Poke Version 2 Reflection Activity

Q1 The code for the decide_continue method from Poke Version 1 is given in the top pane. Write the code for the decide_continue method in Poke Version 2 in the bottom pane. How has the code changed in the new version?

Poke Version 1

```
def decide_continue(self):
        # Determine if the game should continue
        # - self is the Game to update

        pass
```

Poke Version 2

```
if self.small_dot.collide(self.big_dot):
   self.continue_game = False
```

checks if dots have collided
assigns new value false to identifier self.continue_game if they have

Q2 List every operator(if any) and type of operand for the following code segment:

Code Segment	Operators and Operand Types
def intersects(self,other_dot):	
distance_x = self.center[0] - other_dot.center[0]	int - int
distance_y = self.center[1] - other_dot.center[1]	int - int
distance = math.sqrt(distance_x**2+distance_y**2)	int ** int + int ** int
return distance <= self.radius + other_dot.radius	float <= int + int

Q3 Replace the if statement in the given code segment with a single return statement so that the two code segments produce the same result.

Given Code Segment

```
def isCold(temp):
    if temp <= 10:
        return True
    else:
        return False
```

Equivalent Code Segment

```
def isCold(temp):
   return temp <= 10
```

Q4 For each of the following code segments indicate the output, assuming that the identifiers not_raining and sunny are both bound to True.

Code Segment	Output

`if not_raining:`
`   if sunny:`
`       print(‘Play outside’)`
	Play outside

`if sunny:`
`   if not not_raining:`
`       print(‘Rainbow’)`
`else:`
`   print(‘Blue Sky’)`	

`if not sunny or not_raining:`
`   print('Cloudy')`
`else:`
`   print('Watch TV')`
`print(‘Windy’)`
	Cloudy
	Windy

Q5 The following table gives a list of expressions used in the move method of the Dot class and the purpose they are being used for. Match each expression (in the Expression column) to its purpose (in the Purpose column).

	Expression	Purpose
1	`self.center[0] < self.radius`	`check if dot has moved past bottom edge`
2	`self.center[0] +self.radius > size[0]`	`check if dot has moved past left edge`
3	`self.center[1] < self.radius`	`reverses horizontal direction of the dot`
4	`self.center[1] + self.radius > size[1]`	`reverse vertical direction of the dot`
5	`self.velocity[0] = -self.velocity[0]`	`check if dot has moved past right edge`
6	`self.velocity[1] = -self.velocity[1]`	`check if dot has moved past top edge`

Q6 Modify the given code for the draw_score method in the Game class such that the score is displayed at the top right corner of the window, in font size 100, in green foreground color on a black background.

Given Code

```
def draw_score(self):
     # Draw the time since the game began as a score
     # in white on the window's background.
     # - self is the Game to draw for.

     text_string = 'Score:' + str(self.score)
     text_fg_color = pygame.Color('white')
     text_font = pygame.font.SysFont('', 70)
     text_image = text_font.render(text_string, True, text_fg_color, self.bg_color)
     text_top_left_corner = (0, 0)
     self.surface.blit(text_image, text_top_left_corner)
```

Modified Code

```
def draw_score(self):
  text_string …
  text_fg_color = pygame.Color(“green”)
  text_font = pygame.font.SysFront(‘’ , 70)
  text_image …
  a = self.surface.get_width()
  b = text_image.get_width()
  text_top_right_corner = (a - b, 0)
  self.surface.blit(…
```

Q7 Write the Python code for drawing Hello in red 70 font size on green background at the bottom left corner of the window whose surface is bound to an identifier w_surface.

```
string = “Hello”
fg_color = pygame.Color(“red”)
bg_color = pygame.Color(“green”)
font = pygame.font.SysFront(“”, 70)
text_box = font.render(string, True, fg_color, bg_color)
x = 0
y = w_surface.get_height() - text_box.get_height()
location = (x, y)
w_surface.blit(text_box, location)
```