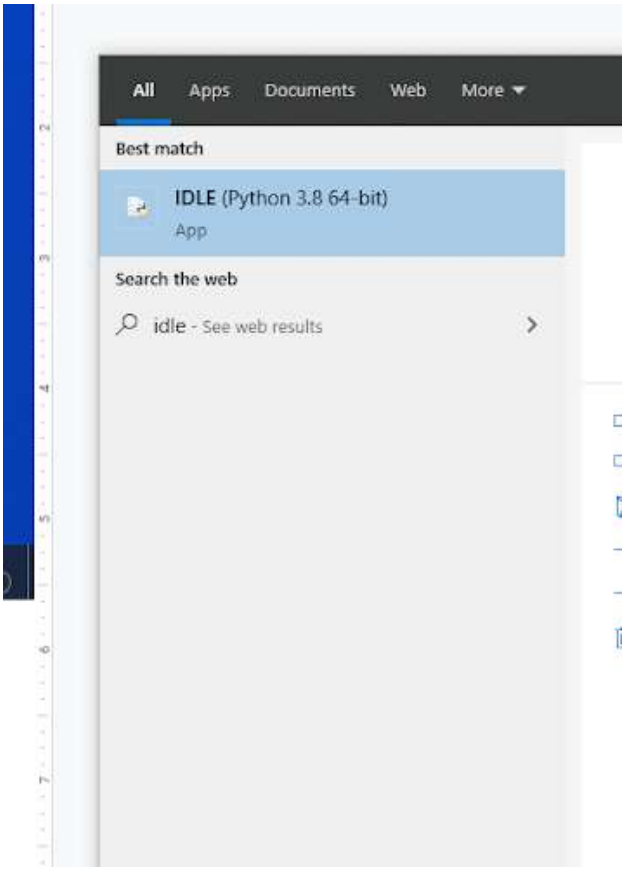
Chapter 7 Practice Directions

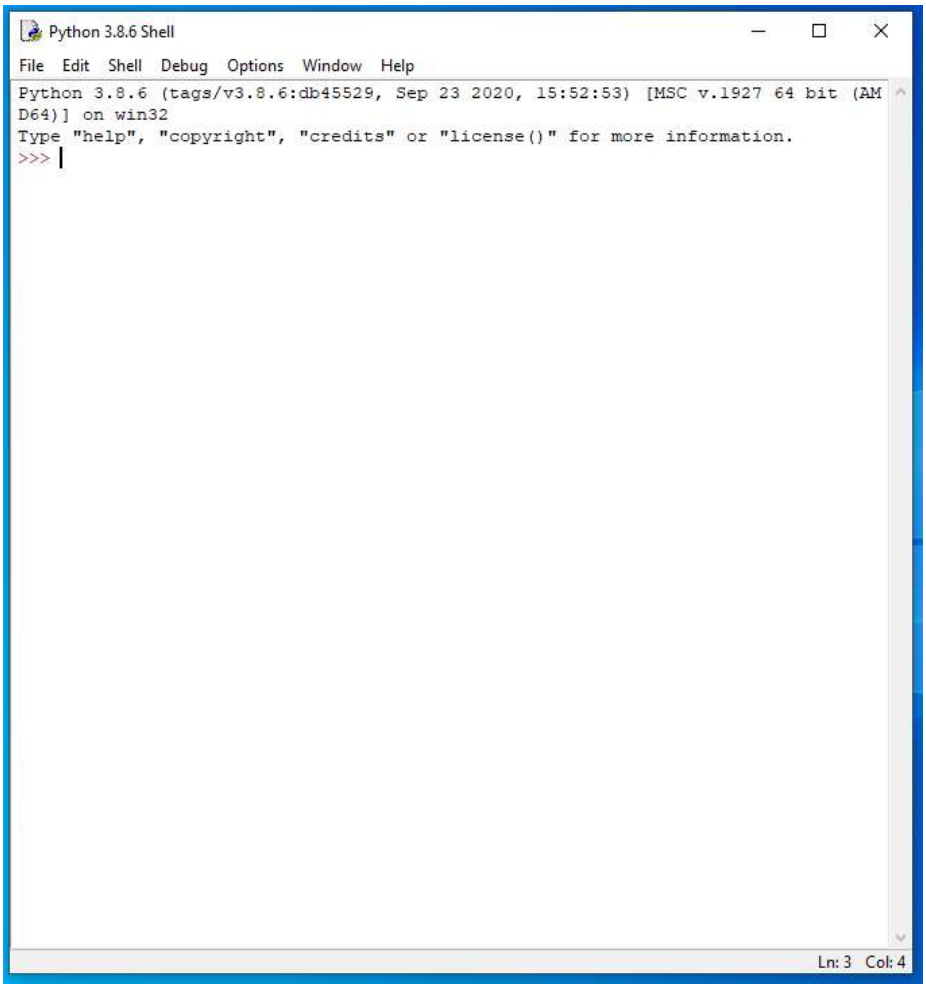
1. . Using your Windows button menu, find and launch your IDLE program.



IDLE is the integrated development environment associated with Python. It is made up of a code editor where you type your code along with other helpful tools that allow you to write, save, and test run programs.

IDLE is designed to recognize Python code, compile Python code, and provide basic debugging tips to programmers if there are problems with their code.

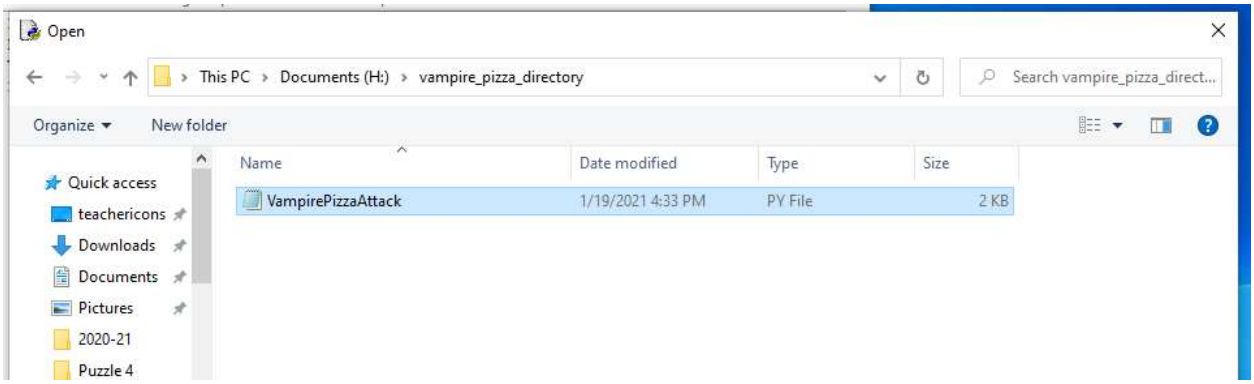
2. Your IDLE window should look something like this once it has launched.:



On Startup, IDLE will display the Python Shell, which can be used to give commands to the computer's operating system. Since we are viewing the shell through IDLE and not the actual command prompt window, the commands that we type into the Shell will not communicate directly with our operating system. However, you can type similar commands in the Python Shell directly from the Python program (not through IDLE) and, if you have permission to access the operating system's commands, you can communicate with the computer's operating system that way.

In IDLE, the shell is mainly used as a launching screen for other activities that we will do, like writing code for our game or debugging a file.

3. Go to File > Open and then browse to find your VampirePizzaAttack file that we created last chapter and open it.



4. Your Python file and code from last chapter will open up.

5. We will now begin to code the next part of our game. I like to make my coding window larger so that I can see all of my code a bit better, but that is a personal decision. Remember as we move through these exercises that your spelling, capitalization, and indentation should match. If it doesn't, your program likely won't work.

6. Click at the end of Line 7.

```
#Import Libraries
import pygame
from pygame import *
from random import randint

#Initialize pygame
pygame.init()
```

7. Press ENTER twice.

8. Type the code that you see on Lines 9 - 10 of the screenshot below, then press ENTER:

```
6  #Initialize pygame
7  pygame.init()
8
9  #set up clock
10 clock = time.Clock()
```

   The code on Line 9 contains a comment.

   The code on Line 10 creates a variable called clock. The variable's value is set to be equal to the value of the time.Clock() method. Every Pygame file comes with the time module built in, and the time module is actually a collection of various different objects/methods that can be used for different timing components of games. Each object/method within the time module provides a variety of different functionality options. In this case, Line 10 sets up a new clock object in the game and assigns that object to the variable called clock. The clock object gives the programmer the ability to track time, in seconds and milliseconds, for anything in the game.

9. Click at the end of Line 28, as shown in the screenshot below.

```
#Define colors
WHITE = (255, 255, 255)

#Set up rates
SPAWNRATE = 360
```

10. Press ENTER.

11. Type the code that you see on Line 29 in the screenshot below.

```
24 #Define colors
25 WHITE = (255, 255, 255)
26
27 #Set up rates
28 SPAWNRATE = 360
29 FRAMERATE = 60
```

   Line 29 creates a new constant variable called FRAMERATE and sets its value to 60. We will use this variable later in the code when we want to tell the game how many frames to display per second.

12. Click at the end of Line 66, as shown in the screenshot below.

```
        self.rect = self.image.get_rect(center = (1100, y))

    #This function moves the enemies from right to left and destroys them after they've left the screen
    def update(self, game_window):
        game_window.blit(self.image, (self.rect.x, self.rect.y))
```

13. Press ENTER.

14. Type the code that you see on Lines 67 – 68 of the screenshot below.

```
65      #This function moves the enemies from right to left and destroys them after they've left the screen
66      def update(self, game_window):
67          game_window.blit(BACKGROUND, (self.rect.x, self.rect.y), self.rect)
68          self.rect.x -= self.speed
69          game_window.blit(self.image, (self.rect.x, self.rect.y))
```

The code on Line 67 creates a command to erase a sprite. The game_window.blit portion of the command specifies the surface to display the image on. Within the parentheses, the BACKGROUND variable is the image that will be displayed (the background image). You also see the location to display the image at (the location of self.rect.x and self.rect.y). Finally, the final argument in the parentheses on Line 67 tells the program to display only the image in that particular rectangular area that the sprite was at. Essentially, you are covering up a sprite with a background image rectangle of the same size in the same location.

Line 68 adjusts the position of the sprite's rectangle to move along the negative x-axis by 2 pixels, since the self.speed variable is set to 2.

The code on Line 69, which you typed last chapter, will then display another sprite at the new, updated x-coordinate location.

15. Click at the end of Line 117, as shown in the screenshot below:

```
#-------------------------------------------------------------
    #Update display.
    for vampire in all_vampires:
        vampire.update(GAME_WINDOW)

    display.update()

#Close main game loop
#-------------------------------------------------------------
```

16. Press ENTER twice.

17. Type the code that you see on Lines 119 – 120 of the screenshot below.

```
117     display.update()
118
119     #set the framerate
120     clock.tick(FRAMERATE)
```

    Line 119 contains a comment.

    Line 120 calls the clock.tick function using the FRAMERATE argument. The clock.tick function measures the time that has passed, in milliseconds, since it was last called. Our FRAMERATE variable that we created earlier in this exercise is set to 60. By entering the optional argument of FRAMERATE at the end of the method, we tell the clock that it should pace the game loop so that it runs the tick method no more than 60 times per second. The .tick function will pace the game so that there are 60 frames that pass each second.

    Remember that with each loop of the game, we move the vampire pizza sprites to the left by 2 pixels (as specified in our self.speed variable and the code on Line 68). Since this game loop will loop 60 times each second, and it is moving the sprite by 2 pixels each loop, our vampire sprites should move 120 pixels to the left each second.

18. Go to File > Save Now to save your game.

19. Go to Run > Run Module to preview your game. After it loads, you should notice your enemy sprites start to move to the left gradually.

20. You may now close out of IDLE and the game window and submit this assignment.

Final Code:

```
#Import Libraries
import pygame
from pygame import *
from random import randint

#Initialize pygame
pygame.init()

#set up clock
clock = time.Clock()

#-----------------------------------------------------------------
#Define constant variables

#Define the parameters of the game window
WINDOW_WIDTH = 1100
WINDOW_HEIGHT = 600
WINDOW_RES = (WINDOW_WIDTH, WINDOW_HEIGHT)

#Define the tile parameters
WIDTH = 100
HEIGHT = 100

#Define colors
WHITE = (255, 255, 255)

#Set up rates
SPAWNRATE = 360
FRAMERATE = 60

#-----------------------------------------------------------------
#Load assets
```

```
#Load assets

#Create window
GAME_WINDOW = display.set_mode(WINDOW_RES)
display.set_caption('Vampire Pizza')

#Set up the background image
background_img = image.load('restaurant.jpg')
background_surf = Surface.convert_alpha(background_img)
BACKGROUND = transform.scale(background_surf, (WINDOW_RES))

#Set up the enemy image
#Load the image into the program
pizza_img = image.load('vampire.png')
#Convert the image to a surface
pizza_surf = Surface.convert_alpha(pizza_img)
VAMPIRE_PIZZA= transform.scale(pizza_surf, (WIDTH, HEIGHT))

#-------------------------------------------------------------
#Set up classes
#Create an enemy class
class VampireSprite(sprite.Sprite):

    #This function creates an instance of the enemy
    def __init__(self):
        super().__init__()
        self.speed = 2
        self.lane = randint (0, 4)
        all_vampires.add(self)
        self.image = VAMPIRE_PIZZA.copy()
        y = 50 + self.lane * 100
        self.rect = self.image.get_rect(center = (1100, y))

    #This function moves the enemies from right to left and destroys them after they've left the screen
    def update(self, game_window):
        game_window.blit(BACKGROUND, (self.rect.x, self.rect.y), self.rect)
        self.rect.x -= self.speed
        game_window.blit(self.image, (self.rect.x, self.rect.y))

#-------------------------------------------------------------
#Create class instances

#create a sprite group for all VampireSprite instances
all_vampires = sprite.Group()

#-------------------------------------------------------------
#Initialize and draw the background grid

#Define the color of the grid outline
tile_color = WHITE

#Populate the background grid
for row in range(6):
    for column in range(11):
        draw.rect(BACKGROUND, tile_color, (WIDTH*column, HEIGHT*row, WIDTH, HEIGHT), 1)

#Display the background image to the screen
GAME_WINDOW.blit(BACKGROUND, (0, 0))

#-------------------------------------------------------------
#Start main game loop

#Game loop
game_running = True
while game_running:
```

```
#-------------------------------------------------------------
#Create VampireSprite instances
    if randint (1, SPAWNRATE) == 1:
        VampireSprite()

#-------------------------------------------------------------
    #Update display.
    for vampire in all_vampires:
        vampire.update(GAME_WINDOW)

    display.update()

    #set the framerate
    clock.tick(FRAMERATE)

#Close main game loop
#-------------------------------------------------------------

#Clean up game
pygame.quit()
```