exercise 6
Write **a function** that checks if a given point is inside a box

function header: inside_box(x, y, box_x, box_y, w, h)

(x,y): the point we want to check

(box_x, box_y): top-left corner of the box

w, h: width and height of the box

return True if (x,y) is inside the box, False otherwise

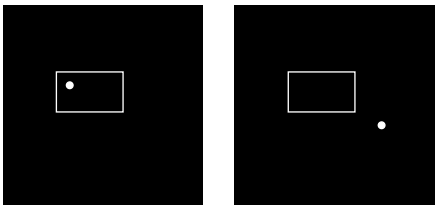


Figure 1: On left: inside the box. On right: outside the box.

Solution

```
# CMPT 140 - Conditional Branching
# Topic(s): If-Else Statements, Boolean expressions

x = -10
y = -10
box_x = 40
box_y = 50
box_w = 50
box_h = 30

def setup():
    size(150, 150)
    fill(0)
    stroke(255)

def draw():
    background(0)
    global box_x, box_y, box_w, box_h, x, y
    fill(0)
    rect(box_x, box_y, box_w, box_h)
    fill(255)
    ellipse(x, y, 5, 5)

def inside_box(x, y, b_x, b_y, w, h):
    if ( (x > b_x) and (x < (b_x + w)) and (y > b_y) and (y < (b_y + h)) ):
        return True
    else:
        return False

def mouseClicked():
    global box_x, box_y, box_w, box_h, x, y
    x = mouseX
    y = mouseY

    if (inside_box(mouseX, mouseY, box_x, box_y, box_w, box_h)):
        print("Inside!")
    else:
        print("Outside!")
```

This code can be found in `cmpt140-ch11-py/cmpt140_ch11_inbox/cmpt140_ch11_inbox.pyde`

Exercise 7

Suppose we are given functions that return a letter grade based on a numerical grade:

```
def letter_grade_1(grade):
  letter = "F"
  if grade > 85:
    letter = "A"
  if grade > 70:
    letter = "B"
  if grade > 50:
    letter = "C"
  return letter
```

```
def letter_grade_2(grade):
  letter = "F"
  if grade > 85:
    letter = "A"
  elif grade > 70:
    letter = "B"
  elif grade > 50:
    letter = "C"
  return letter
```

If 75 is given as an argument to both functions, what do they return and why?

Solution

`letter_grade_1(75)` returns `"C"` while `letter_grade_2(75)` returns `"B"`.

The difference in return values is due to the seemingly subtle difference between the code snippets. As you may have noticed, the code is identical between the two functions except that `letter_grade_2` uses chained `elif` statements instead of multiple `if` statements. If you'll recall from the readings, the kind of conditional statement in conjunction with the ordering of conditions may adversely impact your code.

`letter_grade_2` is composed of one long chain of conditional statements that accepts only one condition to be `True`, executes its code, and then skips all other succeeding statements. However, `letter_grade_1` is composed of multiple if statements and will check each one, even if multiple conditions are `True`, because each if-statement counts as its own conditional statement.

So, in both functions, `letter` is set to `"B"`. However, in `letter_grade_1`, `letter`'s referred value is overridden by the block of code whose condition `grade` $>$ `50` is `True` and thus it is set to its final return value of `"C"`.

Exercise 8

Pretend we're planting rare yellow flowers in a garden:

```
# the number of remaining flowers to plant
flowers_to_plant = 5

def setup():
    size(300,300)
    background(0,100,0)

def draw():
    return

def mousePressed():
    # draws flower at cursor, if able to
    global flowers_to_plant
    flowers_to_plant = plant_flower(flowers_to_plant)
```

We only have a maximum of five flowers available, represented by the variable `flowers_to_plant`.

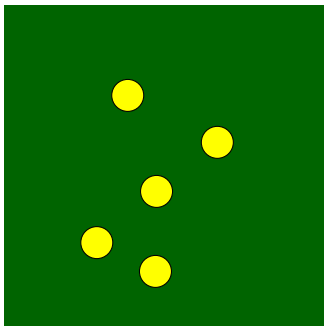
Your job: Define the function `plant_flower()`.

Gets called when the mouse is clicked

The "flower" is just a yellow circle

If there are no flowers left (remember, we only have 5!), print a message to the console instead of placing a new flower

Returns the remaining number of flowers left to plant



Solution

```
# CMPT 140 - Conditional Branching
# Topic(s): If-Else Statements


# the number of remaining flowers to plant
flowers_to_plant = 5

def plant_flower(flowers_to_plant):
    """ draws a flower to the screen if there
    are still flowers left to plant
    flowers_to_plant: number of flowers (if any) left to be planted
    return: number of remaining flowers to plant
    """

    # if there are still flowers to plant, draw one
    if (flowers_to_plant > 0):
        fill(255,255,0)
        ellipse(mouseX,mouseY,30,30)
        flowers_to_plant = flowers_to_plant - 1
    # ran out of flowers, print out message saying so
    else:
        print("Sorry, planted all the flowers!")

    return flowers_to_plant

def setup():
    # green grassy background
    size(300,300)
    background(0,100,0)

def draw():
    return

def mousePressed():
    # draws flower at cursor, if able to
    global flowers_to_plant
    flowers_to_plant = plant_flower(flowers_to_plant)
```

This code can be found in `cmpt140-ch11-py/cmpt140_ch11_if_else/cmpt140_ch11_if_else.pyde`

Interlude: Colour Variables

Processing gives us a convenient way to store an RGB colour in a variable.

```
shade = color(200, 100, 0)
```

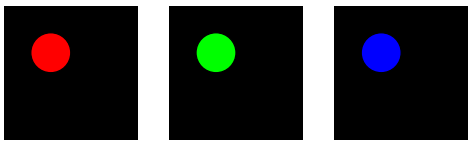
We can then use the variable `shade` as an argument to colour-related functions (like `fill()` and `stroke()`).

Exercise 9
Write Processing code which draws a **coloured** circle that follows the mouse.

initially, the circle is red

the user can change the circle's colour by pressing the 'r' (for red), 'g' (for green) or 'b' (for blue) keys

if any other key is pressed, display an error message on the console.



Solution

```
# CMPT 140 - Conditional Branching
# Topic(s): If-Elif-Else Statements


circ_colour = color(255,0,0)

def setup():
    background(0,0,0)

def draw():
    background(0,0,0)
    global circ_colour
    fill(circ_colour)
    ellipse(mouseX,mouseY,30,30)

def keyPressed():
    global circ_colour
    if key == "r":
        circ_colour = color(255,0,0)
    elif key == "g":
        circ_colour = color(0,255,0)
    elif key == "b":
        circ_colour = color(0,0,255)
    else:
        print("Invalid colour")
```

This code can be found in `cmpt140-ch11-py/cmpt140_ch11_if_elif_else/cmpt140_ch11_if_elif_else.pyde`

Extra Practice

Given the following **integers** describing fish:

```
n_dorsal_fins       # number of dorsal fins the fish has
n_scales            # number of scales the fish has
n_stripes           # number of stripes the fish has
n_spots             # number of spots the fish has
n_gills             # number of gills the fish has
```

Write Python expressions to determine if a fish:

(a) Does not have stripes

(b) Has more spots than stripes and has at least one dorsal fin

(c) Has either more spots than stripes or has exactly nine stripes

(d) Does not have more than two dorsal fins and does not have more stripes than spots

(e) Has at least as many stripes as spots and more than twelve stripes, or has more spots than stripes though no more than thirteen spots

Solution

```
# CMPT 140 - Conditional Branching
# Topic(s): Logical and Relational Operators


# pre-defined variables
n_dorsal_fins     # number of dorsal fins the fish has
n_scales          # number of scales the fish has
n_stripes         # number of stripes the fish has
n_spots           # number of spots the fish has
n_gills           # number of gills the fish has

# Part a)
not n_stripes > 0

# Part b)
n_spots > n_stripes and n_dorsal_fins >= 1

# Part c)
n_spots > n_stripes or n_stripes == 9

# Part d)
not n_dorsal_fins > 2 and not n_stripes > n_spots

# Part e)
(n_stripes >= n_spots and n_stripes > 12) or\
  (n_spots > n_stripes and not n_spots > 13)
```

This code can be found in `cmpt140-ch11-py/cmpt140_ch11_mixed_ops/cmpt140_ch11_mixed_ops.pyde`

There are many different ways to express Boolean expressions, so the code solutions presented should not be considered the only possible answers. For instance, with part a), it is possible to forgo the `not` operator and write `n_stripes == 0` to check if a fish does not have stripes. Just be careful that your logic is sound and expresses what you want it to!