**Poke Version 1 Reflection Activity**

**Q1 For each of the following underlined and bolded functions used in Poke Version 1, indicate which is a regular function and which is a method. For EACH function or method, indicate whether it is built-in, user-defined or imported. For EACH function or method, also indicate what is the type of object returned by the function or method.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Function or Method** | **Built-in,User-defined or imported** | **Type of Object** |
| surface = **create\_window()** | **function** | **User defined** | **pygame.Surface** |
| game = **Game(surface)** | **function** | **User defined** | **Game** |
| **game.play()** | **method** | **User defined** | **NoneType** |
| **pygame.quit()** | function | imported | NoneType |
| **pygame.init()** | function | imported | NoneType |
| surface = **pygame.display.set\_mode(size, 0, 0)** | Function | imported | pygame.Surface |
| **pygame.display.set\_caption(title)** | function | imported | NoneType |
| self.bg\_color = **pygame.Color('black')** | method | imported | pygame.Color |
| self.small\_dot = **Dot(pygame.Color('red'), [50,75], 30, [1,2], surface)** | function | User defined | Dot |
| **self.draw()** | method | User defined | NoneType |
| **self.update()** | “” | “” | “” |
| **self.decide\_continue()** | “” | “” | “” |
| **time.sleep(self.pause\_time)** | function | imported | NoneType |
| event = **pygame.event.poll()** | function | imported | pygame.event? |
| **self.big\_dot.move()** | method | User defined | NoneType |
| **self.surface.fill(self.bg\_color)** | method | imported | NoneType |
| **self.small\_dot.draw()** | method | User defined | NoneType |
| **pygame.display.update()** | function | imported | NoneType |
| **pygame.draw.circle(self.surface, self.color, self.center, self.radius)** | function | imported | NoneType |
| size = **self.surface.get\_size()** | method | imported | pygame.Surface.Size? |
| for coord in **range**(0,2) | function | Built in | list |

**Q2 Use the following code fragment from Poke Version 1 to answer the questions.**

def play(self):

# Play the game until the player presses the close box.

# - self is the Game that should be continued or not.

self.draw()

while not self.close\_clicked:

# play frame

self.handle\_event()

if self.continue\_game:

self.update()

self.decide\_continue()

self.draw()

time.sleep(self.pause\_time)

|  |  |
| --- | --- |
| 2.1 | What object is self.close\_clicked bound to when the play method is called from the main function for the first time? |
|  | False |
| 2.2 | For what condition (write the identifier and its value) will the body of the while loop not be evaluated? |
|  | Self.close\_clicked == True |
| 2.3 | How does the handle\_event method in the Game class impact the evaluation of the condition of the while loop? |
|  | It checks to see if the event type is equal to QUIT, and if so, sets close\_clicked to True |
| 2.4 | What object is self.continue\_game bound to when the play method is called in the main function for the first time? |
|  | True |
| 2.5 | What effect does the update method have on the Dot objects? |
|  | It changes their center attributes |
| 2.6 | What single assignment statement can be included in the decide\_continue method of the Game class so that the condition of the if statement becomes False? |
|  | Self.decide\_continue = False |
| 2.7 | Why is the draw method in the Game class called twice in the play method? |
|  | The first time is to draw the initial position of all objects, the second time is to draw the next frame on the screen |
| 2.8 | Why is the sleep function in the time module called in each iteration of the while loop? |
|  | The game sleeps so that the while loop isn’t constantly running, and the player has time to see what their previous action/what the game previously did. |

**Q3 List all of the instance attributes of the Game class in Poke Version 1. For each attribute, identify what type that identifier is bound to.**

|  |  |
| --- | --- |
| **Instance Attributes** | **Type** |
| **surface**  **bg\_color**  **pause\_time**  **close\_clicked**  **self.continue\_game**  **self.blue\_dot**  **self.red\_dot** | **pygame.Surface**  **pygame.Color**  **Int**  **Boolean**  **Boolean**  **Dot**  **Dot** |

**Q4 The given code changes the entries in the dimensions list object by using the entries in the change list object. Modify the given code so that it uses a repetition statement and the built-in range function to produce the same output that the given code produces.**

|  |
| --- |
| **Given Code** |
| dimensions = [200,700,500,900]  change = [2,4,6,8]  dimensions[0] = dimensions[0] + change[0]  dimensions[1] = dimensions[1] + change[1]  dimensions[2] = dimensions[2] + change[2]  dimensions[3] = dimensions[3] + change[3]  print(dimensions) |
| **Modified Code** |
| dimensions = [200,700,500,900]  change = [2,4,6,8]  For index in range(len(dimensions)):  dimensions[index] = dimensions[index] + change[index]  **print(dimensions)** |

**Q5 For each of the following Python code fragments from Poke Version 1, list every operator and the type of the operand(s).**

|  |  |
| --- | --- |
| **Python Statement** | **Operator(s) and type of operand(s)** |
| while not self.close\_clicked: | **Operator: not Operand:self.closed\_clicked** |
| if event.type == QUIT: | **Operator: == Operand:event.type, QUIT** |
| self.center[coord] = (self.center[coord] + self.velocity[coord]) % size[coord] | **Operator: =, +, % Operand:**self.center[coord],  Self.velocity[coord], size[coord] |

**Q6 For each of the following classes, listed in the Class column, choose and match a line of code,listed in the Initialization Statement column, that is used to obtain a new instance of that class.**

|  |  |  |
| --- | --- | --- |
|  | **Class** | **Initialization Statement** |
| **1** | **Game** | 3. surface = pygame.display.set\_mode(size, 0, 0) |
| **2** | **pygame.Color** | 6. pygame.draw.circle(self.surface, self.color, self.center, self.radius) |
| **3** | **pygame.Surface** | 1. game = Game(surface) |
| **4** | **pygame.event.Event** | 2. self.bg\_color = pygame.Color('black') |
| **5** | **tuple** | 4. event = pygame.event.poll() |
| **6** | **pygame.Rect** | 5. size = self.surface.get\_size() |